G-4
19400658

9400658

SAI 9400658

CONFERENCE PROCEEDINGS

REFERENCE COPY

RAND

Proceedings of Conference on Variable-Resolution Modeling, Washington, DC, 5–6 May 1992

Paul K. Davis, Richard Hillestad, (eds)

29737

DO NOT DESTROY
30 DAYS LOAN
RETURN TO AFSAA/SAI
PENTAGON, RM 1D383
425

National Defense Research Institute

20101015254

G-4
19400658

Legacy Number: 9400596 19400596

The research described in this report was sponsored by the Defense Advanced Research Projects Agency under RAND's National Defense Research Institute, a federally funded research and development center supported by the Office of the Secretary of Defense and the Joint Staff, Contract No. MDA903-90-C-0004.

The RAND Conference Proceedings series reproduces conference papers and proceedings exactly as they were submitted and delivered at the conference. RAND is a nonprofit institution that seeks to improve public policy through research and analysis. Publications of RAND do not necessarily reflect the opinions or policies of the sponsors of RAND research.

Published 1992 by RAND
1700 Main Street, P.O. Box 2138, Santa Monica, CA 90407-2138

H-15 19400658

CONFERENCE PROCEEDINGS

RAND

Proceedings of Conference on Variable-Resolution Modeling, Washington, DC, 5–6 May 1992

Paul K. Davis, Richard Hillestad, (eds)

CF-103-DARPA

Prepared for the
Defense Advanced Research Projects Agency

National Defense Research Institute

Approved for public release; distribution unlimited

PREFACE

This document collects papers delivered at a conference on variable-resolution modeling and the related issue of developing integrated hierarchies of models. The conference was sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Defense Modeling and Simulation Office (DMSO). It was co-hosted by RAND and the University of Arizona.

Preparation for the conference and development of this proceedings was accomplished in the Applied Science and Technology program of RAND's National Defense Research Institute, a federally funded research and development center sponsored by the Office of the Secretary of Defense and the Joint Staff.

SUMMARY

One of the most important scientific challenges associated with modern-day modeling and simulation is learning how to design models or, quite often, model *families*, so that workers can move from one level of resolution to another as needed. Major advances have recently been made in connecting dissimilar models, including models differing in resolution, but these advances have had more to do with software technology than with substantive validity and usefulness. That is, it is one thing to connect models of different resolution so that they "run." It is quite another to have this connection be meaningful and useful. For this one needs to have a deep scientific understanding of the relevant phenomena, the approximations and representations used by the models in question, and appropriate ways to calibrate models against each other or external data. By and large, however, there has been relatively little in-depth discussion of such matters in the defense community. The conference documented here represented a first attempt to remedy this situation.

The first paper, by Paul Davis of RAND, introduces the subject theoretically, providing definitions, motivations, an approach emphasizing integrated hierarchical modeling, and a worked-out example illustrating classic difficulties and prospective solutions.

The next two papers, by Frederick Eddy of General Electric and Bernard Zeigler of the University of Arizona, describe object-oriented design techniques,which deal effectively with some aspects of variable-resolution modeling, primarily by encouraging hierarchical design of simulation objects. The paper by Timothy Horrigan of Horrigan Analytics highlights a class of conceptually difficult and potentially very important problems that arise in attempting to develop aggregate descriptions of physics-level interactions in which "spatial configuration" plays an important role. The paper by Reiner Huber of Germany's Armed Forces University then describes one of the motivations for developing models of varied resolution, the desire to provide aggregate perspectives and ability to ask What if? questions appropriate to general-officer discussion, while assuring consistency with more detailed modeling.

The second group of papers reflects a number of applications and particular experiences. Peter Cherry of Vector Research describes (viewgraphs only) Vector's experience in working with a corps-level and a theater-level model over a period of some years. Mr. Roy Reis of Air Force Studies and Analysis reviews (viewgraphs only) standard problems that have been encountered in cross-resolution work over the years. Patrick Allen of RAND illustrates with a highly specific air-force problem the difficulties associated with connecting appropriately a deterministic low-resolution model with a stochastic higher

resolution model. Mr. Gunther Scheckler describes the extensive experience of the German IABG in developing and maintaining a relatively well cross-calibrated hierarchy of ground-combat models. Mr. David Frelinger's paper sketches RAND experience with an Air Force hierarchy of models and notes conceptual difficulties associated with developing an improved hierarchy. The paper by Michael Mattock of RAND sketches a logic-programming approach to modeling and its application to strategic mobility modeling. Walter Perry and John Schrader of RAND describe a multi-resolution model of command and control in disaster management. Keith Brendley and Jed Marti, also of RAND, describe ongoing work to connect meaningfully models of very different resolution running on a distributed network.

The third group of papers turns to theory. Richard Hillestad and Mario Juncosa of RAND describe in some detail, working with a Lanchester problem, the conditions under which certain aggregations can and cannot be accomplished rigorously. This work bears on the perennial issue of whether score-based methods can reasonably be used in aggregate models. The paper by Richard Hillestad, Donald Blumenthal of Livermore, and John Owen of Great Britain's Defense Operational Analysis Establishment (DOAE) then describe a series of cross-organization experiments with overlapping models of the same problem, a simple ground-combat battle of armored forces. They demonstrate a wide range of problems that are probably quite typical of those encountered in attempting to aggregate or to compare models developed in different organizations. The papers are also humbling in that they candidly acknowledge that even highly sophisticated modelers can be insensitive to very serious problems or disconnects until the problems manifest themselves dramatically, as in the comparisons they describe. In retrospect, the problems are all "trivial," but there is reason to believe that there are many such "trivial" problems throughout the community. The paper by Bruce Fowler of the Army Missile Command suggests the potential of a partial-differential-equation representation of many combat problems. One important conclusion, which was echoed by other papers, was that success depends critically on recognizing the natural phases of battle and treating them separately. The paper (viewgraphs only) by Darrel Fowler of SRI International, who argues (viewgraphs only) that many of the critical issues in variable-resolution design can be understood in terms of general systems theory. Julian Palmore's paper deals with an aspect of variable resolution that is usually is ignored, notably the issue of computational resolution, which is a critical factor underlying common problems such as structural variance or apparent chaos in discrete-event simulations.

The fourth group of papers addresses model "environments." Bernard Zeigler describes an approach based in general systems theory that emphasizes hierarchical and modular design methods that makes reusability of modules straightforward. Christopher

Landauer of Aerospace reviews a number of software tools and techniques that improve capabilities to connect and evaluate dissimilar models. Randy Brown of the Wright Laboratory sketches basic features of the J-MASS approach to modular object-oriented programming with an emphasis on reusable objects (viewgraphs only). The next paper, by Candace Conwell, Larry Peterson and Richard Freund of the Naval Command, Control, and Ocean Surveillance Center, describes an environment being developed there (viewgraphs only). Michael Bailey of the Naval Postgraduate School discusses a scientific method of choosing model fidelity. Barry Silverman of George Washington University describes issues of cost and productivity in model software, which involve the challenge of making software reusable.

The conference ended with a discussion of lessons learned and what next to do. One consensus conclusion was that much more work now needs to be done on stochastic aggregated models. Aggregated models are essential, but they often need to be stochastic because the very issues of most concern to policymakers, general officers, and other recipients of analysis may be very poorly described by point-value approaches. The obvious example here is that battles seldom end with "average" outcomes. Instead, one side wins and the other side loses. Another consensus was that the "configuration problem" discussed by Horrigan (and illustrated in various ways also by Davis and Hillestad) is severe. There is no generally understood theory on how and when to aggregate. Instead, there is a widespread tendency to assert intuitively plausible aggregated relationships that may, upon inspection, prove quite mischievous. A third consensus was that in a next conference it would be desirable to have fewer papers, circulated in advance, and discussed in depth. The problems at issue are very difficult and the community is at a stage at which working through representative problems in detail is exceptionally valuable. It may be possible to develop a text book describing the relevant theory and methodologies in a year or so, but at this stage we are still collecting insights and seeking ways to organize our thinking. Success is very important, however, because it is a prerequisite for success in distributed simulation and distributed war gaming.

ACKNOWLEDGMENTS

We acknowledge the valuable contributions and partnership of Professor Bernard Zeigler in making possible the conference.

CONTENTS

Preface	iii
Summary	v
Acknowledgements	ix
An Introduction to Variable-Resolution Modeling and Cross-Resolution Model Connection (Paul K. Davis, RAND)	1
Object-Oriented Modeling: Holistic and Variable-Resolution Views (Frederick Eddy, General Electric)	44
Modular Hierarchical Model Representation (Bernard P. Zeigler, University of Arizona)	52
The "Configuration Problem" and Challenges for Aggregation (Timothy J. Horrigan, Horrigan Analytics)	102
Crossing Levels of Resolution in Defense Analysis: A Requirement of the New Strategic Environment (Reiner K. Huber, Federal Armed Forces University, Germany)	154
Applications of High- and Low-Resolution Models in Cross-Resolution Analysis (W. Peter Cherry, Vector Research, Incorporated)	171
Hierarchy of Models for Conventional Air Power Analysis (Roy H. Reiss, Air Force Studies and Analyses Agency)	198
Combining Deterministic and Stochastic Elements in Variable-Resolution Models (Patrick D. Allen, RAND)	209
A Hierarchy of Models at Different Resolution Levels (Günther Scheckeler, IABG/WSP, Germany)	215
An Approach to Hierarchies of Models: Process Independence (Edward R. Harshberger, Bart E. Bennett, and David R. Frelinger, RAND)	231
A Distributed Network Approach to Variable-Resolution Modeling (Keith W. Brendley and Jed Marti, RAND)	248
Cutting Some Trees to See the Forest: On Aggregation and Disaggregation in Combat Models (Richard J. Hillestad and Mario L. Juncosa, RAND)	256

Experiments in Variable-Resolution Combat Modeling 293
(Richard Hillestad, RAND; John Owen, British Defence Operational Analysis Establishment, England; and Donald Blumenthal, Lawrence Livermore National Laboratory)

Resolution Changes and Renormalization for Partial Differential Equation Combat Models 332
(Bruce W. Fowler, U.S. Army Missile Command)

A Mathematical Technique for Analysis and Derivation of Models of Varying Resolution 341
(Darrell V. Fowler, Information Systems Engineering Center)

Variable-Resolution Modeling in Mathematics 362
(Julian Palmore, USACERL and University of Illinois)

A Systems Methodology for Structuring Families of Models at Multiple Levels of Resolution 377
(Bernard P. Zeigler, University of Arizona)

Integrated Simulation Environments 409
(Christopher Landauer and Kirstie L. Bellman, The Aerospace Corporation)

Joint Modeling and Simulation System 432
(Randy Brown, Wright-Patterson Air Force Base)

Structured Model Environments 441
(Candace Conwell, Larry J. Peterson, and Richard F. Freund, Naval Command, Control, and Ocean Surveillance Center)

The Scientific Method of Choosing Model Fidelity 448
(Michael P. Bailey, Naval Postgraduate School)

The Software Engineering Paradox: Unexploited Cost Savings and Productivity Improvements 456
(Barry G. Silverman)

A Variable-Resolution Approach to Modeling Command and Control in Disaster Relief Operations 465
(Walter L. Perry, John Y. Schrader, and Barry Wilson, RAND)

1. INTRODUCTION

This paper is an introduction to the subject of variable-resolution modeling (VRM) and the closely related issue of developing integrated families of models with varied resolutions. It addresses the following questions: (1) What *is* variable-resolution modeling?, (2) Why might one want it?, (3) What forms can it take and how does it relate to "families of models"?, and (4) How should one go about it? After providing definitions and some basic concepts,[1] I work through an extremely simple but concrete combat modeling problem to illustrate generic issues. A theme here is that the usual approach to model building, even if undertaken professionally, results in a diversity of independent models with different but overlapping resolutions, models that are difficult to use together because of both obvious and subtle differences in perspective, assumption, and definition. To move across levels of resolution readily it is highly desirable to have designed for that in the first place, or to pay the price of redesigning existing models so that they are truly integrated. Lashing models together without such redesign is likely to cause trouble or require substantial experience, skill, and time on the part of the analyst. Lashups may run, but using them efficiently and understanding the results is a different matter. After illustrating issues and some general methods using the simple models, I conclude with suggestions for both designing from scratch and redesigning existing models to integrate them in a family.

---

[1]See also Appendix A, which provides background for the study and gives citations to some of the more relevant academic literature.

2. DEFINITIONS AND BASIC CONCEPTS

DEFINITIONS

To proceed we need some definitions. The most important for the purposes of this paper are:

> *Variable-resolution modeling:* building models or model families so that users can change readily the resolution at which phenomena are treated (either by "turning resolution knobs" within a single model or turning from one model to another in a family).
>
> *Cross-resolution model connection:* linking existing models with different resolutions (the linkage may be in software so that the models operate together or "external" in which case outputs of one are transferred manually to become inputs of another).
>
> *Seamless design* permits changing resolution with (a) smooth consistency of representation (description) and (b) consistency of prediction.

The distinction here is that the term "variable resolution modeling" applies to designing a *new* model (or family of models). "Cross-resolution model connection" applies when talking about combining *existing* models of different resolution, models that were not originally designed to be combined. Cross-resolution work is often a matter of "coping." Seamlessness in this context means that when one changes resolution, either within a single model or by moving from one model to another within a family, one can do so without mental disruptions and with some confidence that the results are consistent in a sense to be discussed later. While we cannot aspire to continously variable resolution analogous to the zooming of a camera, we *can* aspire to models that allow us to make graduated changes in resolution that are easy to follow and, within limits, valid.

The reason for discussing variable-resolution modeling and cross-resolution model connection together is that the latter can often be accomplished best if one steps back and pretends to have the luxury of starting over again. After understanding what one would *like* to have, one can then make a set of reasonable model adaptations once and for all, rather than starting a process of sequential patches.

SUBTLETIES IN THE CONCEPT OF RESOLUTION

"Resolution" is usually treated as a primitive concept, but it is in fact rather subtle as Fig. 1 suggests. Indeed, we use the same word for very different concepts. For example:

1. A model may have higher resolution because it deals with more fine-grained entities (e.g., companies rather than battalions).[2]
2. A model may have higher resolution than another with the same entities because it ascribes to those entities a richer set of attributes. For example, the companies in one model may be characterized by firepower, and in another model by detailed weapon holdings. Or, to use a different example, targets may be described as having a wavelength-dependent spectral radiance rather than a mere brightness.

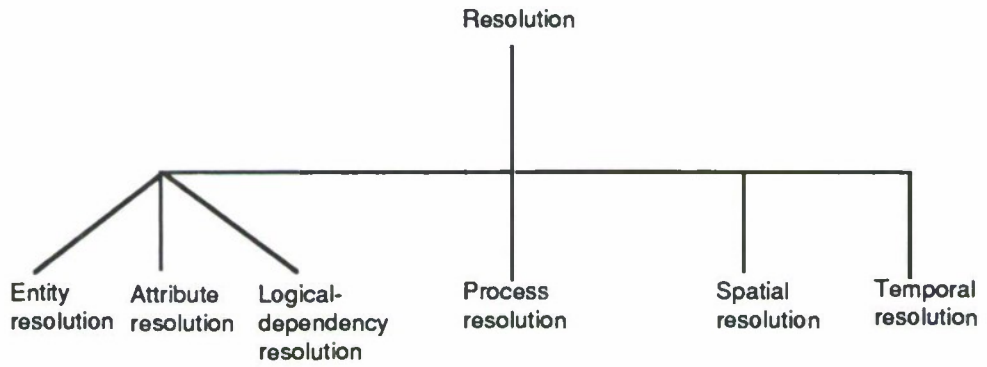


Fig. 1—Aspects of Resolution

3. A model with the same entities and attributes as another may have higher resolution because it describes the relationships among those attributes in more detail (called logical dependencies here). For example, one model may describe the spatial relationship among ground-combat units in rich detail dependent on circumstances, while another may assume that the units always align themselves in a standard formation.
4. Models agreeing in all of the above respects may differ in the resolution of the physical and command-control processes governing changes in entity attributes. One may follow individual battalions, but may assess attrition at the corps level and assume that the attrition is allocated evenly among battalions on the front line. A closely similar model may instead assess attrition for each individual battalion as a function of its particular battle situation.
5. Finally, the spatial grid and the time step, or the discrete-event equivalent, are also dimensions of resolution.

---

[2]This paper uses "objects" and "entities" interchangeably; similarly, "attributes" and "variables." The "processes" that cause changes in attribute or variable values may be implemented at the level of computer code by "functions" called, in object-oriented programming, "methods."

It may also happen that one model has higher resolution than another in some respects, but lower resolution in others.[3] We all "know" that the relative resolution of two models can be ambiguous, but we often ignore this complexity when talking loosely. In practice, this ambiguity of relative resolution is a common problem. When connecting existing models, one often discovers that the allegedly low-resolution model is actually richer in some respects than the high-resolution model. This leads to both technical and sociological problems (as when the organization charged with the high-resolution work has developed a model with some components that are lower in resolution than those of another organization charged with low-resolution modeling).

CONSISTENCY OF PREDICTION

To conclude this section on definitions, let us discuss the important concept of consistency of prediction, which will henceforth be called just consistency. Fig. 2 indicates schematically what may be called consistency in the aggregate. Suppose the system being modeled starts (top left) at time T1 in state S(T1). Suppose further that a detailed model exists, which can be represented by a time-generation operator **G(T2;T1)**. The system changes state from time T1 to T2 in a way that can be denoted $S(T2) = \mathbf{G(T2;T1)}S(T1)$.

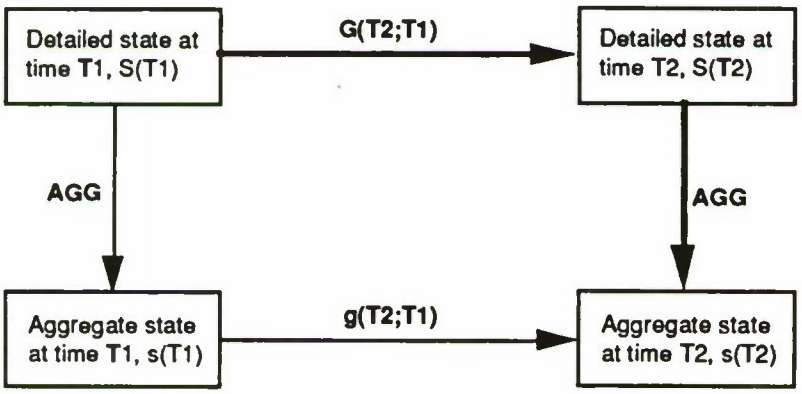


Fig. 2—Consistency in the Aggregate

Suppose, now, that we are interested in a particular aggregation of system characteristics. This might correspond in physics to taking the average over a volume. In combat models it might correspond to aggregating over divisions to get a corps-level depiction

[3]Many combat models have inconsistent resolution in that phenomena of interest to the developing organization are treated in more detail than others. So it is that the Air Force often has higher resolution in air-to-air combat than in ground combat and the Army often treats the air war cursorily if at all. For some applications such inconsistency is acceptable; for others it is not.

of strength. In any case, if we denote the aggregate state by s(T) and the aggregation operator (e.g., one that integrates or adds) by **AGG**, we have (starting from the top left, moving rightward, and then downward):

$$s(T2) = \mathbf{AGG}\,S(T2) = \mathbf{AGG}\;\mathbf{G(T2;T1)}S(T1).$$

But suppose instead we had aggregate the initial state and then used an aggregate model to generate the time behavior of the aggregate system (i.e., suppose we had moved downward and then rightward from the initial state). Would we end up with the same assessment of s(T2)? That is, would we find:

$$\mathbf{AGG}\,\mathbf{G(T2;T1)}S(T1) = \mathbf{g(T2;T1)}\,\mathbf{AGG}\,S(T1)\ ?$$

If so, we could say that there is complete consistency in the aggregate.

Fig. 3 suggests a stronger version of consistency. In this diagram we can ask whether the same detailed state at time T2, S(T2), can be generated by both the detailed model and by the process of aggregating, generating the time dependence of the aggregate state, and then disaggregating. The question is:

$$\mathbf{G(T2;T1)}S(T1) = \mathbf{DISAGG}\,\mathbf{g(T2;T1)}\,\mathbf{AGG}\,S(T1)?$$

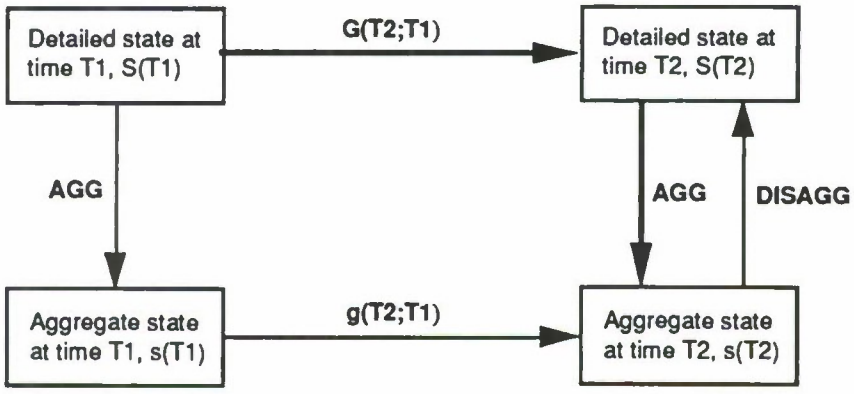


Fig. 3—A Stronger Version of Consistency

If so, we could say that the models are completely consistent. Clearly, this level of consistency would be unusual, because aggregation usually eliminates essential

information.[4] There are, however, real-world examples in which we have extra information that permits us to aggregate, generate time behavior, and then disaggregate without losing information. A familiar example from physics is the one of two falling bodies that just happen to be rigidly attached to one another. One can aggregate to look at the center-of-mass characteristics, follow the dynamics of that "effective" object, and then disaggregate to specify where the two physical objects are. An example in the military domain involves army units, which may go through complicated maneuvers while assembling, moving from point A to point B, and then dispersing for combat. So long as it is reasonable to assume that the combat formation is dictated by the circumstances at point B, a combat model could aggregate the forces for movement from point A to point B (i.e., discarding information on their configuration) and then disaggregate at point B before combat begins.

---

[4]One reviewer opined that the most egregious problems of aggregation and disaggregation occur in assessing attrition. He cited an example in which widely different aircraft are aggregated into a pool of aggregated sorties, which then are broken up into sorties for each of the several air-force missions and subjected to aggregation attrition processes. Then the losses are allocated to the various original types of aircraft, with the result that aircraft for close-air-support end up suffering attrition from deep interdiction missions that they would never perform in the real world.

3. WHY VARIABLE RESOLUTION IS IMPORTANT

GENERAL REASONS

Having defined concepts such as consistency across levels of resolution, the next question might be why one would even *want* variable-resolution models (or equivalent families of models). This deserves a more lengthy discussion (e.g., Davis and Huber, 1992), but some of the principal reasons are as follows:

We need low-resolution modeling for:

- Initial cuts (innovation, exploration,...)
- Comprehension (seeing the forest rather than the trees)
- Systems analysis and policy analysis
- Decision support
- Adaptability
- Low cost and rapid analysis
- Making use of low-resolution knowledge and data.

We need high-resolution modeling for:

- Understanding phenomena
- Representing knowledge
- Simulating reality
- Calibrating or informing lower-resolution models
- Making use of high-resolution knowledge and data.

There are subtleties. First, there is confusion between comprehending a model and comprehending phenomena. It is often essential to use high-resolution models to understand phenomena qualitatively (e.g., to discover what the critical factors are), but having obtained such an understanding, it may be desirable to use the simplest model consistent with that understanding to comprehend what one is doing analytically. Even brilliant people often make gross errors when dealing with models having too many variables and relationships.

Another subtlety is that resolution is a *relative* matter. Thus, just as theater-level analysis may require dipping into corps-level analysis selectively, so also corps-level analysis may require dipping into division-level analysis selectively, and so on. The problem is inherently hierarchical and one person's high resolution is another's low resolution. At every level, however, there are lower and higher resolution views with the advantages listed above.

Some workers argue that "infinite" computing power is on the horizon and will eliminate the need for lower-resolution models. Others argue that high resolution models are complex, incomprehensible, and to be avoided. *A more accurate view is that both low and high-resolution models and analysis will remain critical for the reasons given above, even as computer power continues to increase exponentially.*[5]

SOME EXAMPLES OF HOW VARIABLE RESOLUTION IS USEFUL

There are many current examples of how workers using combat models need to vary resolution. Some worth mentioning here are the following:

- Using high resolution to provide a *picture* when the lower-resolution depiction seems too abstract (e.g., to understand when maneuver and counter maneuver do and do not "cancel out")
- Invoking high resolution for *special processes* within the course of an otherwise low-resolution simulation (e.g., special processes in which physics-level phenomena are critical and cannot be well represented by averages; this arises, e.g., when one force has a distinct but situationally dependent qualitative advantage such as weapon range)
- Using high resolution to *establish bounds* for parametric analyses using lower-resolution models (e.g., bounds on the number of passes per sortie of a fighter-bomber)
- Using high (low) resolution to *calibrate* lower (higher) resolution models, recognizing that our knowledge of the world comes at all levels of detail[6] (e.g., calibration of killer-victim scoreboards based on a detailed physics-level weapon-on-weapon simulation)
- Using low resolution for *decision support, including rapid analysis of alternative courses of action* (e.g., battlefield decisions under enormous uncertainty, or

---

[5]As noted to me by Dr. Ralph Toms of Lawrence Livermore National Laboratory, this doesn't mean that in all applications one needs low-resolution models. Given, for example, well established high-accuracy computer codes for performing aerodynamic calculations, knowledge of the input parameters, and adequate computer capacity, it is often preferable to use those rather than attempting instead to apply simpler models through a series of idealizations, boundary-value tricks, and off-line analysis. In most applications of combat modeling, however, the detailed models are not well validated and the data required for them is highly uncertain (especially for combat conditions). In such applications, one wants a combination of low and high resolution models for the reasons indicated in the text.

[6]Organizations often gravitate toward a mindset in which lower resolution models are calibrated against higher resolution models, but not vice versa. This is wrong headed unless there is reason to believe the higher-resolution depictions—and the relevant data for them—are more reliable than lower resolution depictions and data.

peacetime acquisition decisions taken years before performance parameters and interrelationships can be fully understood)

- Using low resolution to *generate adaptive scenarios*, as when one attempts to provide and maintain context for higher resolution war gaming in which objectives and even tactics depend on higher level considerations and coordination requirements.

Needing variable resolution is one thing, but having it is another. There are many common difficulties. The first is that in attempting to use one model to provide a higher resolution view of the phenomena described in a second model, one often discovers that the models simply don't fit together. They may include inconsistent descriptions of the same phenomena, or the descriptions may be consistent only in a way that is quite obscure because of differences of perspective. This is particularly so with higher level analytic models, which necessarily exploit abstractions, because which abstraction is appropriate depends on the application.

Another problem is that when models are allegedly calibrated to each other, they are often calibrated at only a single point alleged to be representative. Little information is typically provided on how quickly the models get out of calibration. And, frankly, there is a lot of ad hocery. That is, the fitting of one model to another is as often as not done in off-line analysis that may not even be documented and, even if it is, would not pass muster in a mathematics class.

A basic motivation for RAND's related project work under DARPA sponsorship is that the issues involved in sound variable-resolution modeling are to a large extent generic, but are not well known—probably because design continues to be underemphasized in university curricula. With this background, then, let us next consider the types of variable-resolution modeling one might consider using and then begin moving toward generic concepts—first by working through a very simple example, and then by abstracting some principles from that example.

4. TYPES OF VARIABLE RESOLUTION MODELING

As discussed in some length in Davis and Huber (1992), it is useful to distinguish among three approaches to variable-resolution modeling in the community. These are summarized in Fig. 4.

Selected viewing consists of providing aggregate displays from a more detailed underlying simulation. It is quite valuable and should be encouraged, but it often has problems (e.g., lack of transparency and hidden-variable effects), including an implicit dependence on a mass of high resolution input data that may be very difficult or impossible for the user to review.

Alternative submodels is the most common form of variable-resolution modeling. By and large, however, the alternative submodels and data bases are inconsistent to varying degrees. While this approach is common and quite useful, it is often inelegant and full of seams. There is, in practice, a wide variation in the quality of variable-resolution models of this type.

	Common Problems
• Selected viewing —Carry along full resolution —Display lesser resolution as appropriate	• Quality depends on invisible underlying model and data • Hidden variable problems • Interactivity requires disaggregation, usually done ad hoc at interface
• Alternative submodels (or model families) —Models have switches —Submodels have different resolution —Submodels may or may not be integrated	• Inconsistencies • Differerent perspectives • Lots of seams • Point-case calibrations, if any
• Integrated hierarchical variable resolution (IHVR)	• Requires good design • Not always possilbe • Choice of hierarchies depends on perspective; limits later choices • Can be burdensome

Fig. 4—Classes of Variable Resolution

The third approach, *integrated hierarchical variable-resolution modeling* (IHVR), is quite uncommon as a consciously chosen approach, but I have used the method successfully in my own work. By IHVR I mean modeling that describes critical processes (e.g., attrition, movement, or air attack of ground targets) as being composed hierarchically of subordinate processes. These processes may be fully represented, in which case they generate

dynamically the inputs to the higher level processes. Alternatively, they may be replaced, in an approximation, with trivial processes that always provide the same constant inputs to the higher level processes. That is, they may be approximated by parameters.[7]

Importantly, most hierarchical families of models are not integrated and do not have the advantages associated with IHVR design. Instead, most model hierarchies are simply a collection of models of varied resolution along with some procedures for using the more detailed models to calibrate selected input parameters of the lower resolution models of the hierarchy. This calibration is often painful, inelegant, imprecise, and even dubious. In other cases, the approach is basically sound but painful. Documentation on how the various models of the hierarchies are calibrated against each other is unusual.

We shall work through an example of IHVR in what follows, but Fig. 5 provides a first glimpse at what is involved.

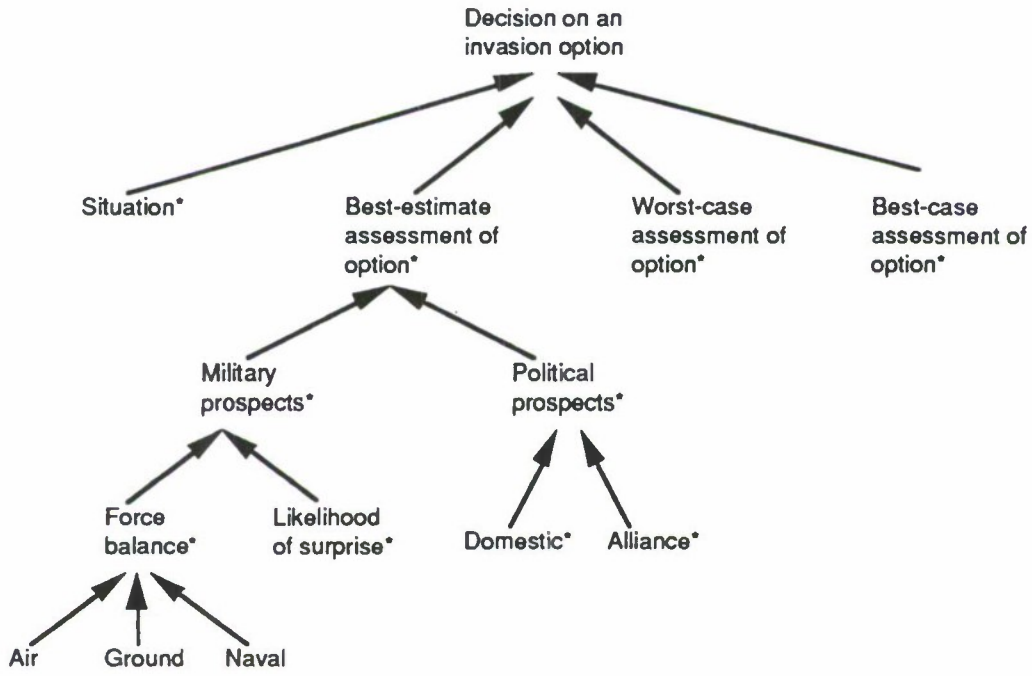


Fig. 5—An Example of Hierarchical Design: Political Decision Models

This figure has nothing to do with combat modeling; instead, it depicts the relationship among variables in a political model developed for work on deterring opponents in crisis (adapted from Davis and Arquilla, 1992). Each node is a class of variables. When

[7]Alternatively, they may be approximated by very simple processes that generate parameter values that vary with gross situation as defined by a lookup table.

two or more arrows feed into a node from below, it means that the higher-level variable is determined by some function of the lower-level variables. In a computer version, each node would be a function producing the variable shown (or a set of variables).[8] For simplicity, I have only elaborated the tree of variables for the Best-estimate assessment, but there should be similar trees for the other intermediate variables.

With this convention the top-level variable here is a decision by a potential invader, who is deciding which of several strategic options to pursue (e.g., no invasion with a reliance instead on bellicose threats, a limited invasion, or a full-scale invasion). This decision depends, for each option, on an assessment of the situation and on best-estimate, worst-case, and best-case (most-optimistic-case) assessments of the option in question. These assessments, in turn, may depend on lower-level variables such as the likelihood of surprise attack.

Note that the variables are arranged in a perfect hierarchy: each variable affects only variables above it in a single tree. There is no cross-talk between branches and no cycling (i.e., no feedback).[9] This makes it straightforward to implement variable-resolution modeling as follows.[10] Wherever one sees an asterisk, one can build in an option to generate the variable from lower-level variables (higher resolution) *or* to specify the variable directly as a parameter. This model is an example of IHVR. An exceptionally important aspect of IHVR is that one can "see" the relationships among variables in the different levels of resolution. This is a basic element of "seamlessness:" one can change levels of resolution without confusion (especially if the variables are appropriately named).

---

[8]This IHVR approach was used earlier to build large artificial intelligence models of potential Soviet and U.S.leadership reasoning in crisis and conflict (Davis, 1987; Davis, Bankes, and Kahan, 1986). The hierarchical design was critical, because some applications required linking the models to detailed simulation of combat, while others were more profitably accomplished using the higher level aspects of the political models independently, in which case inputs were specified as parameters.

[9]Feedback can be modeled in this type of structure with a time delay. Thus, the military prospects assessed at the end of one time period may determine the political prospects in the next period.

[10]I mean here that the *structure* of the problem is straightforward. It may or may not be the case that the more detailed variables change over time in such a way that their effects on higher-level functions can be approximated adequately by using average values.

5. A WORKED-OUT EXAMPLE

SIMPLE COMBAT MODELS PERMITTING MATHEMATICAL ANALYSIS

In a preliminary workshop on variable resolution held at RAND in November, 1991, participants expressed the need for simple examples that could be worked through in detail and fully understood. With that in mind let us now consider an exceptionally simple combat-modeling problem. The approach will be as follows: (1) to describe lower- and higher-resolution models developed quasi independently for the illustrative problem; (2) to examine whether they can be used together as a "hierarchical family of models" by calibrating the lower-resolution model against the higher resolution model; and (3) upon seeing and describing the difficulties in doing so, to describe how one could instead have used IHVR methods to develop the models in an integrated manner, and why doing so improves clarity and seamlessness. Consistent with what typically happens when attempts are made to connect existing real-world models, we will try initially to proceed without knowing the details of the models, but rather only qualitative descriptions (the models are defined in detail, however, in Appendix B).

The purpose, then, is first to give a case history of how it is that we so often end up with nonintegrated and confusing models, which we wish were easier to use together so that we could change resolutions at will; and, second, to describe a better way to proceed.

The problem we shall consider is simple ground combat between an attacker and a defender engaged in a straightforward head-on-head attrition battle, perhaps at the level of an army attacking a corps.

The nature of the low-resolution model is suggested in Fig. 6. In this depiction the forces are characterized in terms of overall attacker and defender "strengths," A and D, measured in equivalent divisions (or something more sophisticated such as the situationally adjusted equivalent-division scores described in Allen, 1992). As detailed in Appendix B, the model assumes that attrition depends solely on the force strengths and attrition coefficients K_a and K_d via the Lanchester square law and the so-called 3 to 1 rule. That is, the model assumes that the rate at which each side loses forces is proportional to the other side's strength. The 3:1 rule states that at an attacker-to-defender force ratio of 3 the sides will be fighting to a stalemate. The argument here is that the defender usually has advantages of prepared positions that provide both concealment and protection. While this is an extremely

simple model, it is one that has been used by many workers over the years because of its intuitive appeal.

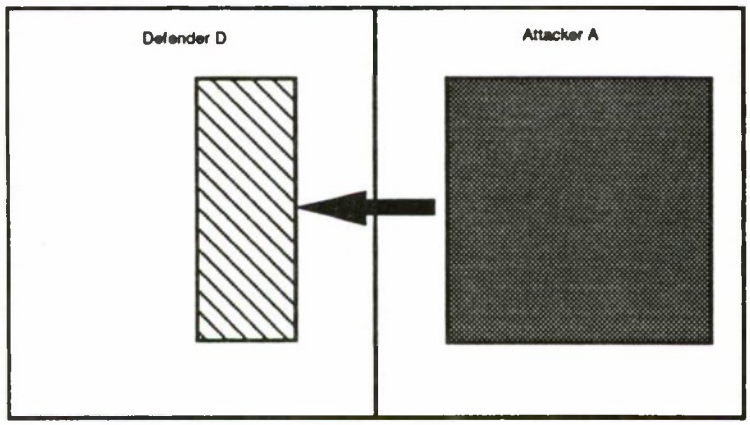


Fig. 6—An Aggregate Model of Ground Combat

Fig. 7 depicts what, for the sake of this example, can be considered as a "detailed" model—recognizing, of course, that it would be considered highly aggregate by those working at the weapon-on-weapon level, as discussed in the companion paper by Hillestad, Owen, and Blumenthal (1992), which goes down to the physics level of individual tanks shooting at individual tanks. The "detailed" model of Fig. 7 breaks the attacker and defender forces down into forces on the forward line of troops (FLOT, flank forces, and reserves. There is a military frontage L across which the battle takes place, there is a specific background of terrain suggested by the shading (e.g., open, mixed, or rough), and there is some type of defense set up by the defender (e.g., hasty, deliberate, prepared, or fortified), which together with the sides' tactics determine the "type battle." The model includes a concept of "break points," under which a side will break off fighting if its attrition is excessive or it is being severely outflanked. The model requires a great many more parameters than the previous model (see Appendix B). For simplicity, this model also assumes a Lanchester square law and the 3 to 1 rule for forces on the FLOT. That is, the same attrition mathematics applies to this model and the more aggregate one, but the level of detail is different and there may be different parameter values.[11]

[11]Even the simple model suggested in Fig. 6, coupled with Lanchester equations in one form or another, are commonly used. See, for example, Epstein (1990). One can also interpret the work of T.N. Dupuy in terms of Lanchester-square models (see Dupuy, 1987). The "detailed" model suggested in Fig. 7 is similar in many respects to those used to calculate daily attrition in a variety of theater-level combat simulations, including the RSAS (Bennett, Jones, Bullock, and Davis, 1988, and Allen, 1992). The RSAS, however, uses score-based methods for only short periods of time and uses coefficients that

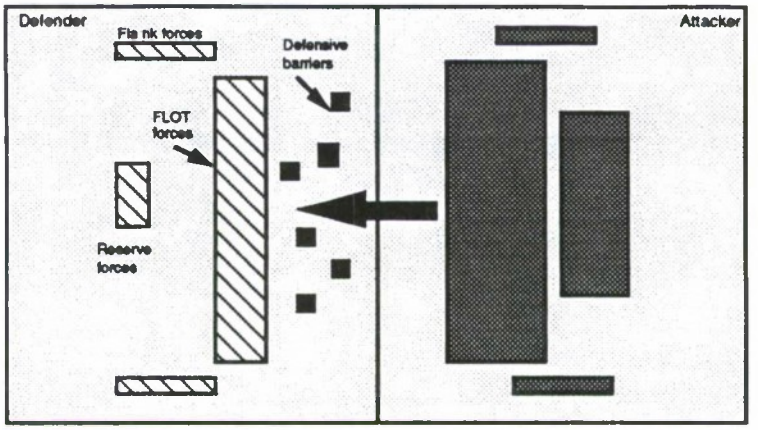


Fig. 7—A "Detailed" Model of Ground Combat

ATTEMPTING TO CREATE A HIERARCHICAL FAMILY FROM EXISTING MODELS

Principle Issues

Given the two models, then, suppose we declare them to constitute a hierarchical family. That is, we observe that one is more detailed than the other and we assert that the lower-resolution model can be calibrated by using the higher-resolution model. Is this true? Can we make them consistent and move from one to the other at will depending on the resolution we seek?[12]

The usual approach when this issue arises seems to be to run the models, compare the results, find a tuning parameter in the aggregate model, and adjust that parameter until results agree. The circumstances and details of that calibration procedure may or may not be recorded well and it may or may not make sense to tune the parameter in this way.

Here let us be a bit more careful. Let us assume that the detailed model is correct and see what is required to make the aggregate model reasonably consistent with it. Thus, the issue becomes one of calibrating, approximately, the attrition parameters K_a and K_d of the aggregate model. We do this by calculating the aggregate behavior of the system (the battle

are highly dependent on the operational situation faced by the forces, which changes from time period to time period. The algorithms used are also more complex than the Lanchester square law.

[12]The whole approach may also be dubious in that the detailed model may not be a reliable basis for calibration when one considers uncertainties in its input parameters and algorithms, but that is another matter.

between attacker and defender) using the detailed model, and then setting the aggregate attrition parameters so as to obtain roughly the same behavior with the aggregate model.[13]

Generating the Relevant Aggregate Behavior With the Detailed Model

As shown in Appendix B, the aggregate model's parameters K_a and K_d can be related to the dynamics of the battle as follows:

$$\frac{K_d}{K_a} = RLR\,F^2$$
$$K_a = -DLR / F,$$

where F is the attacker to defender force ratio, DLR is the defender loss rate (the fraction of the defender's strength lost in a day's battle), and RLR is the ratio of the attacker's and defender's loss rates. That is, if the attacker loses a fraction ALR of its strength per day and the defender loses a fraction DLR of its strength per day, then RLR=ALR/DLR. The time histories of F, DLR, and RLR (and also ALR) can be generated by running the detailed model. As discussed later and as can be appreciated from Appendix B, however, that may not be so straightforward because the variables of one or both models are not always what they seem to be and it is very easy to make serious errors of calibration. Nonetheless, let us assume that those problems have been avoided.

Calibrating Across Cases and Considering Analytic Context

If the aggregate model were exactly consistent with the detailed model, then K_d and K_a would be constants, but more generally the above equations can be valid only if we allow K_d and K_a to be time dependent. Since we want to make the aggregate model work as well as possible with *constant* attrition coefficients, we must now find good "average" values for $K_a(t)$ and $K_d(t)$ by conducting experiments with the detailed model. But what does this mean?

How do we compute averages? The most obvious and common way appears to be to consider a "representative case" (i.e., a particular scenario), including a particular set of all the input values that seems to be "typical." We shall use that method later, but note first that a more proper way to find a good average would be to consider a *range* of cases, or scenarios, and to develop some kind of weighted sum. We should also consider at what

[13] In the terms of Fig. 2, the detailed model corresponds to G and the aggregated model corresponds to g. We are attempting to see whether the aggregated state s can be more or less correctly generated over time by the aggregated model.

points in time we want the calibration to be best. It follows that we might use an expression such as:

$$\frac{K_d}{K_a} = \sum_{cases,i} W_i \frac{\int_0^{T_{max}} w(s)RLR(s)F^2(s)ds}{\int_0^{T_{max}} w(s')ds'}$$

$$K_a = -\sum_{cases,i} W_i \frac{\int_0^{T_{max}} w(s)\frac{DLR(s)}{F(s)}ds}{\int_0^{T_{max}} w(s')ds'}$$

where W_i denotes the weight given to case i (e.g., "scenario i") and w(s) is a weighting factor for time. The time-weighting would be significant if we were going to use the aggregate model in a context such that we needed it to be more accurate in some time periods than others (e.g., the first few days of a battle). Similarly, we might be interested only in times less than some T_{max}.[14]

In principle, we might set up a formal criterion such as choosing values of K_a and K_d that minimize the expected discrepancy between the consequences in a larger application of using the detailed and aggregate attrition equations for a specified set of cases and specified time intervals. This would then *imply* appropriate weighting functions. It should be evident, however, that attempting to do this type of thing is very complex and depends sensitively on application details. In practice, we must usually be satisfied with less than optimal choices of the coefficients, choices defined by relatively simple averages such as those shown, and using simple weighting factors (e.g., treat all cases within a particular set as equally important and ignore the others; similarly, treat all times as equally important for times less than T_{max}).

Even if we are able to define and take such averages, it may or may not make sense to do so. Do we have reason to believe that the distribution of values for the coefficients K_a and K_d will be "normal" around some natural average? Perhaps instead the distribution is

[14]It is common in simulation models to readjust many "constants" at the beginning of each time step or when certain major events occur such as a change in the nature of battle or the arrival of reinforcements. As a result, dynamical equations such as Lanchester expressions often need only to be reasonable approximations for a relatively short period of time. They should be calibrated with this in mind.

multimodal (see, e.g., Hillestad, Owen, and Blumenthal, 1992), in which case no single calibration for an "average" case may be useful.[15] Further, even if there is a natural average to be taken, what is it? What weighting factors should be applied for the cases and for different times? Should they be based on likelihood, importance, or both? Importance to what? How long should the averaging interval T_{max} be? The answer is that the "right" weighting and interval depend on the context in which the results of the modeled combat are to be used (e.g., upon force levels, frontages, etc., and, indirectly, on political-military scenario, strategies, and other variables). Regrettably, such important issues are seldom discussed when claims are made about one model having been calibrated against another.

Working Through a Representative Case

Let us now work through what might be called a "representative case" for corps-level battle. We will not bother with averages over cases. Instead, we will just do the calibration for this case, a dubious but standard practice. For simplicity, we also assume a simple treatment of reserves in which FLOT forces remain on the FLOT and reserve forces reinforce them subject to various constraints discussed later. We will also ignore the issue of flanks. There is no concept of units. Our example will assume 9 equivalent divisions (EDs) attacking 3 EDs across a 50 km frontage, with no flank forces and no break points. An "equivalent division" is a division that is as effective as a standard armored division, even though its composition may be significantly different.

Fig. 8 shows the "actual" behavior of $K_d(t)/K_a(t)$, which can be called the aggregate "defender advantage," and compares it with what would apply if behavior in the aggregate model were correct. That is, the curve marked "actual" is the result of using the detailed model to find the time-dependent ratio of aggregate model's defender advantage using the equation above. If the aggregate model were exact, the ratio $K_d(t)/K_a(t)$ would be constant at a value of 9 (see Appendix B). In fact, it is not constant, but averaging the "actual behavior" over the time interval shown produces an average value of 8.4, with a small standard deviation of only 1.1. The calibration of K_a yields a value of 0.026.

---

[15] As an example here, suppose that in some battles the defender was out in the open and in other battles he was behinded fortifications. "On average," across battles, the defender might barely hold his own, but in particular battles he might be decimated or might decimate the attacker. The "defender advantage," then (which name we give to K_d/K_a because of its effects on ratio of loss rates), is not well described by a single value.

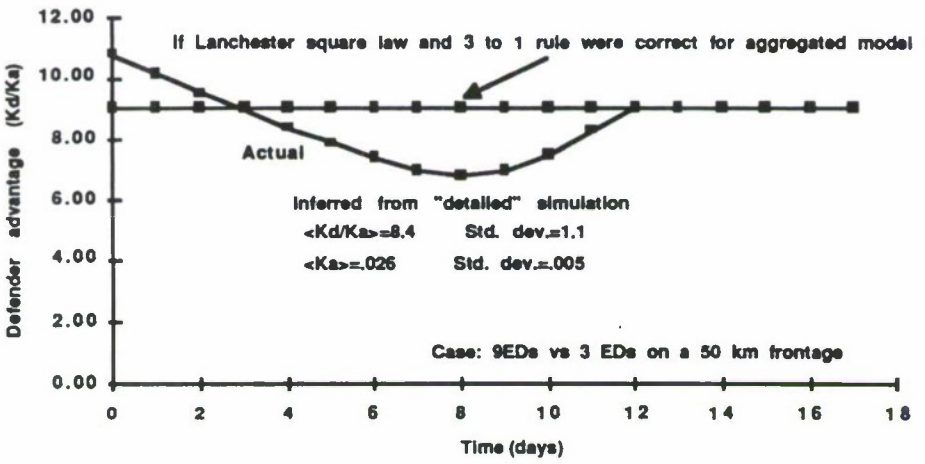


Fig. 8—A First Calibration of the Aggregate Model

How well does the calibration work? Fig. 9 shows, for the same case of 9 EDs vs 3 EDs, the time dependence of attacker and defender forces as predicted by the detailed and aggregate models. The agreement is excellent. Aggregation appears justified, at least as a good approximation.

Sensitivity Testing to See if the Calibration Holds

Now, however, let us try to use the model for a different case, one involving 25 EDs attacking 5 EDs, still on a 50 km frontage and with nothing but force levels changed. Fig. 10 shows the results. Here the agreement, using the same calibration as before, is miserable. If we look now at the aggregate defender advantage K_d/K_a over time for this new case (Fig. 11), we see that the value is even less nearly constant than for the previous case, and nothing like what the constant value of 9 that would apply if the aggregate model were exact. Obviously, the "representative case" wasn't all that representative. Or, to put it otherwise, the calibration was not robust. When dealing with more complex models, especially when they are being treated as black boxes, people are often surprised to find that calibrations don't work very well—i.e., that they prove not to be robust. Further, this may be discovered only after a long time, because there may have been little initial sensitivity testing after the initial calibration based on a "representative case." Initial sensitivity testing is especially difficult with large and complex models.

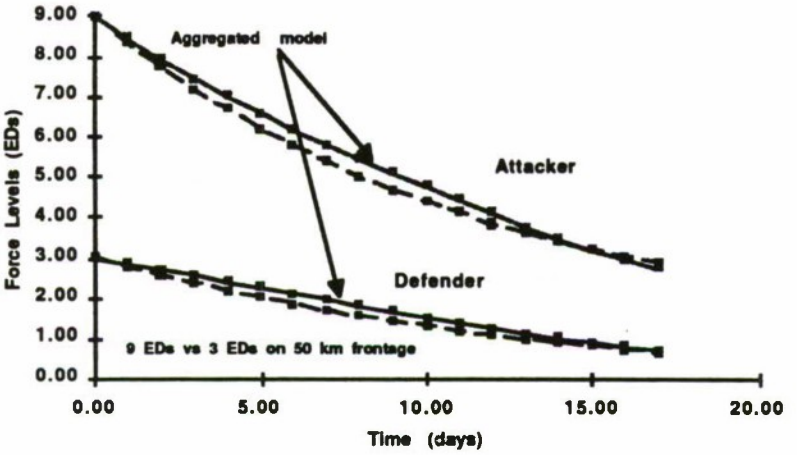


Fig. 9—Consistency (in the Aggregate) Over Time

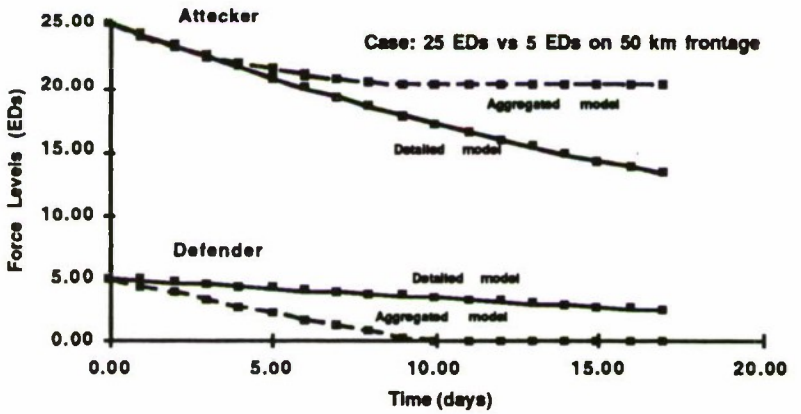


Fig. 10—Failure of Calibration for Other Cases

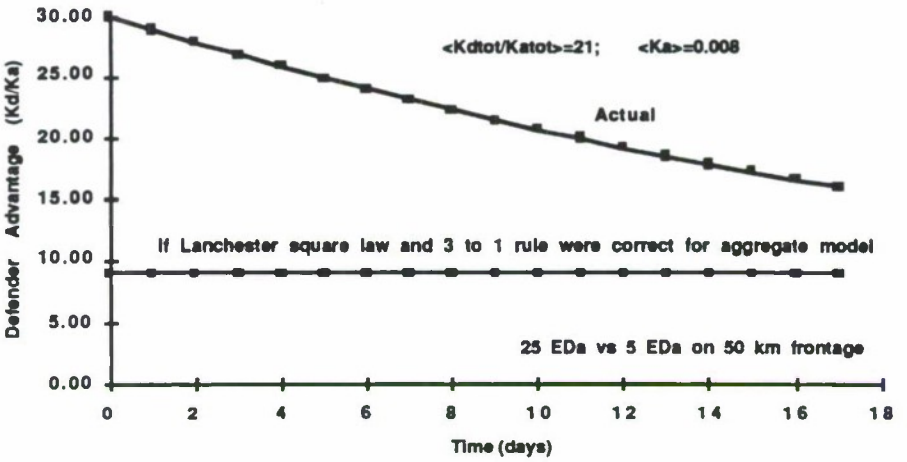


Fig. 11—Badly Nonconstant "Constants"

Diagnosing the Breakdown of Calibration

Why are we getting this behavior? If we go back and compare the original physical pictures (Figs. 6 and 7), we may guess the answer. When dealing with more complex models, however, the reason for the breakdown of calibration might not be obvious, especially to those attempting to treat the models as "black boxes." The reason in this case can be understood from Figs. 12 and 13, which indicate how the fraction of the attacker's and defender's forces on the FLOT vary with force levels.[16] The detailed model includes the concept of "shoulder-space limits," which states that the attacker will try to squeeze as much on line as possible up to the point at which an equivalent division has less than a minimum frontage.[17] Thus, as we increase force levels, the fraction on line goes from 1 to something smaller (Fig. 12). This particular figure comes from making particular assumptions about the shoulder space limit (about 10 km /ED), and on details of the tactical decision rules, but the form is what matters to the concept.

---

[16]For the sake of the example it might have been better to show these fractions as a function of time, because workers often seek to diagnose problems by looking at model behaviors over time rather than understanding the causes of that behavior. Figs. 10 and 11 are more useful, however, in understanding the underlying phenomena.

[17]The concept of a shoulder-space limit is difficult to understand without detailed analysis that accounts for the mutual interference of maneuver vehicles that are too closely spaced and the increased vulnerability of forces that are too concentrated. In World War II divisional frontages were sometimes as small as 1 or 2 km, but in modern warfare doctrine typically calls for frontages of 10 km or more, even for attackers.

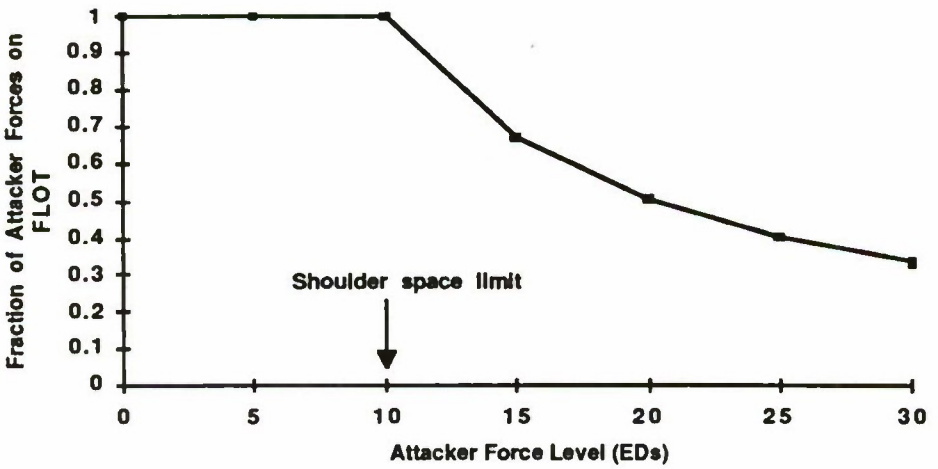


Fig. 12—Attacker May Be Shoulder-Space Constrained

The story for the defender is a bit different (Fig. 13). The detailed model assumes that the defender tries to keep a doctrinal fraction forward (here shown as 2/3, corresponding to the "two up and one back" rule), but never wants the force-to-space ratio to fall below a certain threshold. Thus, if the defender's overall force levels shrink too much, he must put an increasing fraction of his forces on line.

These considerations can lead to a complex time dependence of the defender advantage. In some cases (e.g., 15 EDs vs 3 EDs), the aggregate defender advantage, K_d/K_a, actually oscillates as a function of time (Fig. 14)! First it is high because the defender has a larger fraction of his forces on line; then, as the attacker suffers attrition, he is able to put a larger fraction on line, reducing K_d/K_a. Thereafter, the defender finds himself having to keep a smaller fraction of his forces in reserves, which pushes the ratio up, and so on.

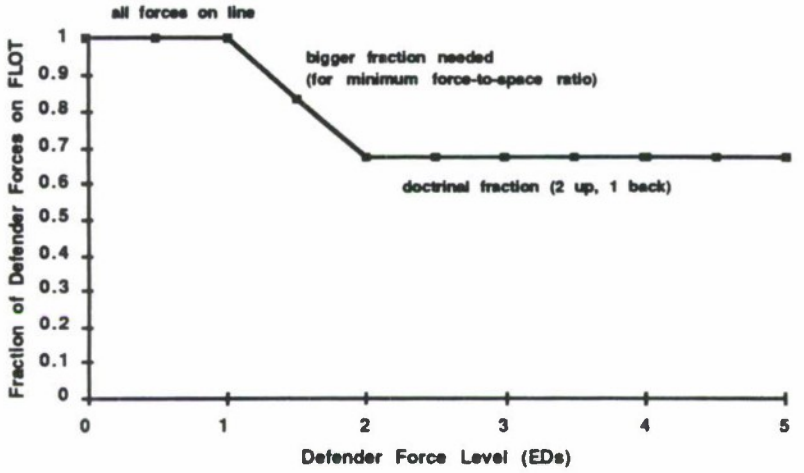


Fig. 13—Defender May Not Be Able To Keep Reserves

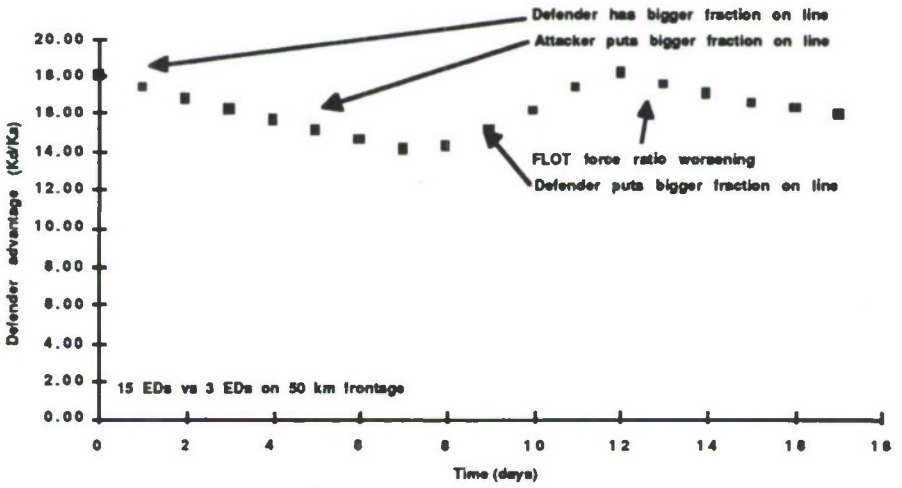


Fig. 14—A Case of an Oscillating Defender Advantage

PATCHING THE AGGREGATE MODEL

Someone who distrusts aggregation generally would stop even experimenting with aggregation at this point (if he even went this far). However, a believer in aggregation, a systems analyst with faith in reductionism, would plunge on. He would say, figuratively,

> Of course. I was stupid. But now I can fix the problem. What's going on is that there is an interaction between frontage and force levels. The aggregation works well only if there is a constant fraction of forces on line over time. But that can't be true if I consider Thermopylae as well as more typical circumstances. In mountains, where the frontage is small, there will be a much smaller number of forces on line. And in open fields, there may be more forces on line (for the attacker) than I assumed previously. So perhaps, then, I need to distinguish three cases: open, mixed, and mountainous terrain; and also three levels of forces: tiny, average, and big. Let's assume that the standard case is mixed terrain and average force levels. We can now develop a slightly more complex model, which we will need to calibrate for the various cases. This, however, should 'do it.' We still don't need to muck around with distinguishing between FLOT and reserves or to worry about the detailed frontage, etc.

Although not shown here, such an approach can be applied with the result that the model works not only for the original "representative case," but also the other cases looked at previously. The believer in aggregation is pleased. Instead of working with constant coefficients K_d and K_a, however, one now has a table-lookup system in which these coefficients vary as a function of the type terrain and qualitative force size.

One trouble, of course, is that the enriched aggregate model still omits many features of the more detailed model and when we look at some of the issues sensitive to those features, the aggregate model fails. For example, if one considers the use of "breakpoints," where a side retires from the field if it loses, e.g., 30% of its strength or is faced with serious flank problems, it is easy to do so in the detailed model, but not in the aggregate model. Further, if one tried to build it into the aggregate model, one would have to go through yet another calibration process, because the break point in the aggregate is not the break point for engaged forces. One could handle this analytically with a bit of effort, but with more general expressions than Lanchester equations, the enrichment would be complex.

There are any number of other detailed issues that could be reflected in aggregate models with proper calibration, but the point is that if we try to address them one-by-one, as "patches" to an originally simple model, the result becomes more and more burdensome and the calibrations more and more difficult to define and track. That, however, is the most common history suffered by aggregate models: they are found wanting, are patched, then patched again, and eventually become tedious and nonsimple.

RECAPITULATION

Let us now recall the generic issues with respect to calibrating aggregate models to detailed models. Our first approach to the aggregate model was to replace the sum over cases by a single, hopefully representative, case. That represented a leap of faith. The second version of the model amounted to saying that the behavior over cases is complex, but

breaks down into a few classes: open, mountainous, and mixed terrain; and tiny, average, and big forces. Within each of those, we could replace the sum by a representative case of that class.

We were also assuming, implicitly, that the appropriate time average was a simple one weighing all times equally over the duration of a representative battle. That would be reasonable enough unless the attrition estimates were embedded in a larger model in which some times are more crucial than others.[18] Even here, however, we should ask "But how long is the representative battle over which the averages should be taken?"

The claim of this example is that we have walked through a relatively common development history for high and low resolution models. We end up with a "hierarchy" of two models, so we have variable resolution, but the relationships between them are sloppy and neither integrated nor seamless. The aggregate model was not really designed to be consistent with the detailed model. Instead, its form was postulated, found wanting, and then patched. In more typical cases, it would be patched again and again. Further, the initial effort to calibrate the models was simplistic and ill conceived. Could we do better? That is the subject of the next section.

---

[18]As an example here, in strategic mobility modeling, it is not unusual to include an explicit time weighting because one wants to make lift decisions that minimize shortfalls in delivered men and equipment, but it is more important to do well initially than later, after substantial forces are in theater.

6. INTEGRATED VARIABLE-RESOLUTION HIERARCHICAL MODELING

Suppose now that we take a different approach to the same problem. If we started over to redesign the models jointly—to develop an integrated family of two models by looking at the internals of the models rather than their black-box behavior—we would quickly find ourselves confused by differences of nomenclature and concept. For example, as one discovers looking at the models in Appendix B, just as one would in looking at two more realistic models that had been developed separately, the models use the same notation for very different things. In particular, A and D mean the total attacker and defender strengths in the first model and the on-FLOT strengths in the detailed model. Similarly, K_a and K_d are obviously not the same in the two models, even though the notation seemed natural to both models when they were built.

These problems of confusing names may seem straightforward to discover and deal with, but there are more subtle ones as well. For example, the aggregate model has a different *concept* than the detailed model of how to represent terrain; it's not merely a matter of aggregation, but also a matter of *perspective* or approach. The simplest version ignored terrain altogether, but the patched-up version has a concept of type terrain, with values of open, mixed, and mountainous. The detailed model also has a concept called type terrain, with roughly the same values (open, mixed, and rough), but it is a very different concept, one applying on a different scale of spatial resolution. The detailed model assumes that combat only occurs on portions of the geographic frontage that are suitable for armored operations. It refers to a "military frontage" L, which is only a fraction of the geographic frontage. The detailed model's notion of terrain type is one that applies for that militarily usable frontage. Thus, one might use the detailed model for analysis in mountains, assume a small militarily usable frontage corresponding to narrow valley roads, but consider the terrain on that frontage to be "open" or "mixed." By contrast, someone using the aggregate model might look at the overall region and assert that the type-terrain is "mountainous." The potential for confusion here is enormous.

This example is not even especially contrived. A few years ago RAND conducted a study concerned with estimating the so-called "operational minimum," the minimum force level with which NATO could reliably defend Western Europe even if Pact forces were at parity with NATO (Despite common views to the contrary, it turned out (Davis, 1990) that no such theoretical minimum exists). In the course of studying why people were coming up with varying claims, we discovered that major organizations throughout NATO used vastly

different methods for characterizing and dealing with terrain—and the discrepancies were generally either unnoticed or ignored, even though they had big effects.

The first step in developing an integrated approach, then, is developing a complete data dictionary with a consistent and intelligible notation. This is not an easy process in general. An important concept here is the notion of a *reference model* that contains all the variables of either of the two original models, plus additional variables needed to clarify relationships and complete the physical picture. These variables, however, must have sensible names that clarify rather than obfuscate. These names will often be different from those of one or both of the original models. In an abbreviated form (i.e., giving only short-form definitions for a subset of the variables), the results might look as shown in Fig. 15 (see also Appendix B).

To avoid confusion the original models might be recoded in terms of the reference-model's variable names (and, also, the names of processes). Sometimes, indeed often, this is considered unnecessary or too expensive, but there is a price paid in comprehensibility, maintainability, and the potential for errors. A compromise, of course, is to build an interface that contains the mappings. Then one can think in terms of the reference-model variables even thought the models continue to operate with their original notation. This may be an adequate approach if indeed the only issues are names. In practice, however, there may also be need to change algorithms, in which case the confusion problem can be severe.

A, D	Attacker and defender force levels
A_{flot}, D_{flot}	Attacker and defender force levels on the FLOT
K_a, K_d	Coefficients in the attrition calculation for FLOT forces
K_{atot}, K_{dtot}	Coefficients in the aggregate attrition model (for total forces)
L_g, L	Geographic and militarily "usable" frontage
DDF_{min}, DDF_{max}	Minimum and maximum defender divisional frontages
ADF_{min}, ADF_{max}	Minimum, maximum, and doctrinal attacker divisional frontages
terr, type battle	Correction-factor parameters adjusting effective strengths to account for terrain and circumstances of battle (e.g., defender's preparations)

A_{break}, D_{break}	Force levels (fraction of original) at which attacker and defender armies break off battle ("break points")

Fig. 15—Defining Variables Consistently in a Reference Model

Given the reference model, I recommend as a next step literally drawing pictures showing functional relationships. Fig. 16 shows such a picture for the original models (simplified to ignore flank and break-point issues, but with the "patches" included for the aggregate model). We can think of these figures as being the skeletons of data-flow diagrams in which the variable names are place holders for bubbles such as "Compute DLR, ALR, and RLR". We see in this depiction of the original models that the variables were confusing, nonhierarchical, and different in perspective. For example, the detailed model highlighted the concept of the fraction of forces on line (see the variables A_{flot} and D_{flot}). That concept obviously doesn't appear in the aggregate model. However, the aggregate model has a concept the detailed model did not: it calculates "effective" force strengths (denoted by primes), which are obtained by multiplying nominal force strengths by correction factors for terrain and force size. By contrast, the detailed model treated terrain in a different way and never used "effective force size" as a concept. Which is right?

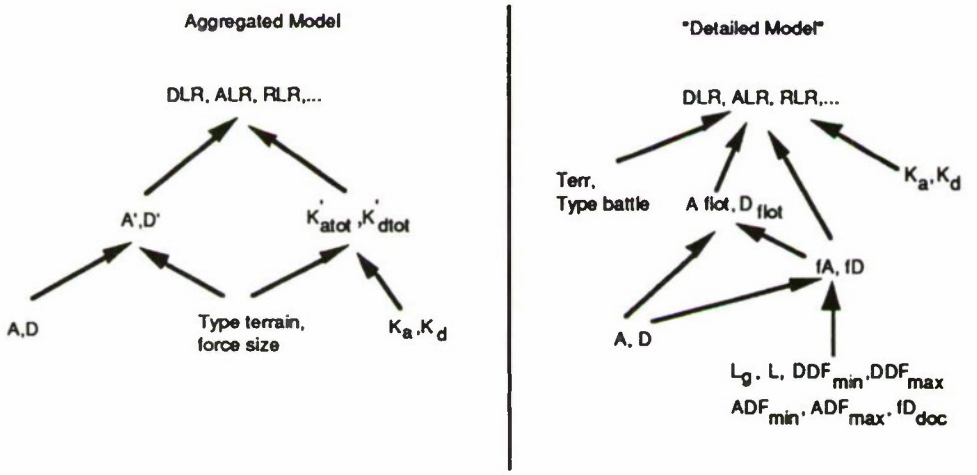


Fig. 16—Original Relationships With Renamed Variables

If one examines the models with this confusion in mind (and here one needs to be looking directly at the computer code unless the model is fully specified outside the program), it becomes clear that one can unify the descriptions as suggested schematically in Fig. 17:

some of the differences between the two models (e.g., the ones mentioned above) were arbitrary and can be eliminated.

Note that we now have hierarchical structures. Further, most of the concepts of the high- and low-resolution models are the same. And, finally, the flow for calibrations is explicitly indicated. In most cases the algorithms for those calibrations are straightforward because the concepts *do* correspond. Where the concepts are not the same (as, here, in the treatment of terrain), one must either make them consistent by changing one model rather fundamentally or develop the appropriate calibrating algorithms. Thus, if we wish to retain the concepts of type terrain and force size in the aggregate model, we need to specify how one can infer their values from the variables of the detailed model. The algorithm for doing so may or may not be satisfying; it will usually be heuristic.

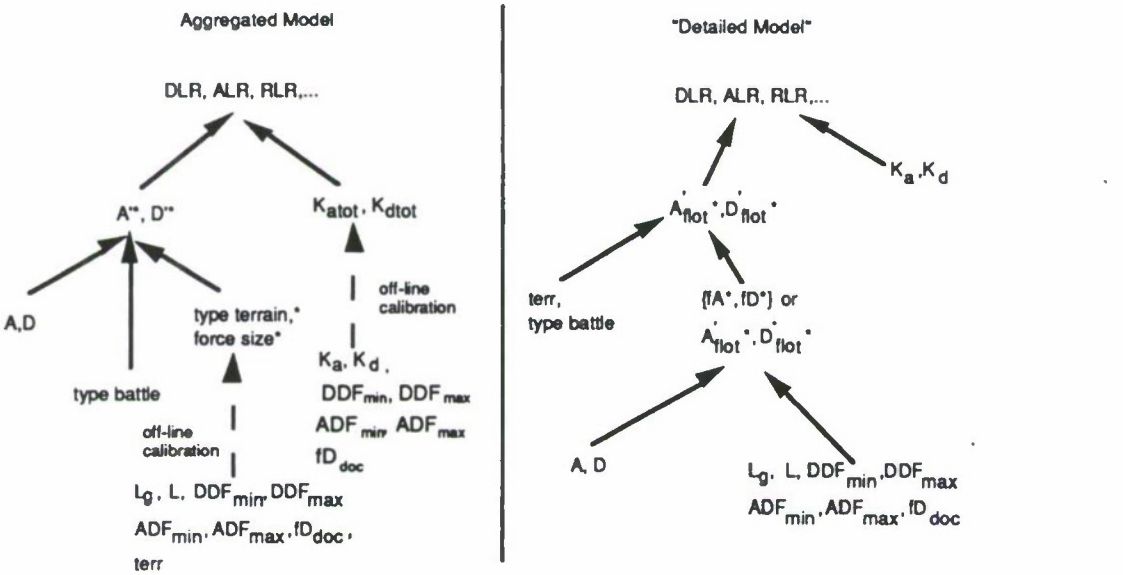


Fig. 17—A Revised and Hierarchical Design With Alternative Resolution and Perspectives

Looking at the detailed model on the right, we see a variable-resolution hierarchical design with several natural levels. Again, asterisks indicate levels at which a user can either provide inputs directly (by specifying the asterisked variables as parameters) or generate the information by executing more detailed functions dependent on the variables shown by arrows coming up from below. Assuming one programmed in the appropriate switches, one could choose to run the detailed model at maximum resolution, or at a level at which one specifies the forces on the FLOT directly, which might be useful. Alternatively, one could use the aggregate model at two levels of resolution (i.e., in addition to having a variable-

resolution model, on the right, one could have an alternative aggregate model with a slightly different perspective than the lowest-resolution version of the variable-resolution model). The aggregate diagram also shows that parameter values of type-terrain and force-size could be calibrated off line in terms of the detailed model's parameters. This, of course, would require appropriate averaging over cases, as discussed earlier.

The family of models indicated schematically here are the same substantively as before, but they are now integrated: relationships are explicit, notation is consistent, and it is clearer how one set of concepts and variables flows into another. Some of this is notation, some of it is design choice, and some of it is the picture itself. *The claim here is that this revised design is more "seamless" because of the integration.* For example, suppose one had been using the detailed model and decided to switch to the aggregate model. A mere glance at Fig. 17 (and Fig. 15) would indicate: (a) in the new model one would be thinking in terms of modified coefficients K_{atot} and K_{dtot}, which can be calibrated to parameters of the detailed model but are not identical with K_a and K_d; (b) in the new model one would think exclusively in terms of total forces, not FLOT forces, but the effective strength of the total forces would indirectly account for the variability of force fraction on the FLOT by applying corrections called type terrain and force size, which can be calibrated to concepts of the detailed model. This seems conceptually straightforward and does not require any mental lurches.

Although this approach to model design is uncommon, it often works and works well (e.g., Davis, 1987). Fig. 18 depicts a model recently developed in Germany by Reiner Huber (see Davis and Huber, 1992). The top level of this model describes an overall effective "force ratio" for the theater. That, in turn, is computed by combining measures of ground-force potential and air-support potential, which in turn depend on more detailed variables, down to the level of sortie rates, kill probabilities, and terrain characteristics. The motivation for this model (called GEFRAM) was the need for policy makers and general officers to be able to understand certain military balances without going through complex simulations. The important point in this paper is that the model's architecture can be understood at a glance and there is true variable resolution capability from physics-level relationships such as range-payload data up to a corps-level force-ratio calculation—all within a single model.

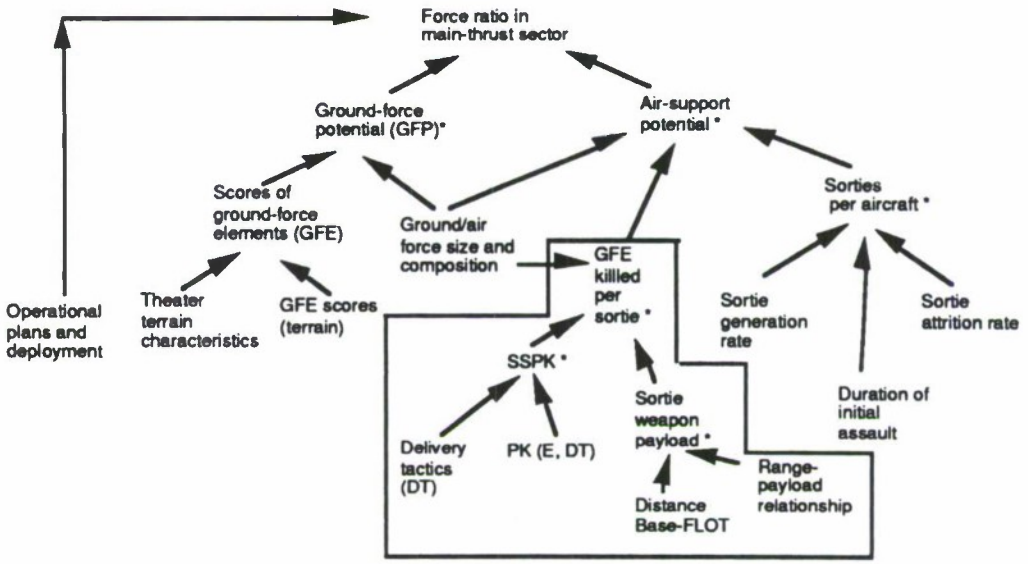


Fig. 18—Another Example of IHVR Design

7. DISCUSSION AND RECOMMENDATIONS

GENERIC CHALLENGES

Let us now consider some of the more generic challenges of variable-resolution modeling and what kinds of approaches make sense. Fig. 19 summarizes such challenges, using a terminology associated with pairs of models, even though in practice we often want multiple levels of resolution. As shown, it is essential to get the concepts and variable names straight, and to generate a complete set of variables and functions so that one has a representation of the "entire" system (this is the "reference representation discussed earlier). It is very useful to draw the relationships and mappings. The next item in the list is to decide on the *form* of reasonable aggregate equations. The last section assumed that the aggregate equations had the same form as the detailed equations, but that is usually not the case. Further, it is usually not a good idea to assume that the aggregate expressions will be simple and intuitive.[19] One needs to add a dose of *theory* to inform one's hypotheses. That said, experience suggests that intuition and first-order theory are often inadequate, especially when (as in many simulations) problems do not lend themselves to closed-form analysis. Experimentation with higher-resolution models is often necessary in order to develop a good sense of proper aggregations. Even so, a moderate amount of equation shuffling can pay big dividends. The last item in the list of Fig. 19 addresses the problem of defining appropriate averages over relevant cases, averages defined with appropriate weighting factors, which unfortunately will depend, often sensitively, on the analytic, educational, or operational context.

- Getting the concepts and names straight
- Completing sets of variables and functions (i.e., defining the reference model)
- Drawing relationships and mappings
- Deciding form of reasonable aggregate equations relative to detailed equations (requires theoretical analysis)

[19]As discussed in Horrigan (1991) standard aggregated models of combat often assume independent events and ignore the role of spatial relationships. The result (configurational errors) can be errors of a factor or 2 or more in the assessed relative goodness of alternative weapon systems. The example in the last section is actually a special case of a configuration problem in that the relationship between FLOT forces and reserves is critical, but cannot in general be accounted for by a constant scaling factor. Other more complex examples are given in Hillestad, Owen, and Blumenthal (1992).

- Finding conditions under which aggregation equations might be reasonably valid (requires theoretical analysis)
- Expressing aggregate-model parameters in terms of outputs of detailed model (requires theoretical analysis)
- Deciding on cases (e.g., scenarios) to be distinguished and how to make calibrations for each case—e.g., how to determine weighting factors over case and time so that calibration will be appropriate for context of larger application (requires theoretical analysis).

Fig. 19—Generic Challenges in Variable Resolution Modeling

ON THE GENERALITY OF INTEGRATED HIERARCHICAL METHODS

In this paper I have emphasized integrated hierarchical variable resolution (IHVR), primarily because it has such a high payoff when it works and because it is not well understood in the community. A few observations are appropriate, however. First, note that the hierarchies treated here involve *processes* (e.g., "Compute attrition"), not objects or entities. This is significant because the powerful methods of object-oriented modeling and programming are primarily focused on objects, not processes. While they make hierarchical modeling straightforward, the hierarchies are in another dimension of the problem (e.g., army groups break into armies or corps, which break into divisions, which break into brigades, and so on). While hierarchical representation of objects is rather widely valid and natural in combat modeling, *straightforward* hierarchical modeling of processes is only sometimes feasible. More generally, the relevant processes have a more complex relationship to each other, with connections across branches of the hierarchical tree and, in some cases, with iterations or cycles of data flow. It follows that to exploit IHVR methods it will be necessary to develop appropriate approximations that break these cross-branch interactions and cycles and compensate, for example, by adjusting coefficient values from time to time in the simulation as gross features of the situation change. This will require theoretical analysis explicitly separating the different spatial and temporal scales that are natural to the problem (e.g., road marches often occur over many hours, while close combat at the brigade level may be completed in tens of minutes).

There are a variety of other complications (see Davis and Huber, 1992), including the tendency to underspecify models, leaving programmers to fill in details, which they often do in ways that limit flexibility later. Another complication is that the approach this paper recommends is one that places a premium on design, at a time when fashion calls for rapid prototyping, which is often viewed as the antithesis of emphasizing design. My own view is that such extreme versions of rapid prototyping are licenses to steal. Further, experience suggests that a moderate amount of initial design goes a very long way, whereas failure to

have any formal design initially reduces substantially the likelihood of achieving well integrated and seamless results later.

Fig. 20 summarizes recommendations on this matter. In this approach, work begins with an initial design—one taking a matter of days or weeks, however, not months or years. The focus is on the big picture, which translates into defining the decompositional issues well, anticipating variable-resolution needs, building in stubs and sketching out the various trees (as in Fig. 19), and getting all the names straight. In doing this one must make choices, because a hierarchy that is natural in one application domain may or may not be natural in another. For example, the GEFRAM model described above develops a measure of overall force ratio that combines effects of air and ground forces. That has proven quite useful for some applications. By contrast, it might seem quite unnatural in a war gaming simulation used for education, operational planning, or analysis informed by operational considerations.

Having made a first set of choices and designs, the next step is indeed rapid prototyping, focusing on inputs and outputs and obtaining enough insights to permit iteration of the design based on initial simulation results. After some iteration, the structure of the model may well settle down, at which point one can work out details of algorithms and relationships, including the appropriate aggregation relationships and calibration methods. Iteration should continue, but there should be a major effort not to undercut the overall architecture or one will quickly generate numerous seams.

- Develop initial design, focusing on composition and top-down views.
- Anticipate need for variable resolution. Build in "stubs." Draw "trees." Choose names to clarify hierarchical relationships.
- Make choices of perspective to determine "best" hierarchical structures. Create explicit hierarchy-breaking approximations.
- Use rapid prototyping. Use first-order algorithms. Focus on inputs and outputs. Use theory to tighten calibration relationships.
- Experiment and iterate design.
- Complete top-level design and proceed. Use more serious algorithms.
- Do not lightly assume "simple" aggregation relationships. Derive from theory when possible. Experiment. Iterate.
- Adapt with applications, but don't undercut basic design.

Fig. 20—Recommended Approach to Design

Is this approach a panacea? By no means. It has proven useful, however, in several quite different applications. Further, it seems to have a fair degree of generality that has not

yet been exploited—in part because the value of doing so has not previously been emphasized and in part because, as noted above, the method will require theoretical efforts to develop good approximations that separate phenomena occurring on different scales, and that recognize that aggregation relationships are often complex and nonintuitive. There is a great potential for developing powerful and relatively seamless variable-resolution combat models,[20] but a great deal of work has yet to be done if we are to achieve that potential.

---

[20]An alternative view is that the hierarchy approach is doomed to failure because detailed processes so commonly intrude on higher level views of the problem. Examples of the interference of detailed processes at high levels are very familiar to general officers and even civilians experienced in war gaming. Simulated wars literally go one direction or another as a function of details such as the arrival time of critical reserves in a particular sector, or the range advantage enjoyed by particular weapons. The solution, I believe, is in emphasizing highly interactive models that permit users rapidly and flexibly to "turn the knobs and switches" necessary to include or not include detailed processes in a given application, or even in a given portion of a given simulation run.

Appendix A
BACKGROUND

Variable-resolution issues have been discussed among combat modelers for many years, off and on, but the early work did not leave much of a theoretical legacy. I became interested in the mid 1980s when directing early development of the RAND Strategy Assessment System (RSAS), an analytic war gaming system used for global- and theater-level gaming and analysis. We had multiple objectives and associated tensions regarding the "right" level of resolution. Could we have our cake and eat it? I thought so and encouraged designs that would provide alternative levels of resolution. It was an uphill battle to make this happen, however, and recognized techniques for such designs did not seem to exist. Other organizations encountered similar difficulties. Nonetheless, in the course of the work some techniques emerged that seemed to have general value.

The issue reemerged two years ago in a paper (Davis and Blumental, 1991) that elaborated the irony in combat modeling that workers at *all* levels tend to believe that their level of resolution is "correct," that those working at lower resolution are either naive or sloppy, and that those working at higher resolution are lost in the weeds. By contrast, physical scientists learn early to appreciate both thermodynamics and quantum statistical mechanics. Can we develop similarly enlightened views for combat modeling and analysis? DARPA agreed to sponsor work on the issue and the same problems arose as the Defense Modeling and Simulation Office began worrying about such interoperability issues as how to combine models of different resolutions and have the results be meaningful (DMSO, 1992). Further, the DoD's vision of a seamless continuum between models and reality provided even more motivation for taking the subject seriously. All of this led to a think piece (Davis and Huber, 1992), a small preliminary workshop at RAND late in 1991, and a first interdisciplinary conference on the subject in May of 1992, at which an earlier version of this report served as an introduction.

Although I make no attempt in this study to survey the academic literature, there is considerable relevant material, much of it dealing with what is called model abstraction. Some useful references here include Innis and Rexstad (1983), Zeigler (1984), Courtois (1985), Fishwick (1988), and Sevinc (1991). Also, anyone considering modeling of complex systems should see Simon (1981), which discusses the prevalence of "nearly decomposable hierarchies" in real-world systems, including living organisms.

Appendix B
DEFINITION OF THE MODELS

The models used in this report are highly simplified representations of ground combat within a corps sector. They are by no means supposed to be realistic. However, for completeness, I define the models in the following paragraphs, specifying their variables, processes, calibrations, and implementation as programs. Importantly, I describe them as I first developed them, before attempting to integrate, because I wanted to be able to illustrate the difficulties of after-the-fact integration.

THE AGGREGATE MODEL

Table B.1 lists and defines the variables of the aggregate model. The minimal inputs are initial values of attacker and defender strengths, A_0 and D_0, and values of the attrition coefficients K_a and K_d. A and D constitute a complete set of dynamic variables, but it is convenient to compute additional variables that are determined by their definitions and the values of A and D. These are the variables F, ALR, DLR, and RLR.

Table B.1

Variables of the Initial Aggregate Model

Variable	Meaning
A	Attacker strength (equivalent divisions, EDs)
D	Defender strength (EDs)
F	Attacker to defender force ratio: $F = A / D$
ALR	Attacker loss rate (fractional loss per unit time): $ALR = \frac{dA/dt}{A}$
DLR	Defender loss rate: $DLR = \frac{dD/dt}{D}$
RLR	Ratio of (relative) loss rates: $RLR = \frac{ALR}{DLR}$
Parameter	**Meaning**
K_d	Attrition coefficient: kills per day of attacker EDs per ED of defender
K_a	Attrition coefficient: kills per day of defender EDs per ED of attacker

The only process in this trivial model is that of attrition, governed by a Lanchester square law:

(1) $$\frac{dA}{dt} = -K_d D; \quad \frac{dD}{dt} = -K_a A .$$

It follows from Eq. 1 that

(2) $$ALR = -\frac{K_d}{F}; \quad DLR = -K_a F; \quad RLR = \frac{K_d / K_a}{F^2} \quad .$$

Since there are two independent dynamic variables, two calibration conditions are necessary. The first and most important assumption is the 3 to 1 rule, which says that RLR=1 when F=3. That is, the breakeven point is at a force ratio of 3 to 1. A battle that begins with such a force ratio will result in the antagonists destroying each other, with neither side ever improving the force ratio. The assumption is reasonable only if the defender has substantial advantages by virtue of, e.g., prepared defenses and a favorable terrain.

This expression of the 3 to 1 rule implies (see Eq. 2) that the "defender advantage" obeys

$K_d/K_a = 9.$

The other calibration assumption determines the scale of attrition rather than the relative attrition rates. The calculations in the text assume K_d=0.18, which means that at a force ratio of 3 the attacker would be losing 6% of its strength per unit time (taken to be a day).

The model was implemented as an EXCEL® spreadsheet simulation program. The differential equations were represented crudely by first-order difference equations with one-day time steps.

The patched version of the aggregate model, referred to only briefly in the text, has two additional parameters, type terrain and force size. A process is also added to adjust effective force strengths and/or effective attrition coefficients to reflect type battle.

THE "DETAILED" MODEL

Table B.2 lists and defines the variables of the "detailed" model. The minimum set of dynamic variables is (A_{tot}, D_{tot}), where A_{tot} and D_{tot} are the total attacker and defender strengths. The problem is defined by their initial values and the values of the parameters shown below them in the table. The attacker and defender strengths on the FLOT, A and D, as well as the variables F, ALR, DLR, and RLR can be calculated for convenience or because they represent important intermediate variables of the attrition process. Note that RLR here is the ratio of relative loss rates for FLOT forces, not for total forces.

Table B.2

Variables of the Detailed Model

Variable	Meaning
A_{tot}	Total attacker strength (EDs) (FLOT forces plus reserves)
D_{tot}	Total defender strength (EDs) (FLOT forces plus reserves)
A	Attacker strength (EDs) on the forward line of troops (FLOT)
D	Defender strength (EDs) on the forward line of troops
F	Attacker to defender force ratio: $F = A / D$
ALR	Attacker loss rate (fractional loss per unit time): $ALR = \frac{dA/dt}{A}$
DLR	Defender loss rate: $DLR = \frac{dD/dt}{D}$
RLR	Ratio of (relative) loss rates on the FLOT: $RLR = \frac{ALR}{DLR}$
Parameter	**Meaning**
ADF_{min}	Minimum frontage (km) per attacker ED (shoulder-space limit)
ADF_{max}	Maximum frontage (km) per attacker ED
DDF_{min}	Minimum frontage (km) per defender ED
DDF_{max}	Maximum frontage (km) per defender ED
fD_{doc}	Doctrinally preferred fraction of defender forces on the FLOT
type terrain	Type of terrain on which battle is fought (e.g., open, mixed, or rough)
type battle	Type of battle (e.g., assault on prepared defenses, hasty defenses,...)
L	Militarily usable frontage (km)
A_{break}, D_{break}	Per cent of original force levels at which attacker and defender armies break off battle ("break points")

This model has three processes, one each for attrition, the decision to move reserves to the FLOT and the decision to quit the battle if losses are excessive. The principal process assumed was that of Lanchester-square attrition for those forces on the FLOT. In its simplest form:

$$(3) \qquad \frac{dA}{dt} = -K_d D; \quad \frac{dD}{dt} = -K_a A .$$

where A and D are strengths on the FLOT. This does not include the changes in A and D due to reserves moving forward.[21] The model assumes that forces on the FLOT remain there, but forces in reserve can be sent to the FLOT. The decision rules for doing so are as follows. The attacker places as much of his strength on the FLOT as is permitted by the

[21]In non-standard terrains or types of battle, Eq. 3 is modified with numerical multipliers on the right side. For example, in open terrain, the defender would have a smaller advantage than in "normal" terrain, so the multiplier would be less than 1. None of these corrections were used in this report, but their existence is indicated by the parameters type terrain and type battle. In Fig. 16, the parameter "terr" is the same as type-terrain in the original detailed model.

constraint ADF_{min}, which represents a "shoulder-space constraint" (e.g., at least 10 km of frontage per equivalent division). The defender has more complex rules:

- Never put more forces on line than are permitted by the shoulder-space constraint DDF_{min} (seldom a consideration).
- So long as the frontage per ED on the FLOT is no worse than DDF_{max}, maintain a doctrinal fraction fD_{doc} of forces forward .
- If there are too few forces for this, put a larger fraction of the forces forward as necessary to maintain, if possible, frontages no worse than DDF_{max}.
- When this is no longer possible, put all forces on line.

The last process involves breaking off the battle when losses are excessive. The battle ends when either or both of the sides have fractional losses in excess of the relevant "break-point" thresholds (A_{break} and D_{break}). Most calculations shown in the test assume there is no breakpoint.

The calculations shown in this report assume baseline parameter values as follows:

K_d=0.27 (3/2 the value of the corresponding parameter of the aggregate model, since, nominally, only 2/3 of the forces are on line)

ADF_{min}=DDF_{min}=10 km/ED

DDF_{max}=40 km/ED

fD_{doc}=0.667 (i.e., two up and one back).

Again, these models and calibrations are merely illustrative and should not be taken too seriously. This study is about modeling theoryk, not about combat phenomena.

As with the simple model, implementation involved a simple spreadsheet simulation with one-day time intervals. At the start of each day the new force level on the front was set to the previous day's initial value minus that day's attrition plus whatever reserves would be sent forward consistent with the decision rules discussed above. No effort was made to investigate the effects of variable time step, higher precision, etc.

THE REFERENCE MODEL

As discussed in the text, if one wants to integrate existing models it is important to develop a comprehensible and complete set of variable names. Table B.3 suggests the variables of a reference model that could embrace all the content of the aggregate and detailed models described above. One reads it from the center column. Thus, from the first row, the variable A of the reference model corresponds to A of the aggregate model and A_{tot} of the detailed model. Brackets indicate intermediate variables that were implicit in the

models, but important conceptually. E.g., the strength of attacker reserves, A_{res}, may have been represented simply as A_{tot}-A, but deserves to be given its own name)

Table B.3

Mapping of Variable Names Among Aggregate, Reference, and Detailed Models

Variables of Aggregate Model	Variables of Reference Model	Variables of Detailed Model
A	A	A_{tot}
D	D	D_{tot}
[A']		
[D']		
	A_{flot}	A
	D_{flot}	D
	A_{flot}'	[A']
	D_{flot}'	[D']
	A_{res}	[A_{res}]
	D_{res}	[D_{res}]
F	F_{tot}	
ALR	ALR_{tot}	
DLR	DLR_{tot}	
RLR	RLR_{tot}	
	ALR	ALR
	DLR	DLR
	RLR	RLR
K_d	K_{dtot}	
K_a	K_{atot}	
	K_d	K_d
	K_a	K_a
type terrain	Type-terrain	
force size	Force-size	
(K_a')		
(K_d')		
	ADF_{min}, ADF_{max}, DDF_{min}, DDF_{max}	ADF_{min}, ADF_{max}, DDF_{min}, DDF_{max}
	fD_{doc}	fD_{doc}
	fA_{doc}	
	L	L
	Lg	
	A_{break}, D_{break}	A_{break}, D_{break}
	terr	type terrain
	type battle	type battle

BIBLIOGRAPHY

Allen, Patrick (1992), *Situational Force Scoring: Accounting for Combined Arms Effects in Aggregate Combat Models*, RAND, N-3423-NA.

Bennett, Bruce, Carl Jones, Arthur Bullock, and Paul Davis (1988), *Main-Theater Warfare Modeling in the RAND Strategy Assessment System*, RAND, N-2743-NA.

Courtois, P.J., "On Time and Space Decomposition of Complex Structures," *Communications of the ACM*, **28**, No. 6, June 1985, pp. 590-603.

Davis, Paul K. (1987), "A New Analytic Technique for the Study of Deterrence, Escalation Control, and War Termination," in Stephen J. Cimbala, *Artificial Intelligence and National Security*, Lexington Books, D.C. Heath and Company, Lexington, Massachusetts.

Davis, Paul K. and Donald Blumenthal (1991), *The Base of Sand Problem: A White Paper on the State of Military Combat Modeling*, RAND, N-3148-OSD/DARPA.

Davis, Paul K. (1990), "Central Region Stability at Low Force Levels", in Reiner Huber (editor), *Military Stability: Prerequisites and Analysis Requirements for Conventional Stability in Europe*, Nomos Verlagsgesellschaft, Baden-Baden.

Davis, Paul K. (1986), Steven C. Bankes, and James P. Kahan, *A New Methodology for Modeling National Command Level Decisionmaking in War Games and Simulations*, RAND, R-3290-NA.

Davis, Paul K. and John Arquilla (1991), *Deterring or Coercing Opponents in Crisis: Lessons from the War With Saddam Hussein*, RAND R-4111-JS.

Davis, Paul K. and Reiner Huber (1992), *Variable Resolution Combat Modeling: Motivations, Issues, and Principles*, RAND, N-3400-DARPA.

Defense Modeling and Simulation Office (1992), *Defense Modeling and Simulation Initiative*, Office of the Director, Defense Research and Engineering, Department of Defense.

Dupuy, Trevor N. (1987), *Understanding War*, Paragon House Publishers, New York.

Epstein, Joshua (1990), *Conventional Force Reductions: A Dynamic Analysis*, The Brookings Institution, Washington, D.C.

Hillestad, Richard and Mario Juncosa (1992), *Cutting Some Trees to See the Forest: On Aggregation and Disaggregation in Combat Models*, RAND, R-4250-DARPA.

Hillestad, Richard, John Owen, and Donald Blumenthal (1992), *Experiments in Variable Resolution Combat Modeling*, RAND, N-3631-DARPA.

Horrigan, Timothy (1991), *Configural Theory and the Mathematical Modeling of Combat: Realizing the Potential of Models and Simulations of Combat to Improve the Effectiveness of Weapons, Tactics, and Training*, Horrigan Analytics, HAS 91-17-1, Chicago, Illinois.

Innis, G. and E. Rexstad (1983), "Simulation Model Simplification Techniques," *Simulation*, July 1983, pp. 7-15.

Sevinc, Suleyman (1990), "Automation of Simplification in Discrete Event Modelling and Simulation," in *Int. J. General Systems*, Vol. 18, pp. 125-145.

Simon, Herbert (1981), *Sciences of the Artificial,* 2nd Edition, MIT Press, Cambridge, Mass.

Zeigler, Bernard (1984), Multifacetted Modelling and Discrete Event Simulation, Academic Press, London.

Object Oriented Modeling: Holistic and Variable-Resolution Views

Frederick Eddy
General Electric
Corporate Research and Development
Schenectady NY USA
E-mail: eddy@crd.ge.com
Phone: 518-387-7163

Presented at the
Conference on Variable Resolution Modeling
Washington, D.C., May 5-6,1992

Some of the material in this talk is taken from the book *Object-Oriented Modeling and Design*, by James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen, ©1991 by Prentice-Hall Inc. Reproduced by permission.

©1992 by Frederick Eddy
All rights reserved

Outline

- Introduction to Object-Oriented Modeling
- Object-Oriented Modeling Concepts and OMT Notation
  - The Object Model
  - The Dynamic Model
  - The Functional Model
- An Example
- Variable Resolution in OMT Models

Introduction to Object-Oriented Modeling

- An object-oriented model is based on real-world experience
- Terms used are meaningful in the application domain
- Object models are easily extended in scope by adding new objects and relationships
- Object models may be refined at successively finer levels of resolution by using aggregation, specialization, substates, and subprocesses

Object Modeling Technique (OMT):

- OMT is both a methodology and a notation for object-oriented modeling
- The OMT notation is graphical, and language-independent
- The OMT methodology is a well-defined process
- OMT supports domain modeling, analysis, system and software design, and database design

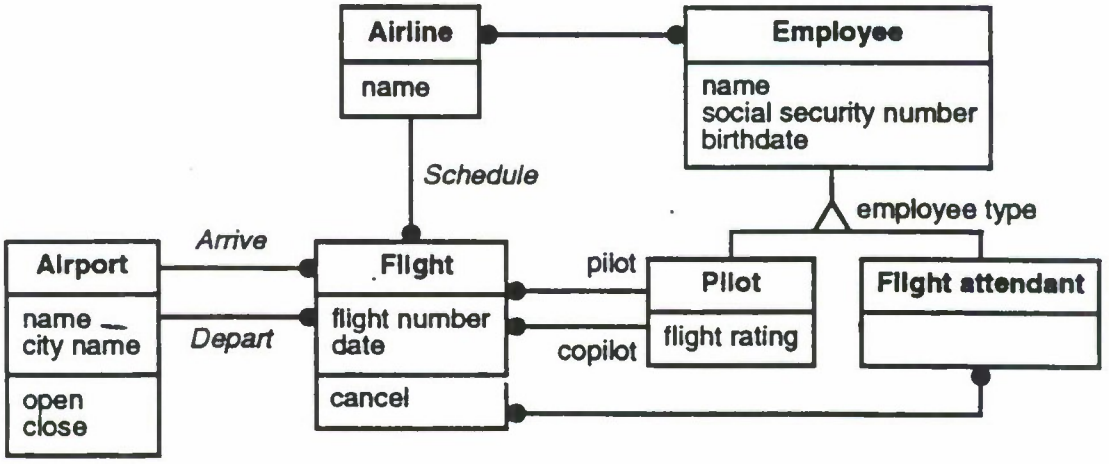
The OMT Methodology Synthesizes Past Work

- System Modeling:
  - Captures three aspects: Structure, Dynamic response, and Functional transformations.
  - Extends existing notations and applies them in a new way.
- Information modeling:
  - Independent entities that are loosely associated.
  - Database technology.
- Object-oriented programming:
  - Objects as coherent entities with identity, state, and behavior.
  - Inheritance and polymorphism.
- Software engineering:
  - The notion of a formal process for software development.
  - Emphasis on the early conceptual stages of software development, rather than on programming and implementation details.

Three Views of OMT: The Object Model

- ***Object Model — Things***
  - ***Static structural relationships of objects.***
  - ***What are the kinds of objects and their relationships?***
- Dynamic Model — Interactions
  - Temporal interactions of objects.
  - What are the stimuli to objects and their responses?
- Functional Model — Transformations
  - Functional dependencies of values.
  - What are the computations that objects perform?

Sample OMT Object Model

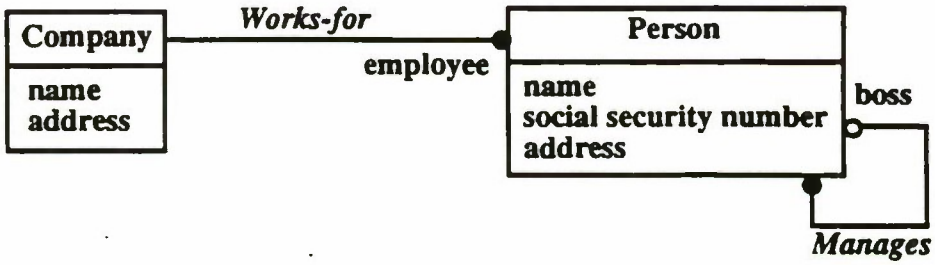


Object and Class

- ***Object***: a concept, abstraction, or thing with crisp boundaries and meaning for the problem at hand. An object is an instance of a class.
- ***Class***: a description of a group of objects with similar properties (attributes), common behavior (operations), common relationships to other objects, and common semantics.
- ***Attribute***: a named property of a class describing data values held by each object of the class.
- ***Operation***: a function or transformation that may be applied to objects in a class.
- ***Method***: the implementation of an operation for a particular class.

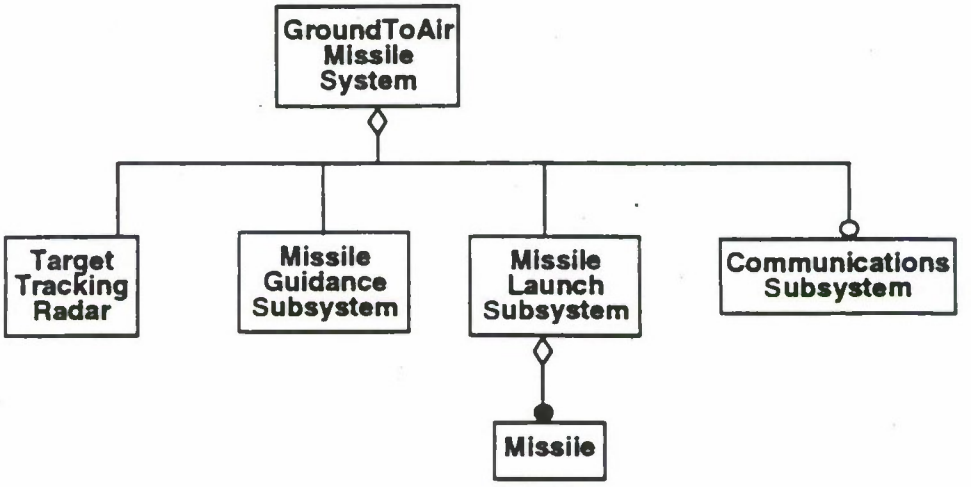
Link and Association

- ***Link***: a physical or conceptual connection between objects. A link is an instance of an association.
- ***Association***: a relationship among instances of two or more classes describing a group of links, with common structure and common semantics.
- ***Role***: one end of an association.
- ***Multiplicity***: the number of instances of one class for each single instance of the other class.



Aggregation: a Special Form of Association

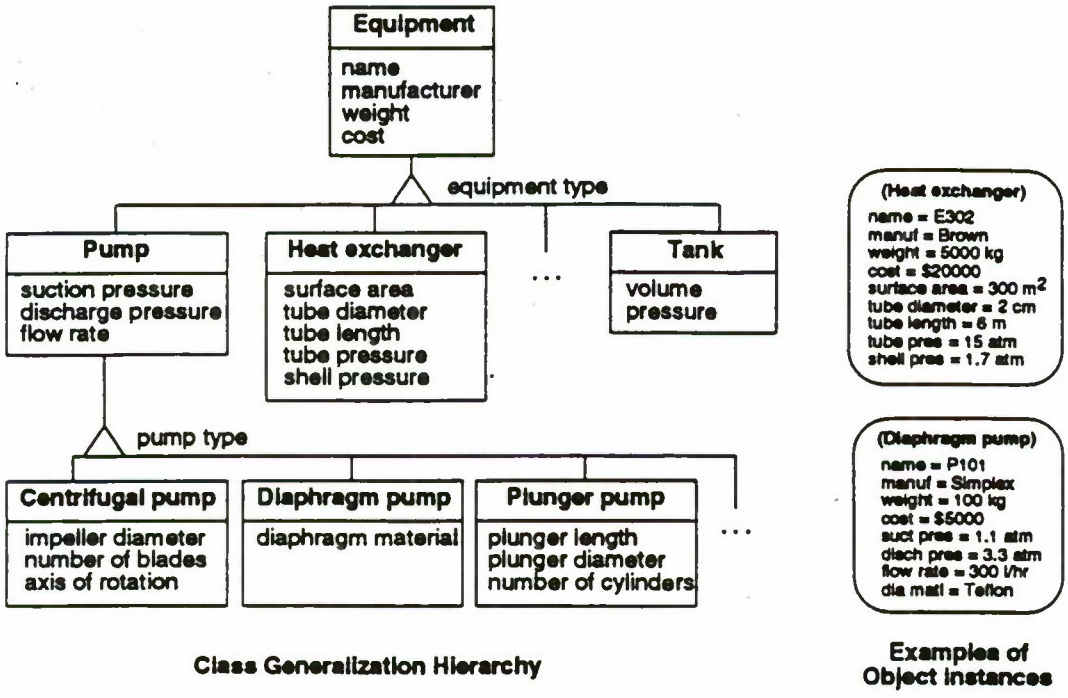
- ***Aggregation***: **a special form of association, between a whole and its parts, in which the whole is composed of the parts.**
- **Example: Parts explosion tree**



Generalization

- ***Generalization***: the relationship that organizes classes based on their similarities and differences.
  - The ***superclass*** holds common attributes and operations.
  - Each ***subclass*** adds its own unique attributes and operations.
  - A subclass ***inherits*** the attributes, operations, and associations of its superclass.
  - A subclass can ***override*** the implementation of an operation—the method.
- The ***Is-a*** relationship: Each instance of a subclass **is** simultaneously an instance (transitively) of all its superclasses.
- ***Multiple inheritance***: a type of inheritance that permits a class to have multiple superclasses and to inherit attributes and operations from all parents. This permits mixing of information from two or more sources.

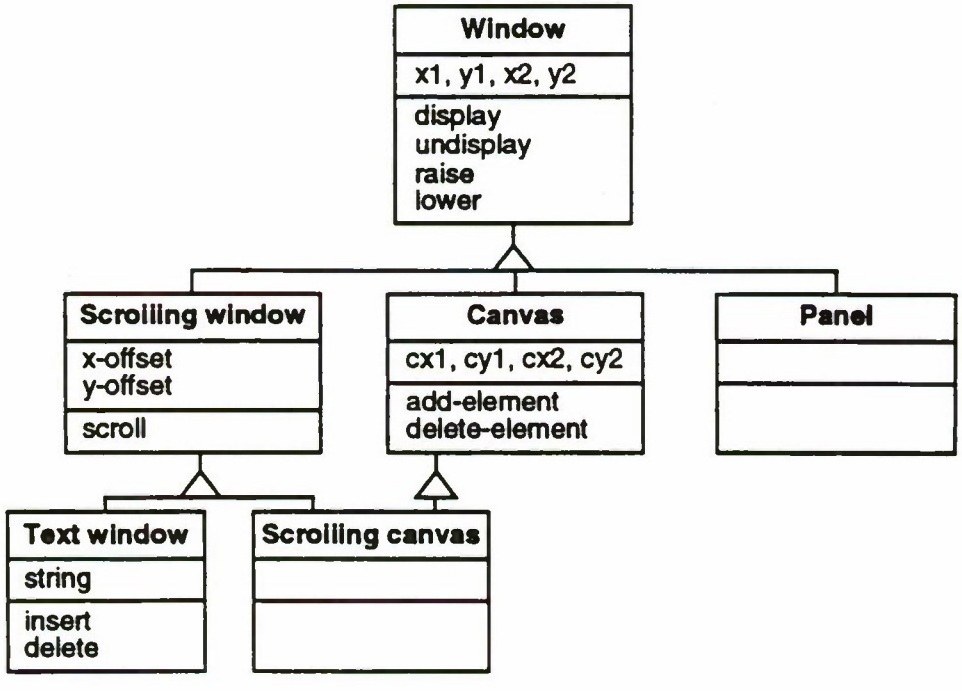
Generalization



Class Generalization Hierarchy

Examples of Object Instances

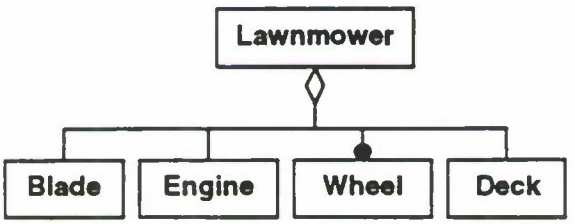
Generalization



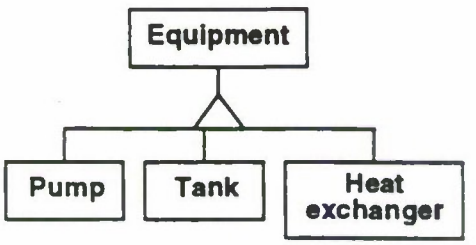
Aggregation vs. Generalization

- Aggregation and generation both form trees.
- Aggregation is the ***Has-A*** relationship; Generalization means ***Is-A***.
- Aggregation relates distinct objects composing a composite object.
- Generalization relates classes describing different aspects of a single object.

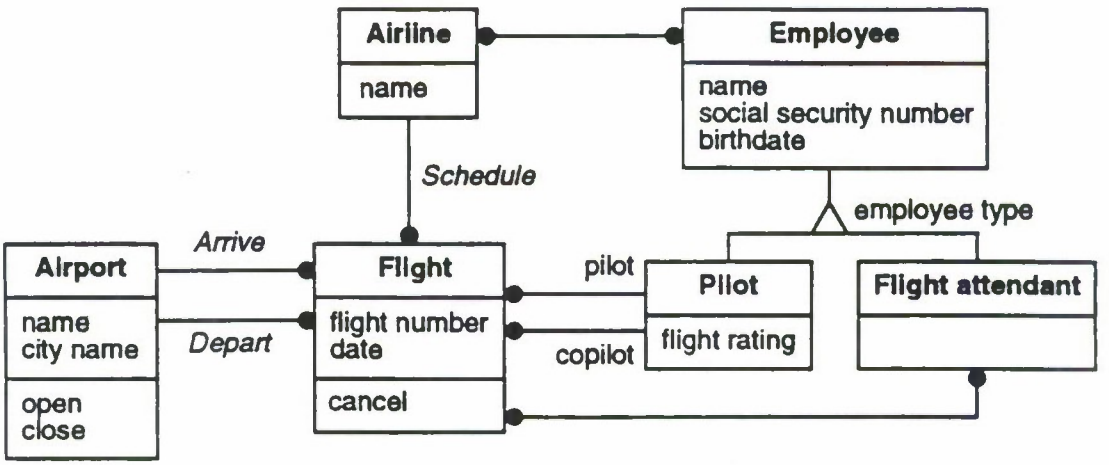
Aggregation



Generalization



Sample OMT Object Model



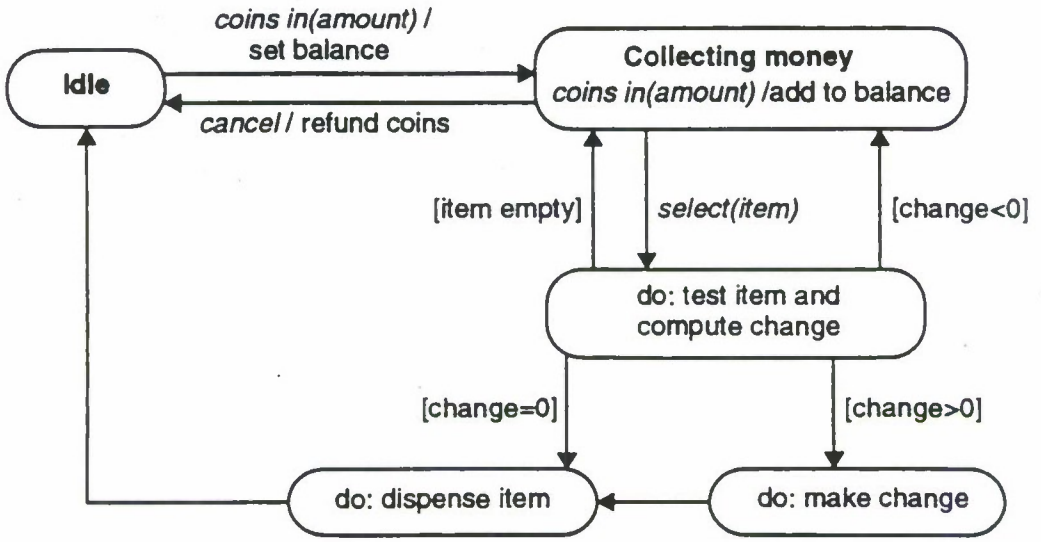
Three Views: The Dynamic Model

- Object Model — Things
  - Static structural relationships of objects.
  - What are the kinds of objects and their relationships?
- ***Dynamic Model — Interactions***
  - ***Temporal interactions of objects.***
  - ***What are the stimuli to objects and their responses?***
- Functional Model — Transformations
  - Functional dependencies of values.
  - What are the computations that objects perform?

Basic Dynamic Modeling Concepts

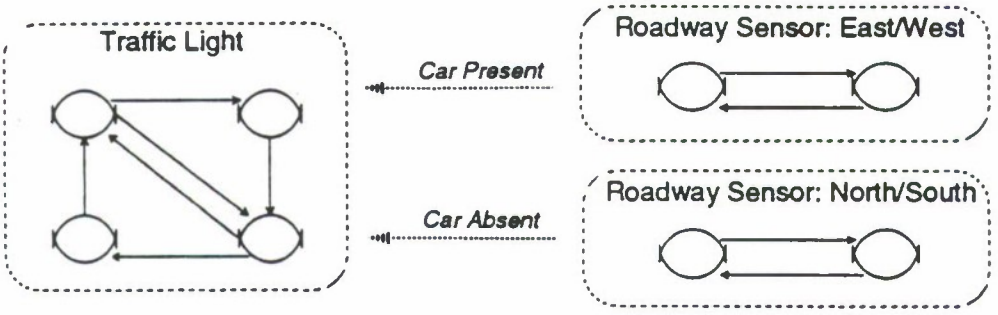
- ***Event***: something that happens at a point in time.
- ***Scenario***: a sequence of events that occurs during one particular execution of a system.
- ***State***: the interval between events.
- ***Transition***: a change from one state to another in response to an event.
- ***Guard condition***: a boolean expression that must be true in order for a transition to occur.
- ***Action***: an instantaneous response to a transition.
- ***Activity***: an ongoing operation within a state.

A Sample State Diagram for a Vending Machine



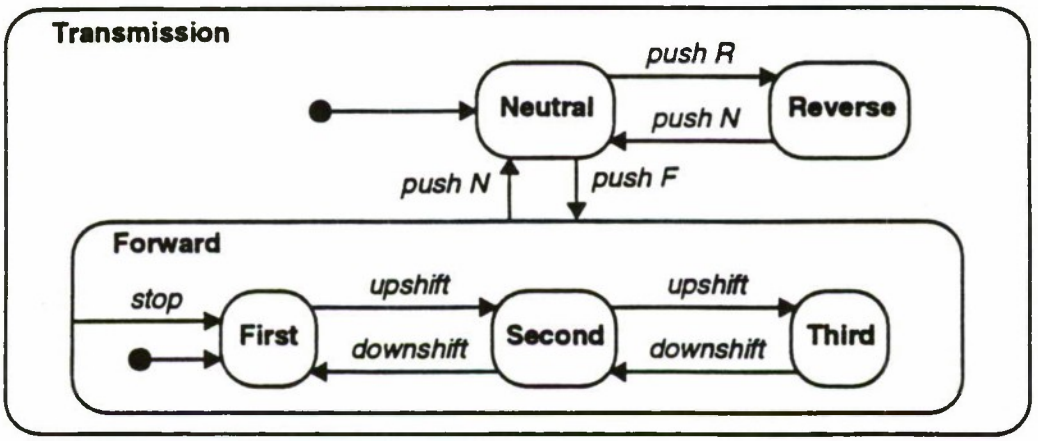
Structure of the Dynamic Model

- **Each object behaves independently, as an autonomous entity**
- **Objects communicate by sending and responding to events**
- **There is a state diagram for each class: sometimes trivial**
- **All objects in a class execute from the state diagram for that class, which models their common behavior**
- **Each instance has its own state; different instances of a class may be in different states and respond individually**



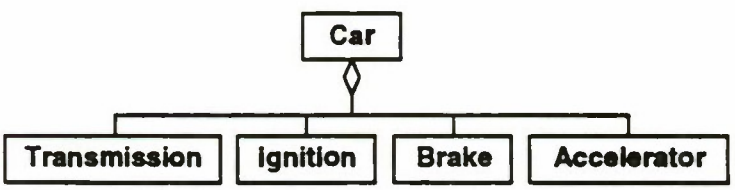
Structured State Diagrams (State Generalization)

- Substates inherit the transitions of their superstates.
  - Shown as a nested contour
  - Notation due to David Harel
- Example for a car transmission



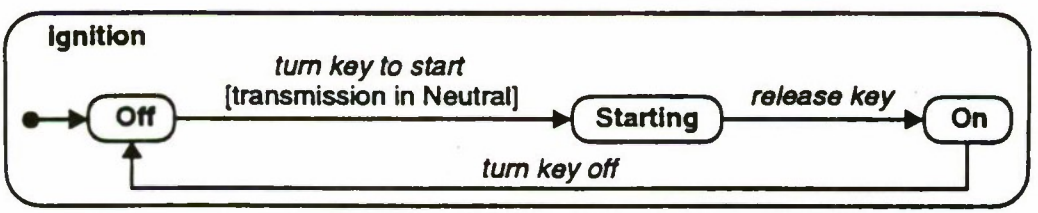
Example of State Aggregation: Car

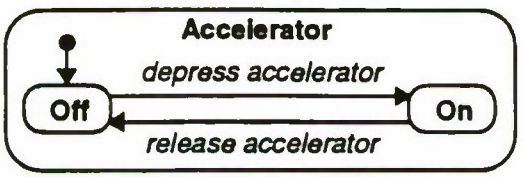
Object Model

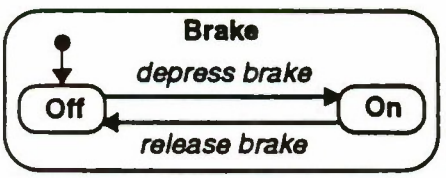


Dynamic Model

Transmission...shown on previous slide







Relation of Object and Dynamic Models

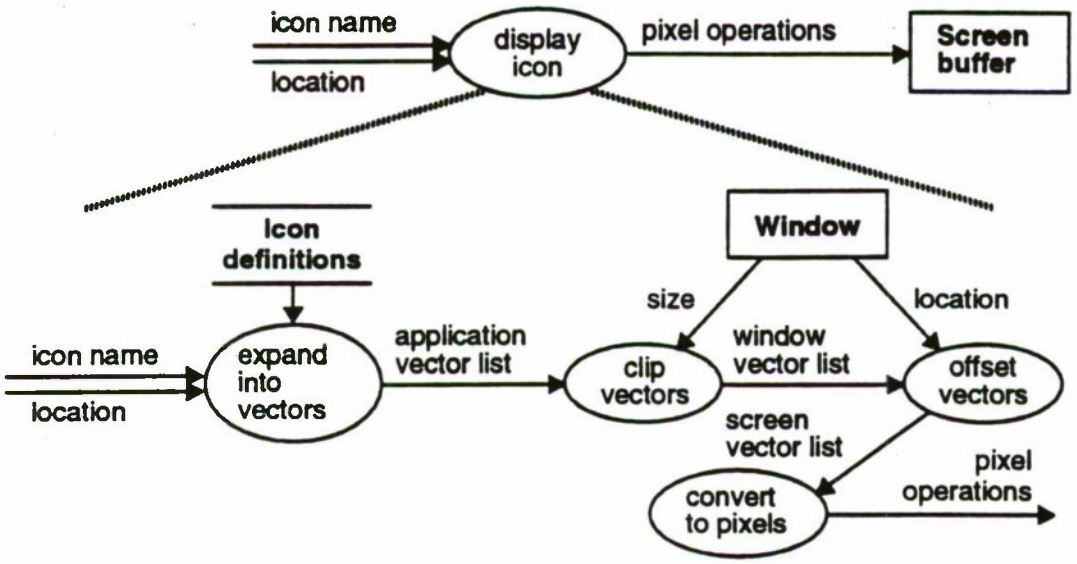
- The dynamic model specifies allowable sequences of changes to objects from the object model.
- One state diagram for each object class with significant behavior.
- Objects communicate by sending and responding to events.
- Events, actions, and activities correspond to operations on the object model.
- States describe stages in the life on an object.
  - The state of object varies while class usually does not.
- The dynamic model of a class is inherited by its subclasses.

Three Views: The Functional Model

- Object Model — Things
  - Static structural relationships of objects.
  - What are the kinds of objects and their relationships?
- Dynamic Model — Interactions
  - Temporal interactions of objects.
  - What are the stimuli to objects and their responses?
- ***Functional Model — Transformations***
  - ***Functional dependencies of values.***
  - ***What are the computations that objects perform?***

Nested Data Flow Diagrams

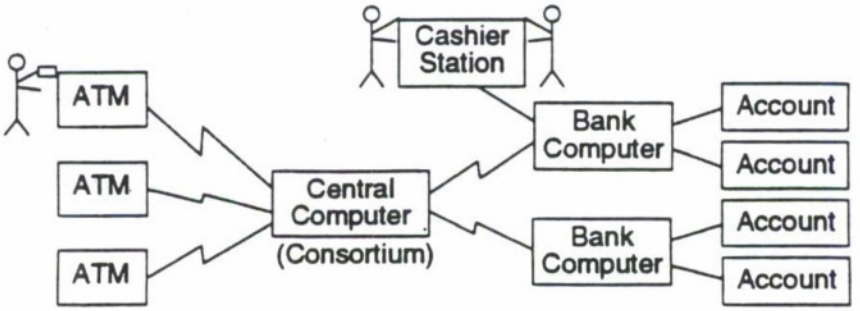
- Processes in data flow diagrams can be recursively expanded into finer degrees of detail.



Functional Model: Practical Tips

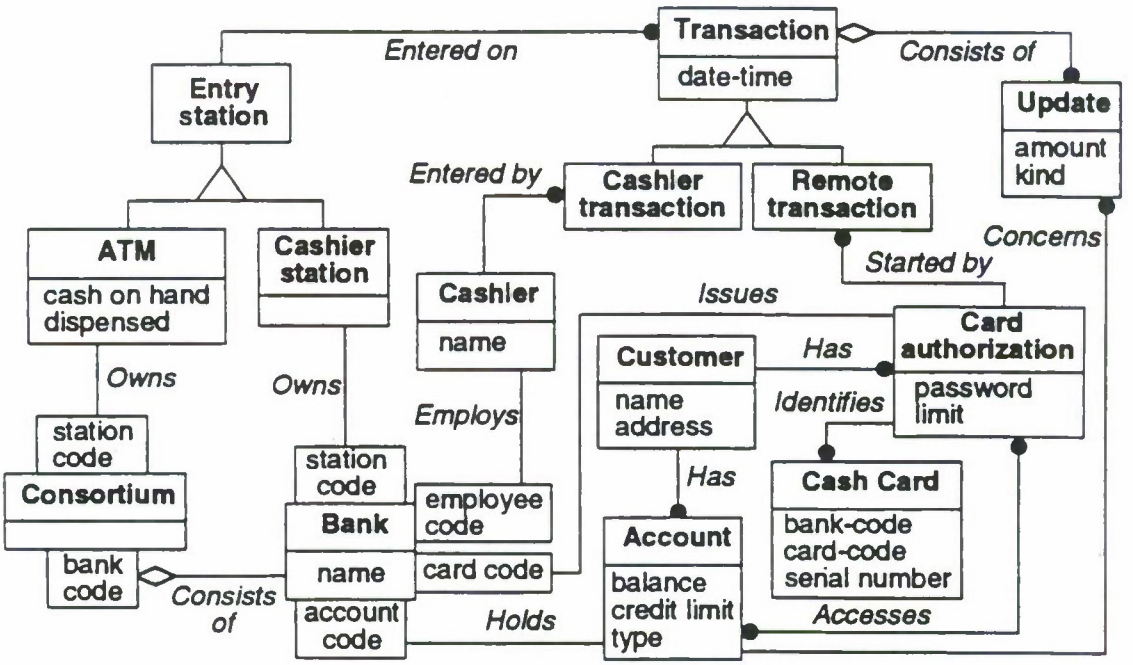
- Use data flow diagrams to document system level computations.
- For some operations, pseudocode is simpler and clearer than DFDs.
- Consider other representations for capturing data dependencies, such as decision tables, constraints, and mathematical equations.
- Do *not* model a system by constructing data flow diagrams and adding object diagrams as an afterthought.

The ATM Example: Requirements Statement

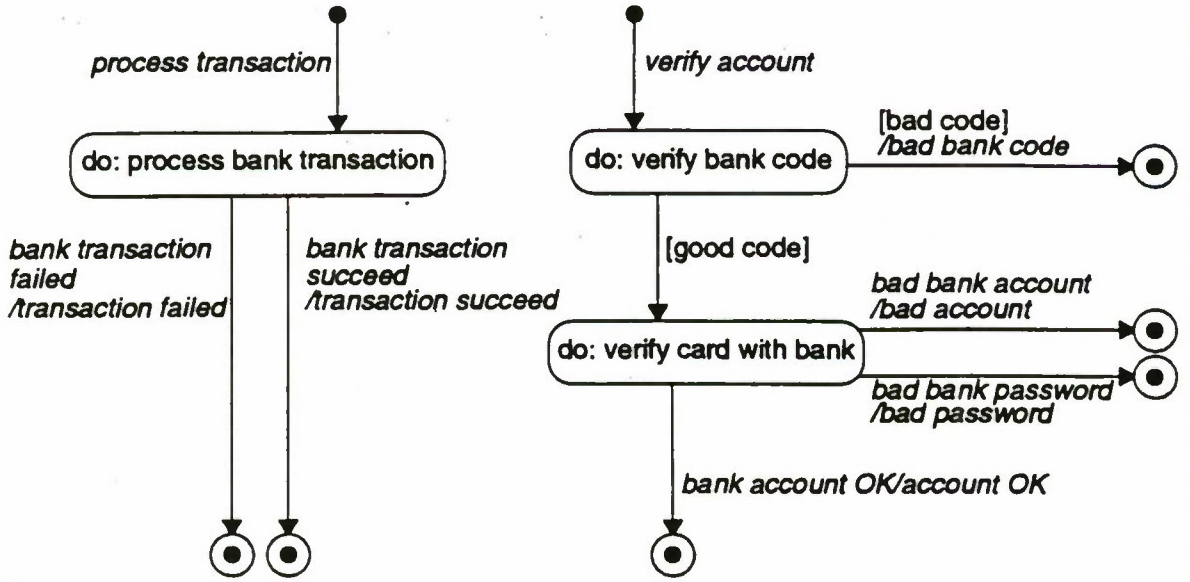


Design the software to support a computerized banking network including both human cashiers and automatic teller machines (ATMs) to be shared by a consortium of banks. Each bank provides its own computer to maintain its own accounts and process transactions against them. Cashier stations are owned by individual banks and communicate directly with their own bank's computers. Human cashiers enter account and transaction data. Automatic teller machines communicate with a central computer which clears transactions with the appropriate banks. An automatic teller machine accepts a cash card, interacts with the user, communicates with the central system to carry out the transaction, dispenses cash, and prints receipts. The system requires appropriate recordkeeping and security provisions. The system must handle concurrent accesses to the same account correctly. The banks will provide their own software for their own computers; you are to design the software for the ATMs and the network.

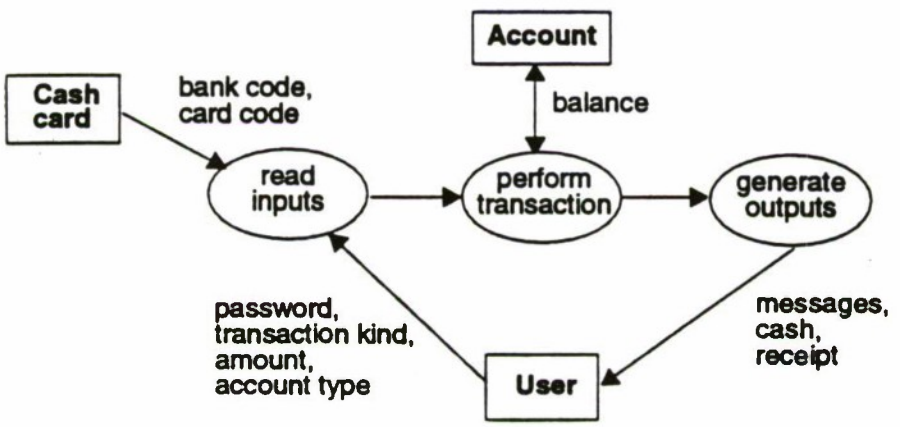
ATM Object Model



State Diagram for *Consortium* Class



Top Level Data Flow Diagram for ATM



Variable Resolution in the Object Model

- **Aggregation** (Part / Whole Hierarchy):
  - Composite object ("mega-object") may be defined in terms of its parts
  - Each part may have independent attributes and operations
  - The composite object responds to operations by invoking operations of its lower-level parts (delegation or forwarding)
  - The configuration of a composite object may change over time
- **Generalization** (Classification Hierarchy):
  - An object's attributes and behavior may be defined at several levels of abstraction (from specific to general)
  - Objects are classified in terms of their similarities and differences
  - General behavior / attributes are shared in common by subclasses
  - Specialized behavior of a subclass overrides more general "default" behavior

Variable Resolution in the Dynamic Model

- **State Generalization** (Substates):
  - Allows fine-grained specification of behavior within a coarser-grained "mode of operation"
- **State Aggregation** (Concurrent submachines):
  - Defines collective behavior of a composite object in terms of independent parts
  - Separates the specifications of orthogonal behaviors occuring concurrently
  - Greatly simplifies state machines by reducing the cross-product of orthogonal machines
- **Event Generalization** (Subevents):
  - Events may be classified at several levels of generality
  - Use more general events when fine-grained distinctions don't matter

Variable Resolution in the Functional Model

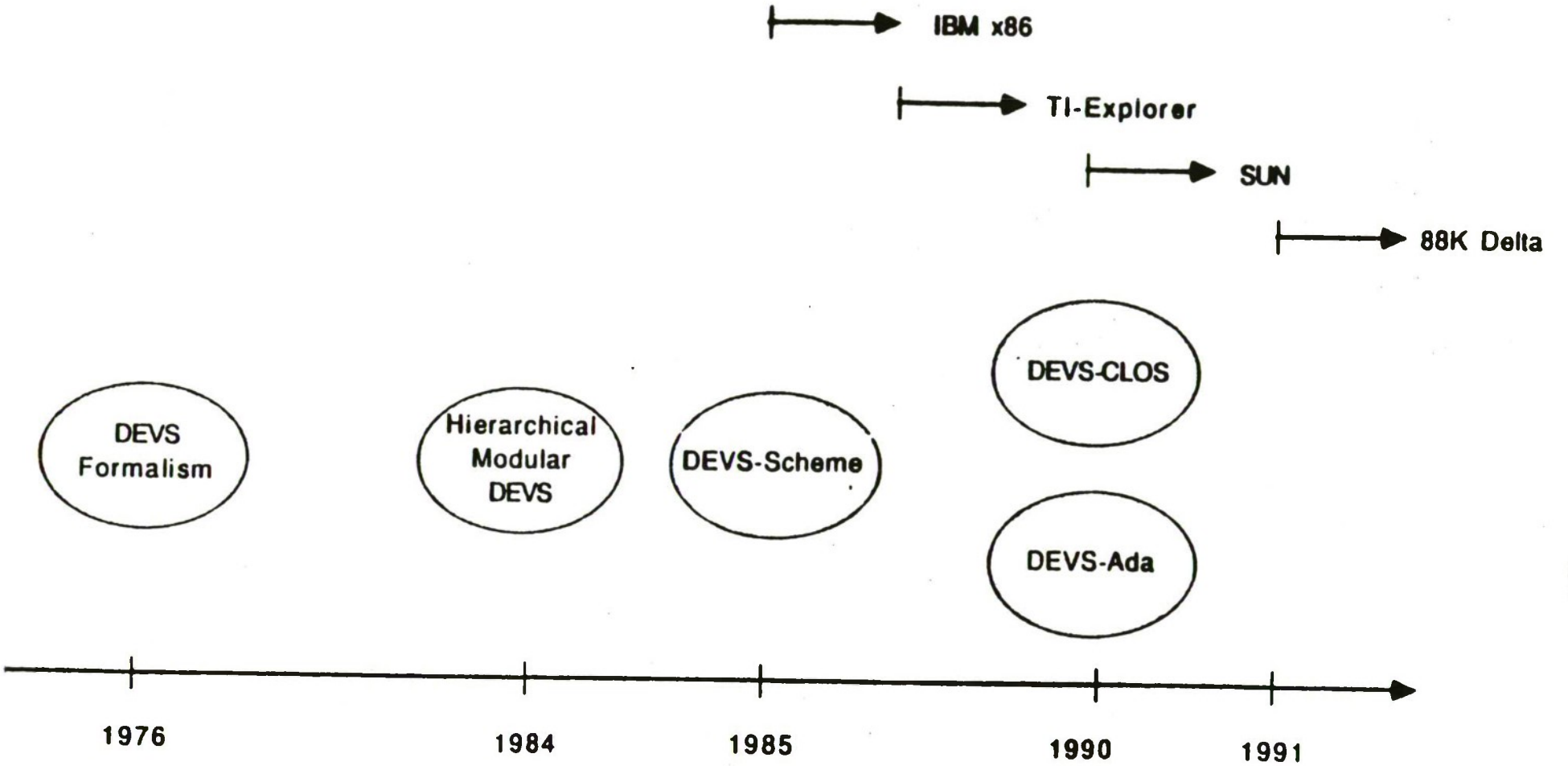
- **Functional Decomposition** (Nested Processes):
  - Defines procedures at multiple levels of resolution
  - Promotes a top-down approach to model definition and understanding
  - May be defined in several notations: (e.g., DFD's, pseudo-code, behavior diagrams, etc.)

MODULAR HEIRACHICAL MODEL REPRESENTATION

B. P. ZEIGLER

Summary of Formal Properties of the DEVS Formalism

- Modularity
- Closure under Coupling
- Hierarchical Construction
- Stand-alone and Bottom-up Testability
- Experimental Frame/Model Separation



53A

Figure. Evolution of DEVS

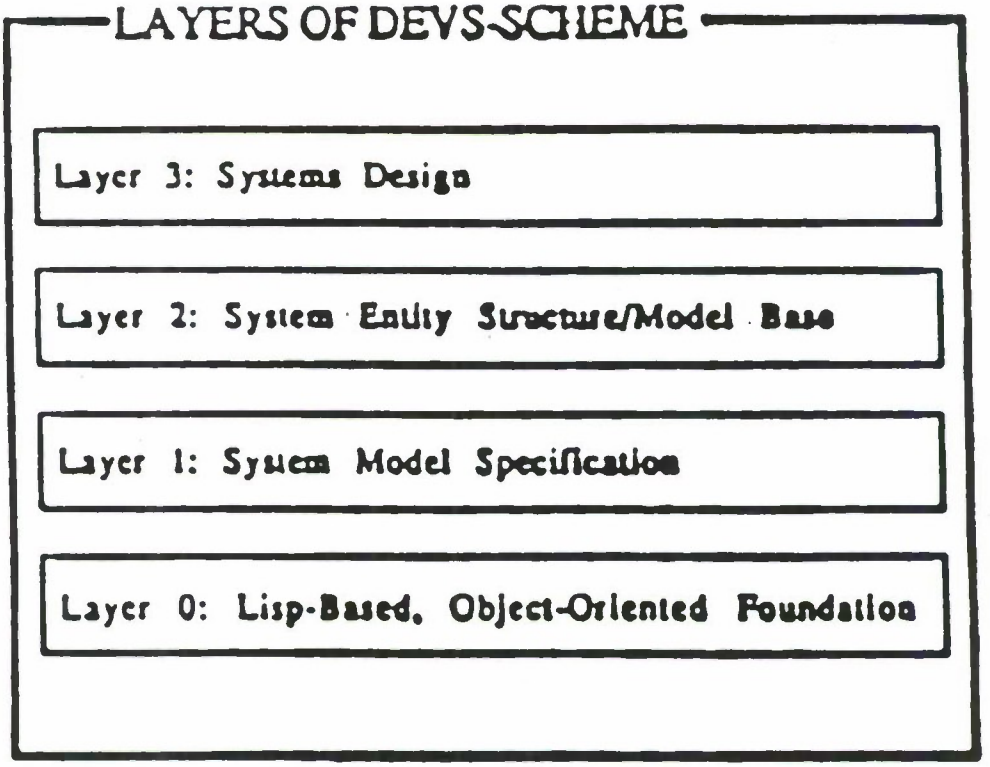


Figure1 Layers of DEVS-Scheme

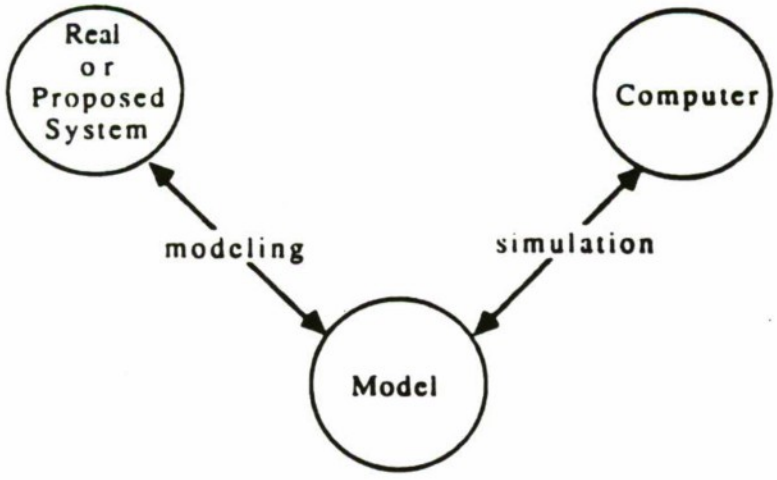
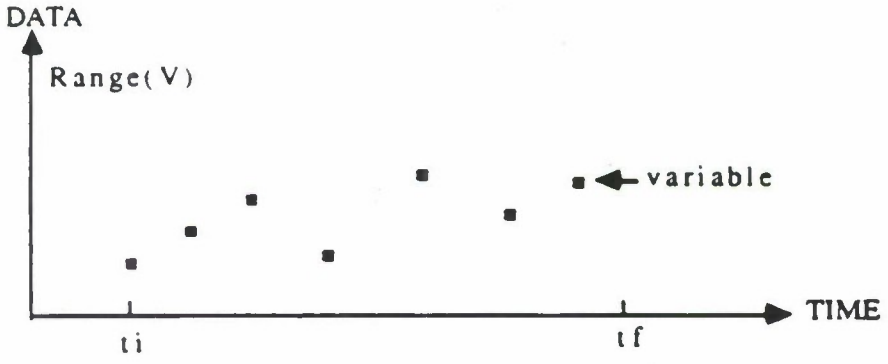


Figure 3.1 Entities and relations in simulation

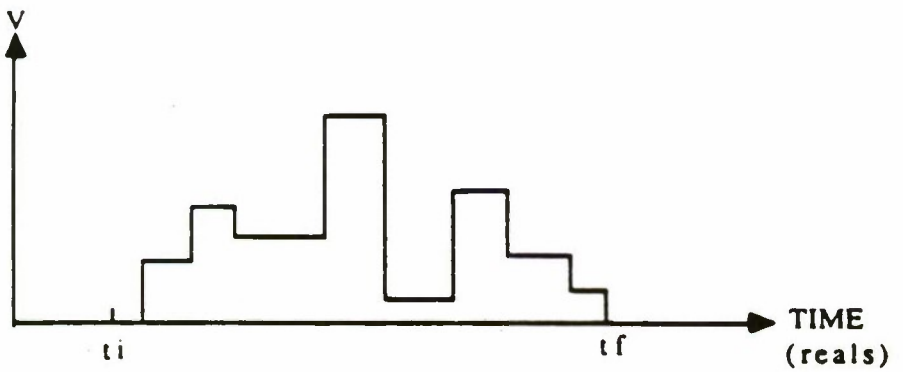


Range(V): set of values that V can assume

ti: initial (starting) time

tf: final (terminating) time

Figure 3.2 Generalized data segment produced by a system or model



Event -> changes in value V

Figure 3.3 Discrete event time segment

Atomic-Model Specification

An atomic discrete event systems (DEVS) is a structure [6,7]

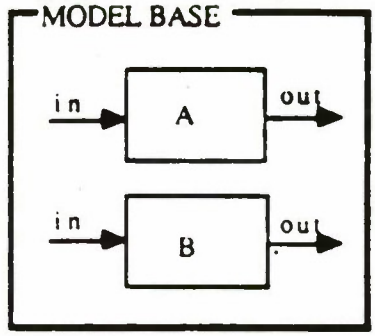
$$DEVS = (X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta)$$

where

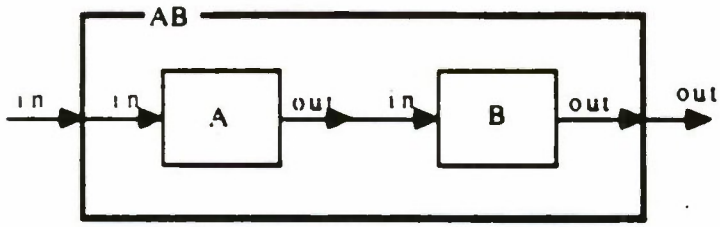
X	is the set of external events
Y	is the set of outputs
S	is the set of sequential states.
$\delta_{ext} : Q \times X \rightarrow S$	is the external transition function
$\delta_{int} : S \rightarrow S$	is the internal transition function
$\lambda : S \rightarrow Y$	is the output function
$ta : S \rightarrow \Re_0^+ \cup \infty$	is the time advance function

and

$Q := \{ (s, e) \mid s \in S, 0 <= e <= ta(s) \}$ is the set of *total states*.



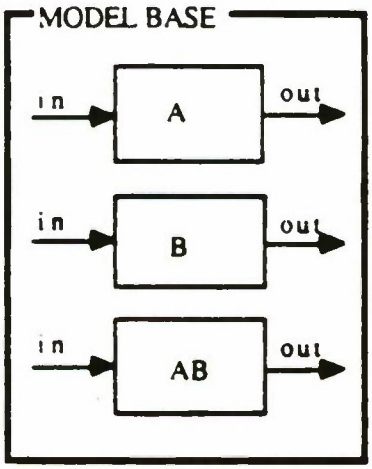
(a)



Coupling:

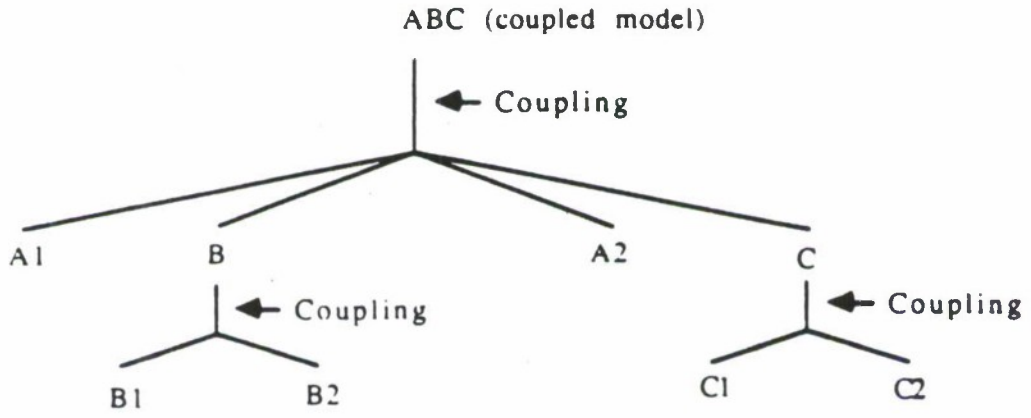
external input: AB.in -> A.in
external output B.out -> AB.out
internal: A.out -> B.in

(b)



(a)

Figure 2.4 Model base concept



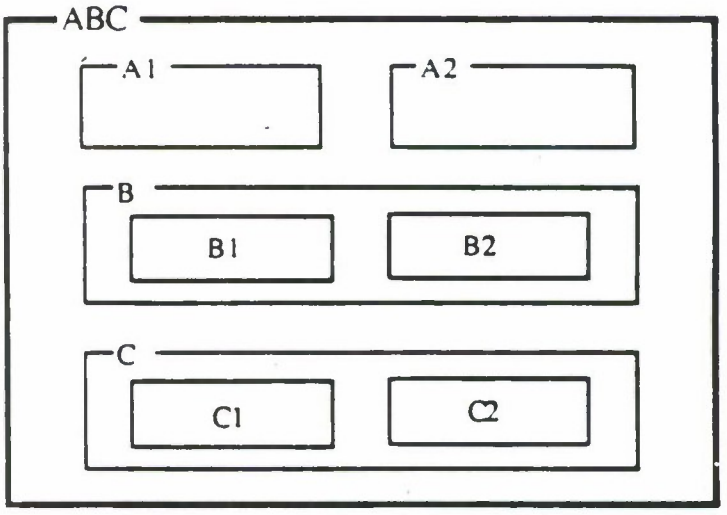
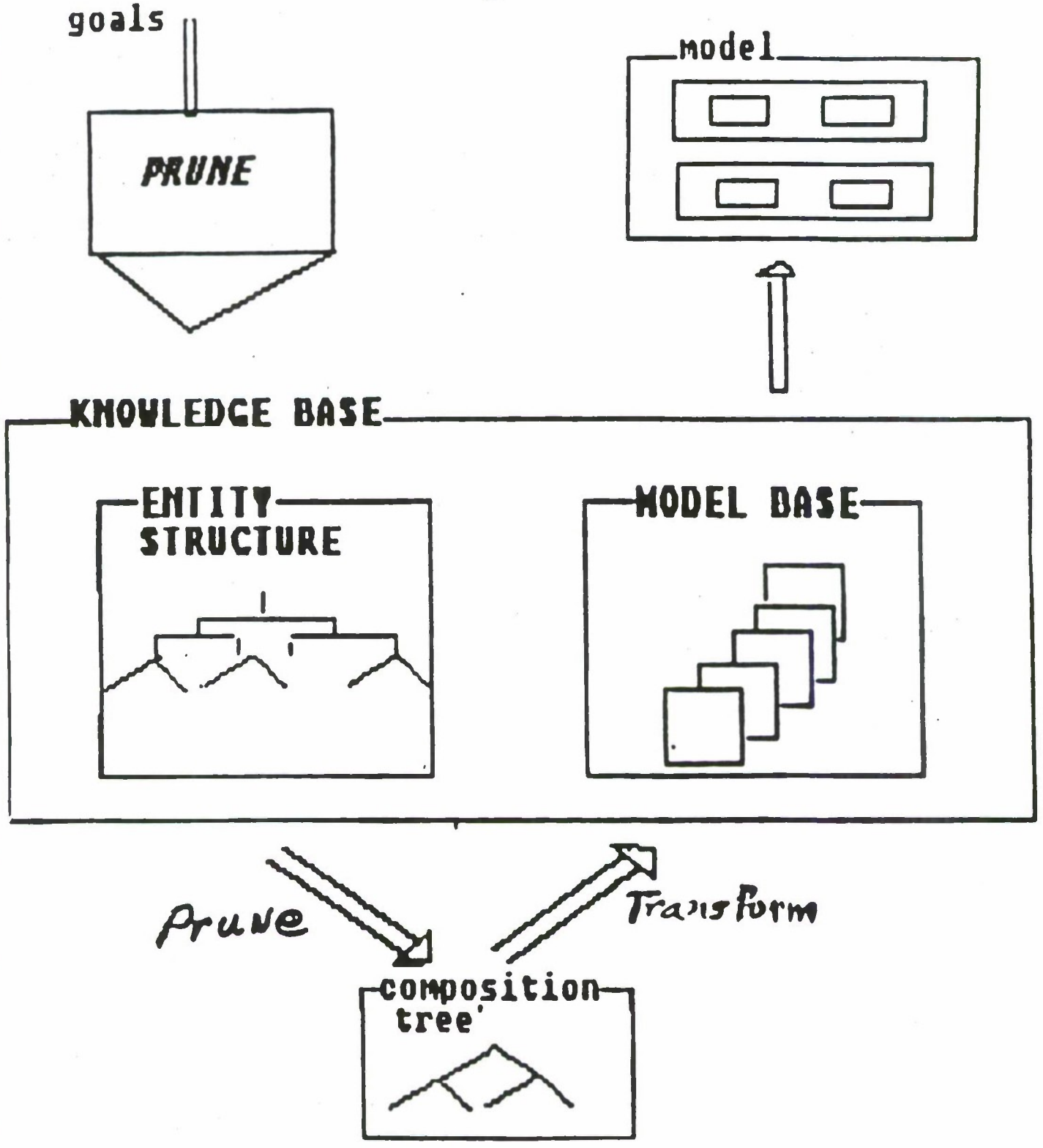


Figure 2.5 Composition tree

Derivation of Experimental Frame Elements

- Based on Modeler's objectives
  - input variables required
  - output variables
  - summary statistics of interest
  - domain of operation


goals
PRUNE
model
KNOWLEDGE BASE
ENTITY STRUCTURE
MODEL BASE
Prune
Transform
composition tree

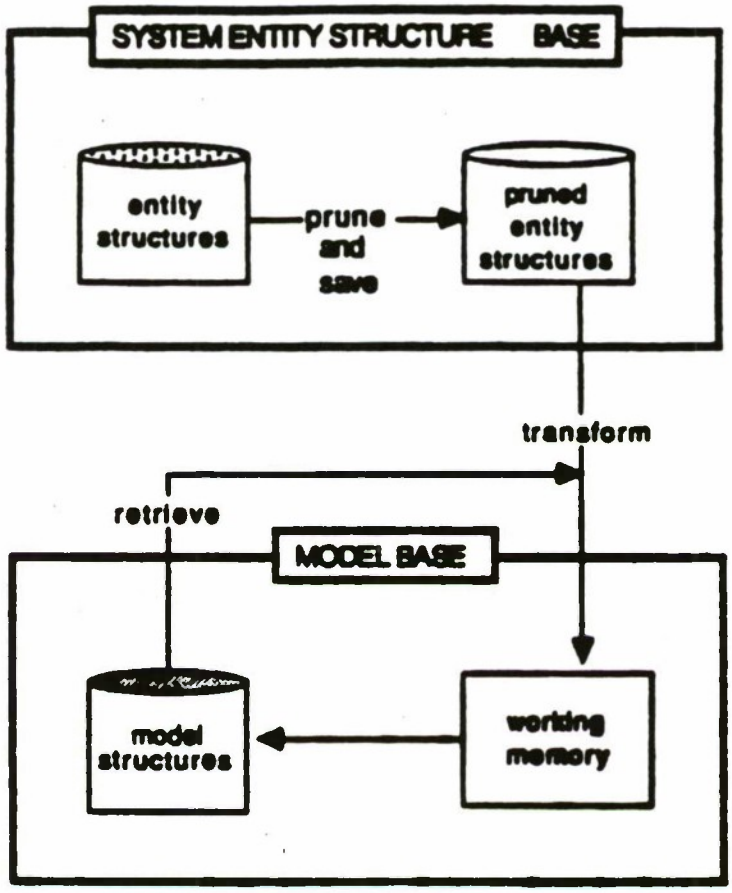
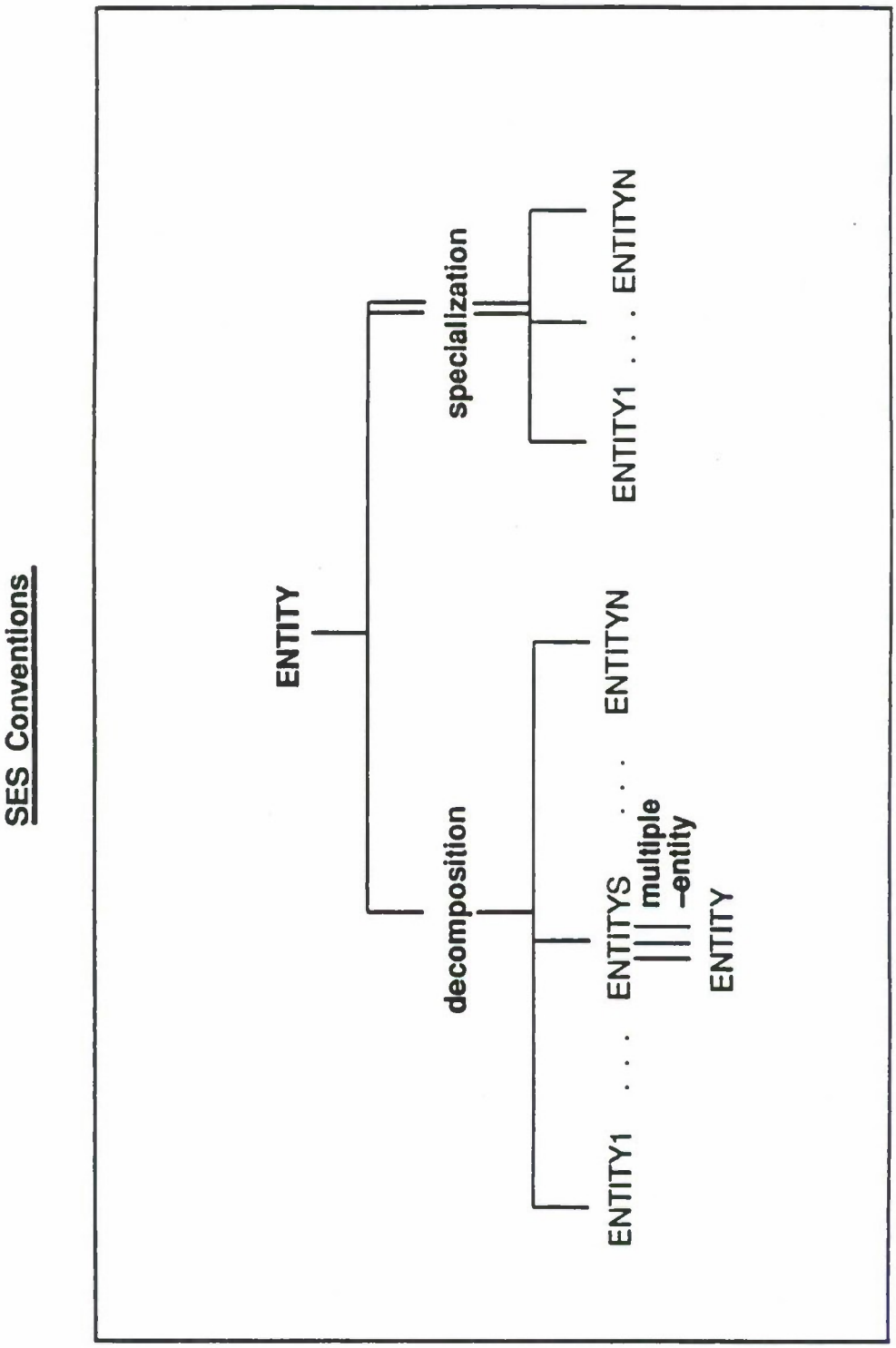
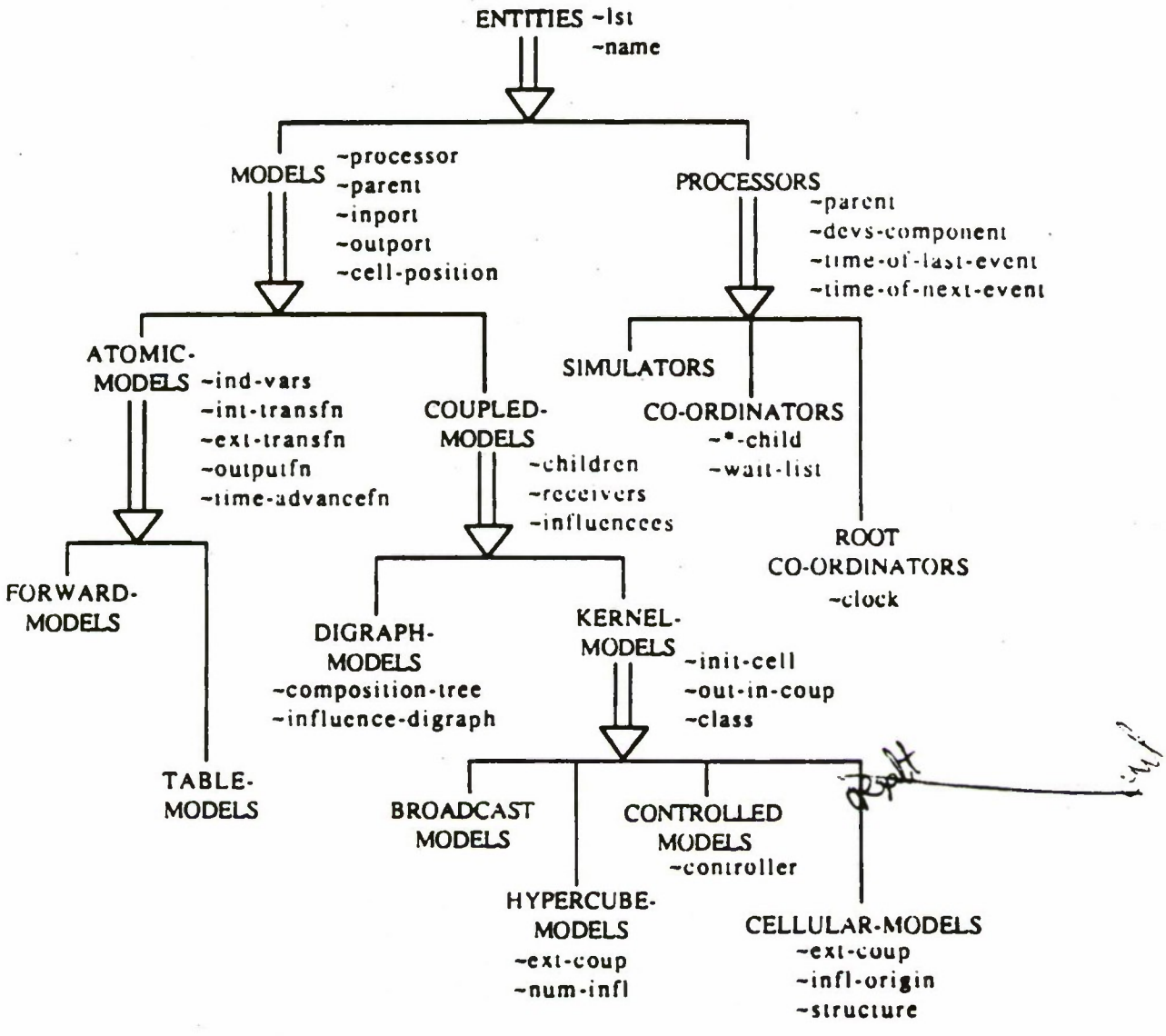


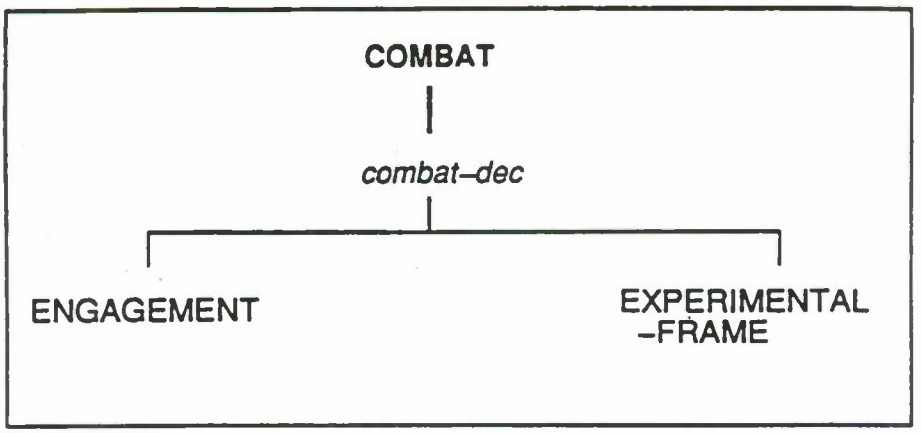
Figure 1. SES/MB environment

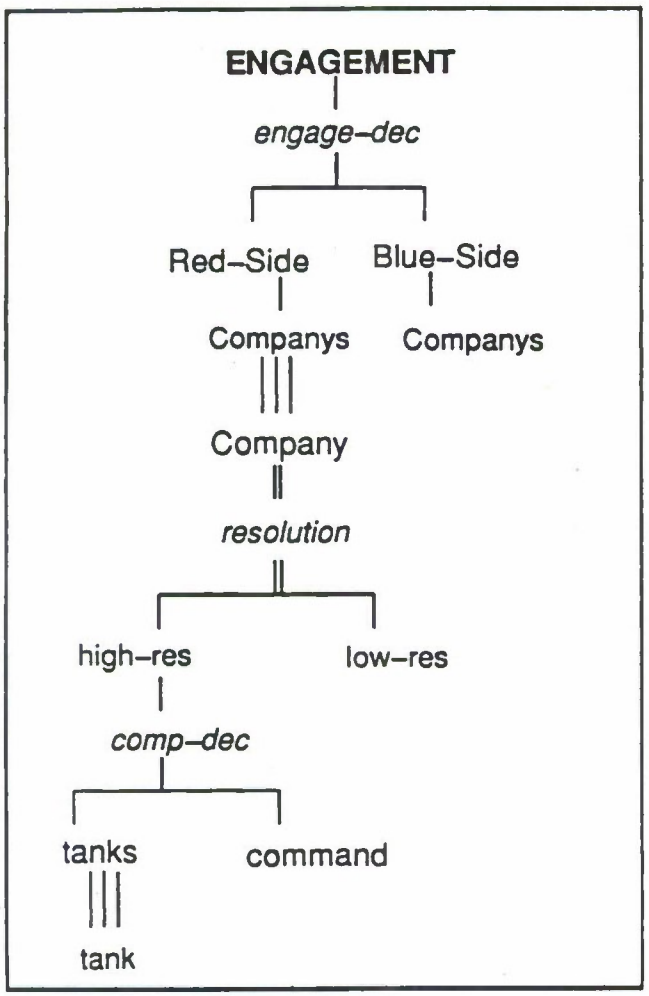

SES Conventions
ENTITY
specialization
ENTITY1 . . . ENTITYN
decomposition
ENTITY1 . . . ENTITYS . . . ENTITYN
multiple
entity
ENTITY

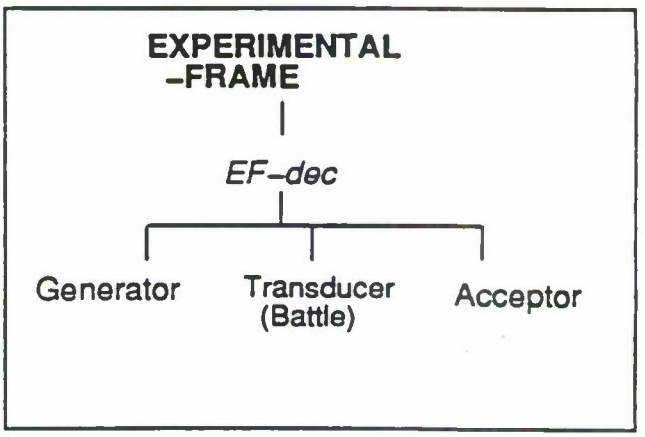


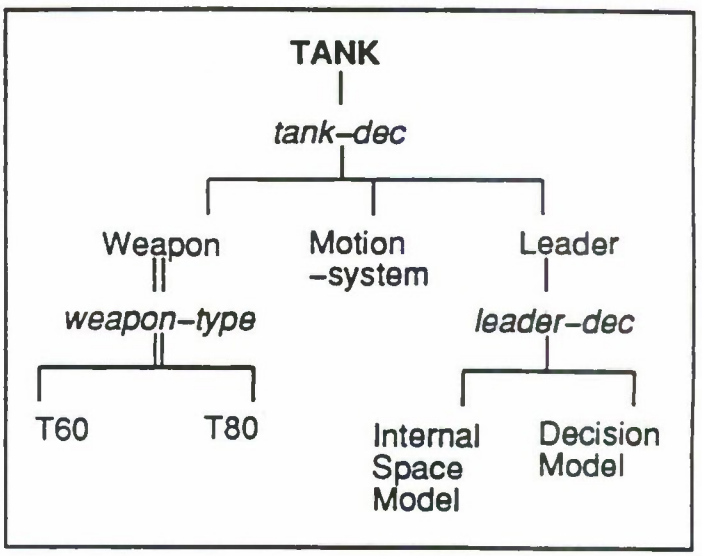
- Uppercase Letters: Classes
- Lowercase Letters: Class/Instance Variables

Figure 3.6 Class Hierarchy of DEVS-Scheme

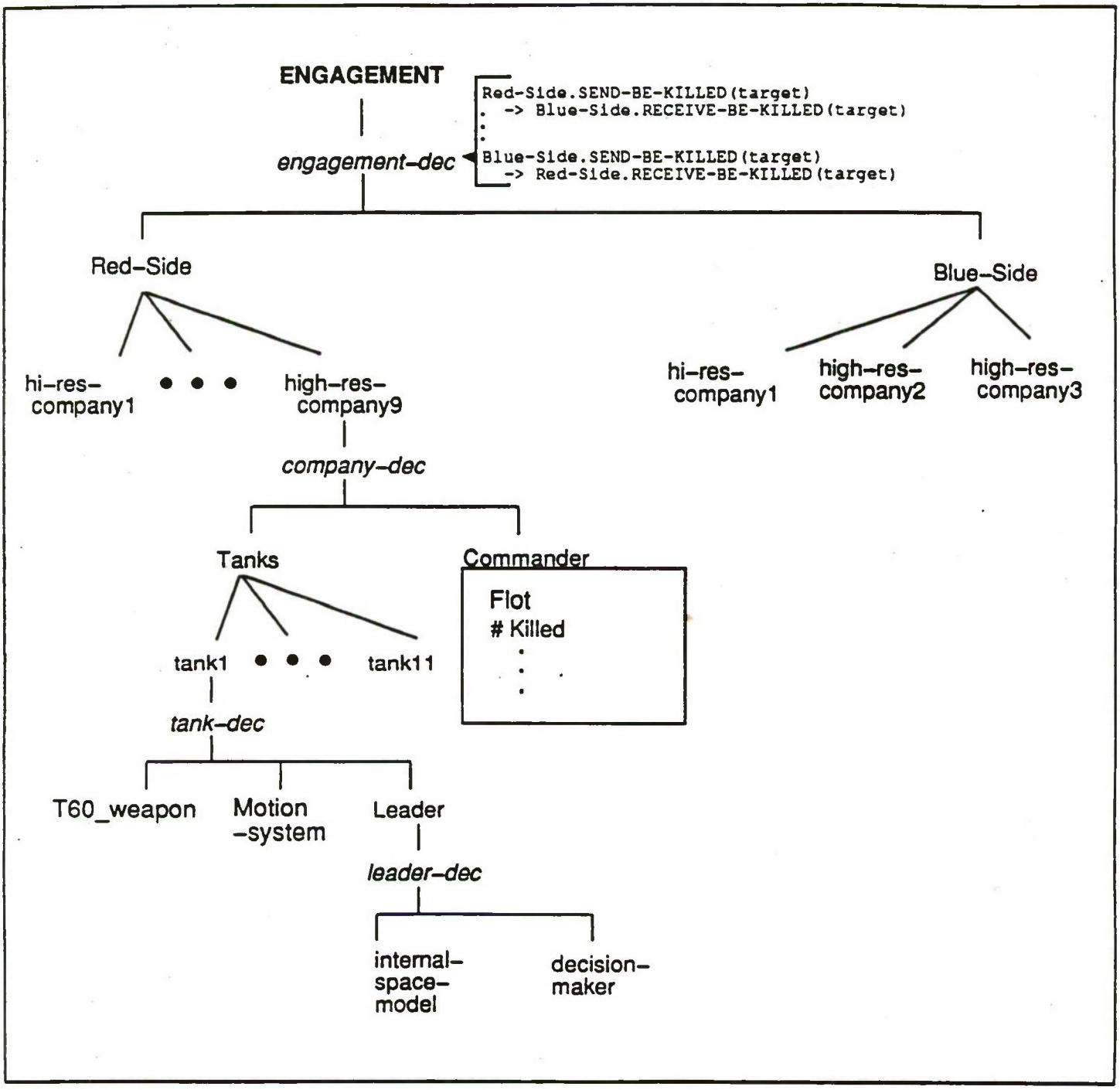




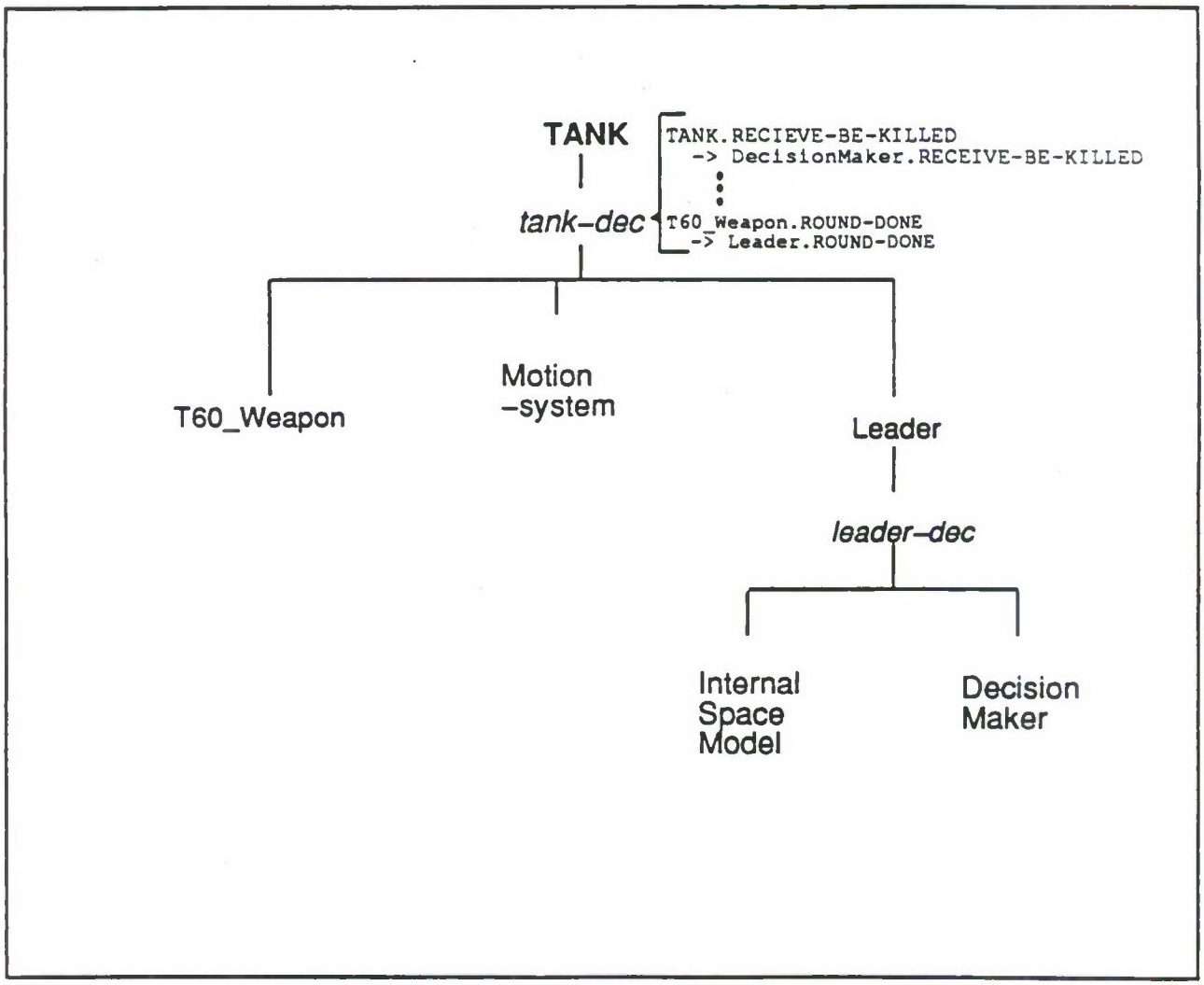




System Entity Structure for Combat



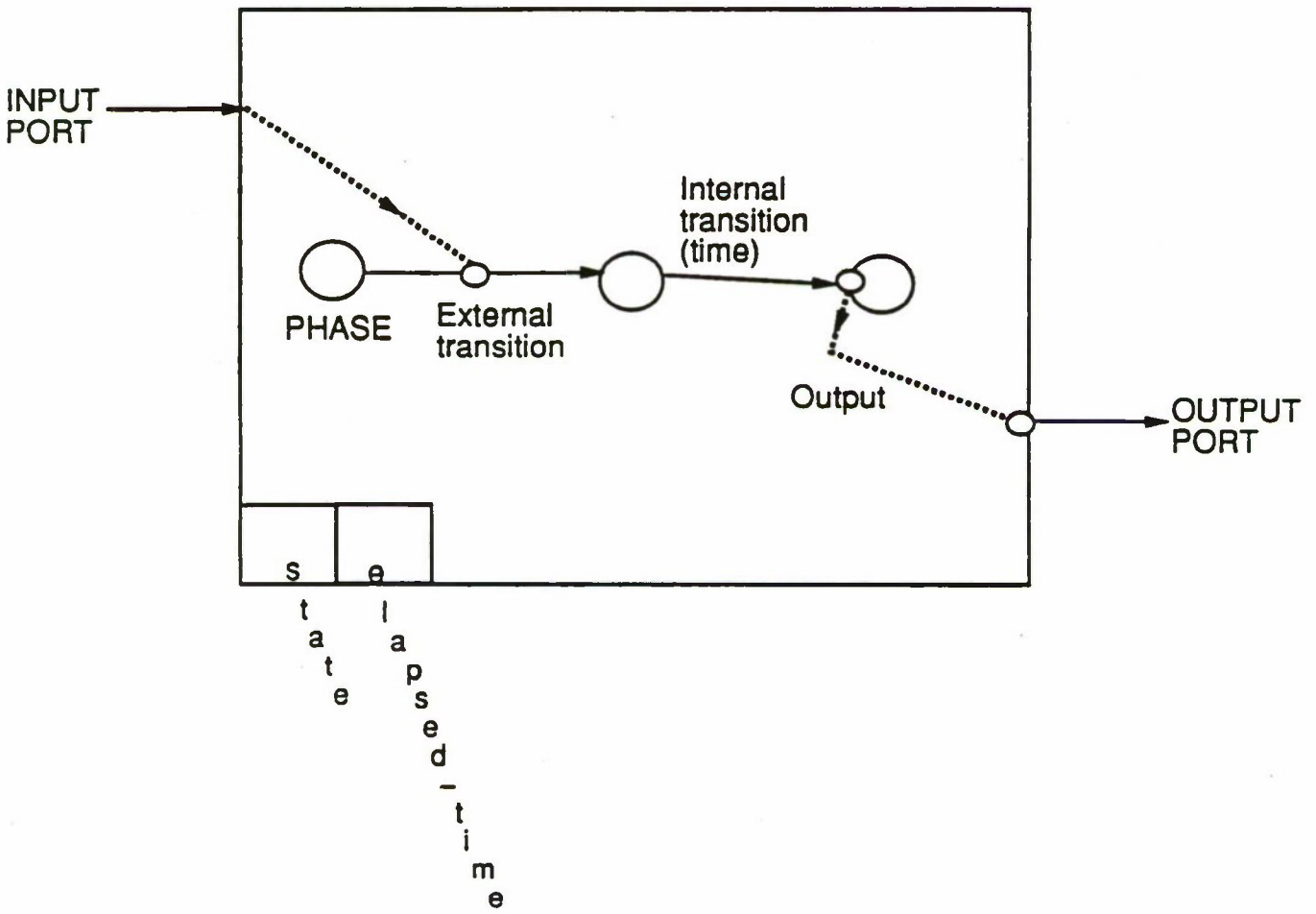
Pruned Entity Structure for Regement on Battalion Engagement

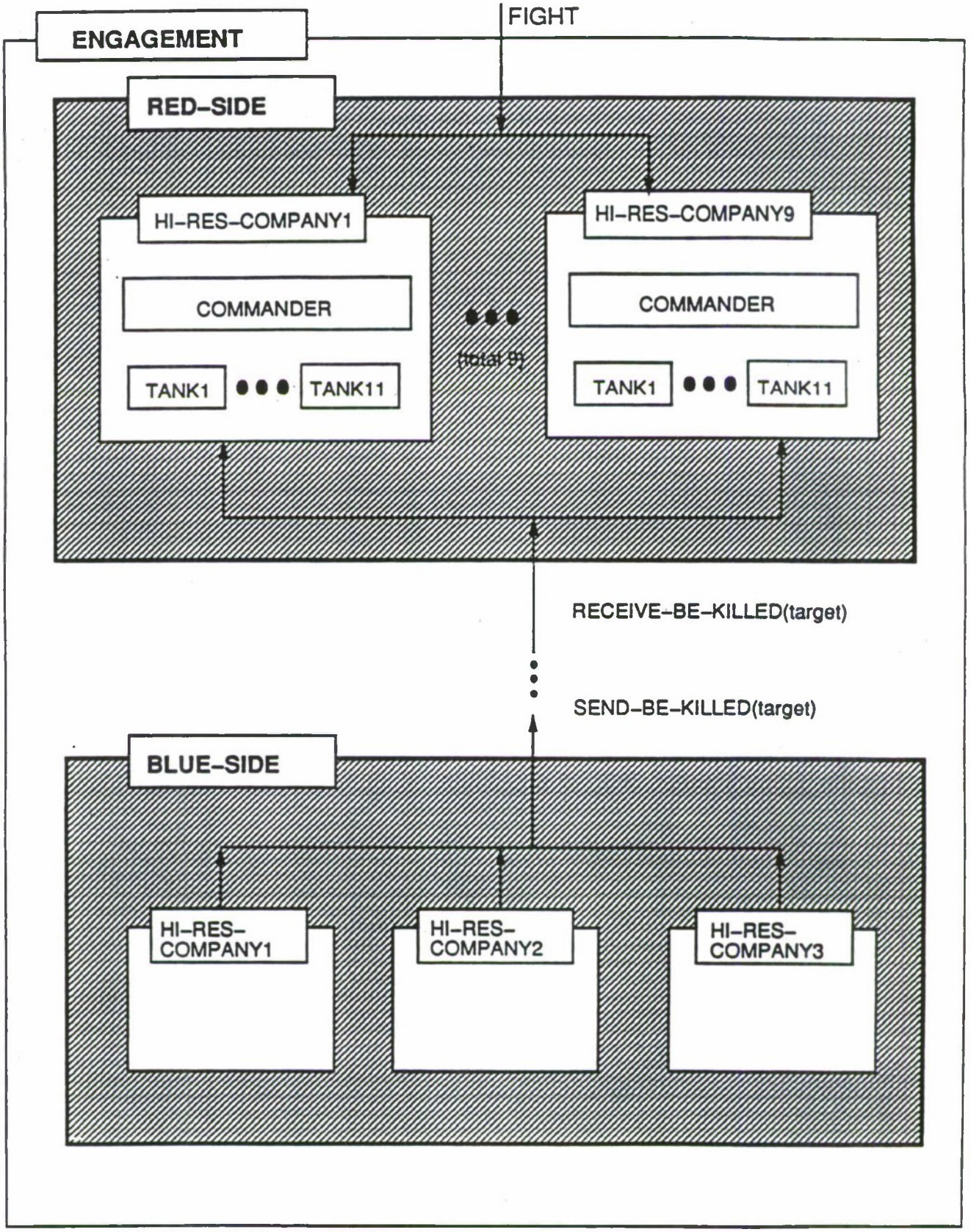


Pruned Entity Structure of TANK

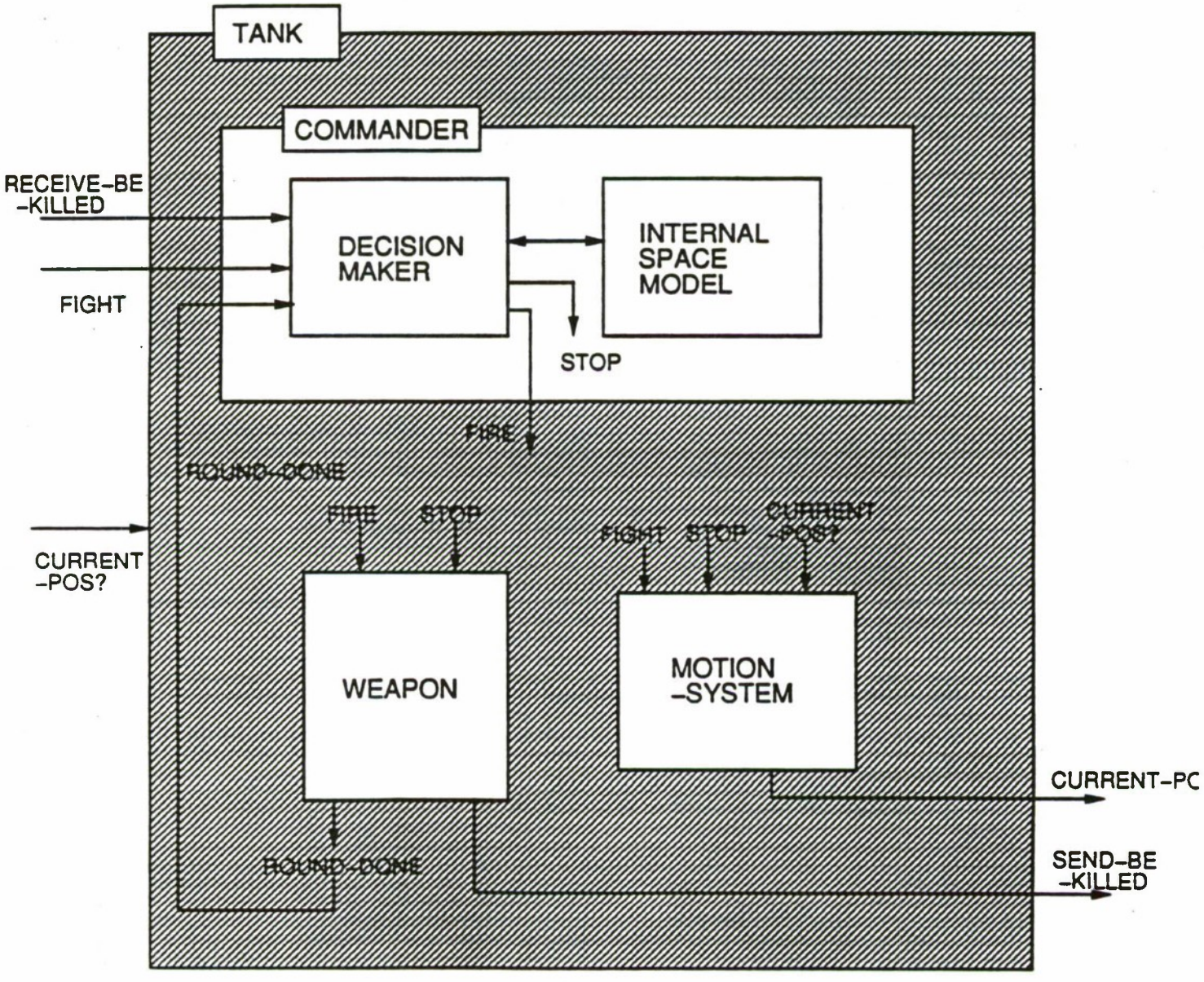
ATOMIC MODEL

State Diagram Conventions

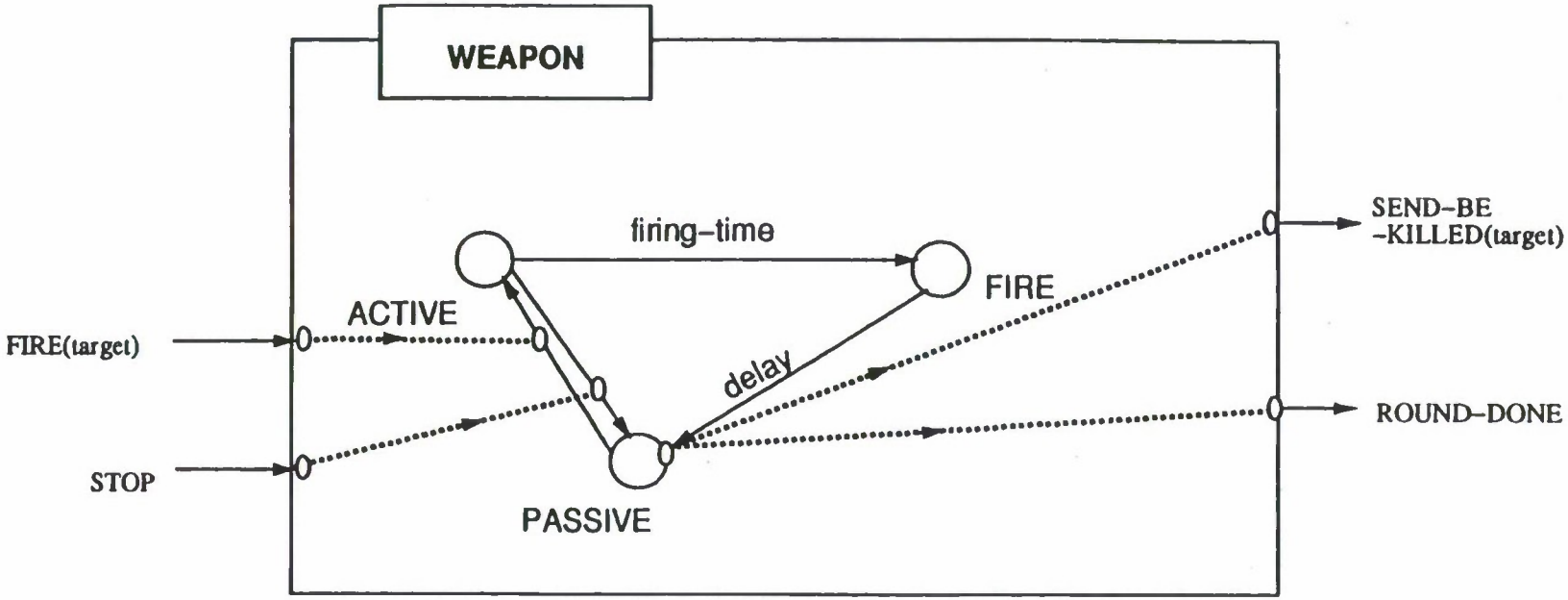




Hierarchical Model Transformation of Pruned Entity Structure of Combat



Transformed Hierarchical Model of TANK

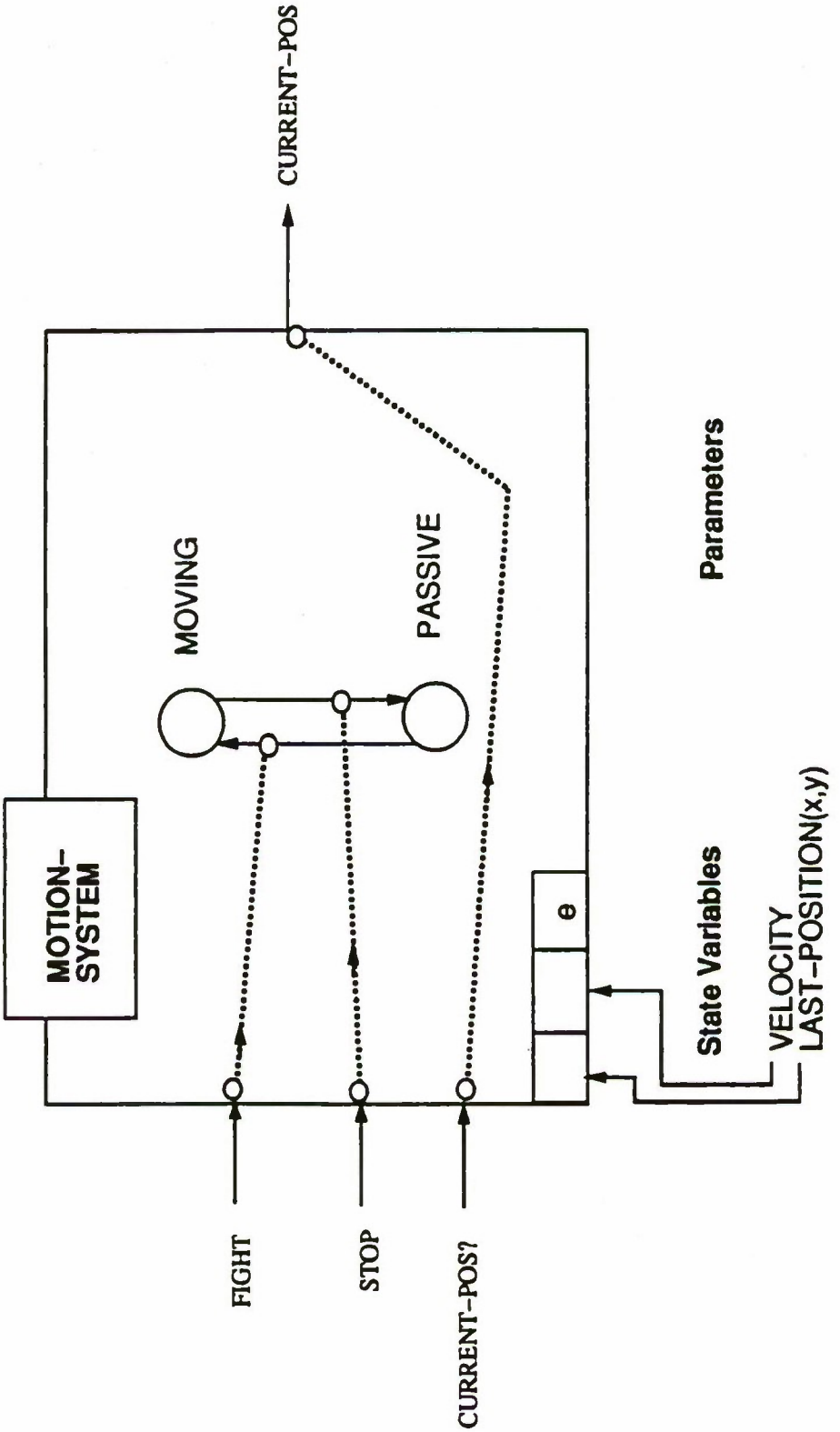


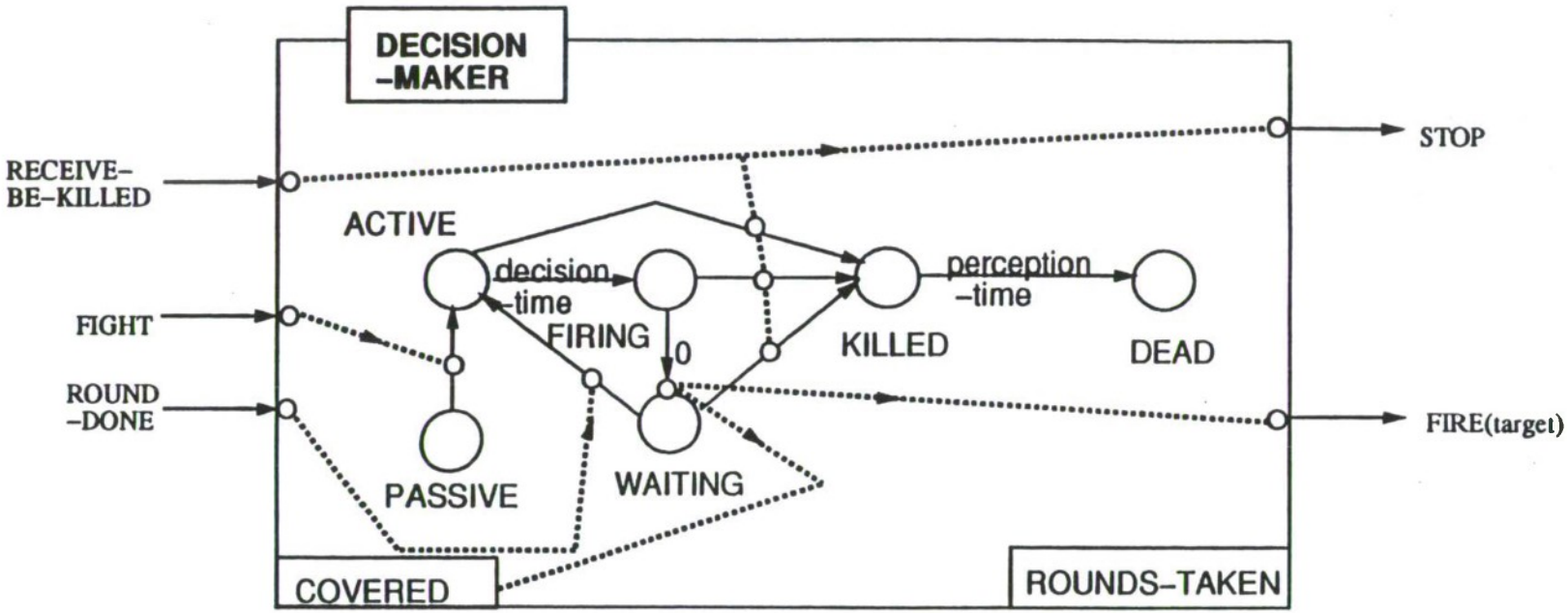
State Variables

CURRENT–ROUNDS

Parameters

FIRING–RATE
ROUND–VELOCITY
PK (PROB. OF KILL)
RANGE


CURRENT-POS
MOVING
PASSIVE
Parameters
MOTION-
SYSTEM
θ
State Variables
VELOCITY
LAST-POSITION(x,y)
FIGHT
STOP
CURRENT-POS?



State Variables

COVERED

ROUNDS-TAKEN

Parameters

PK
RANGE
FIND-TARGET(RANGE, PK)

SYSTEM THEORY

ADDENDUM

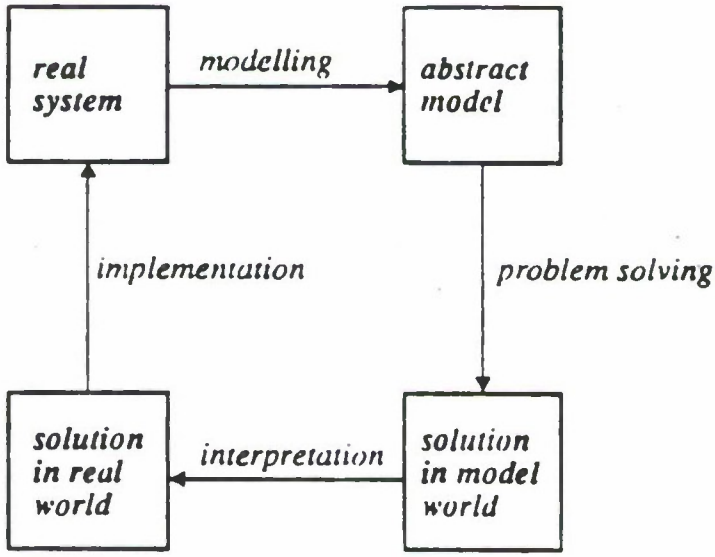


Figure 1.1 general problem solving process

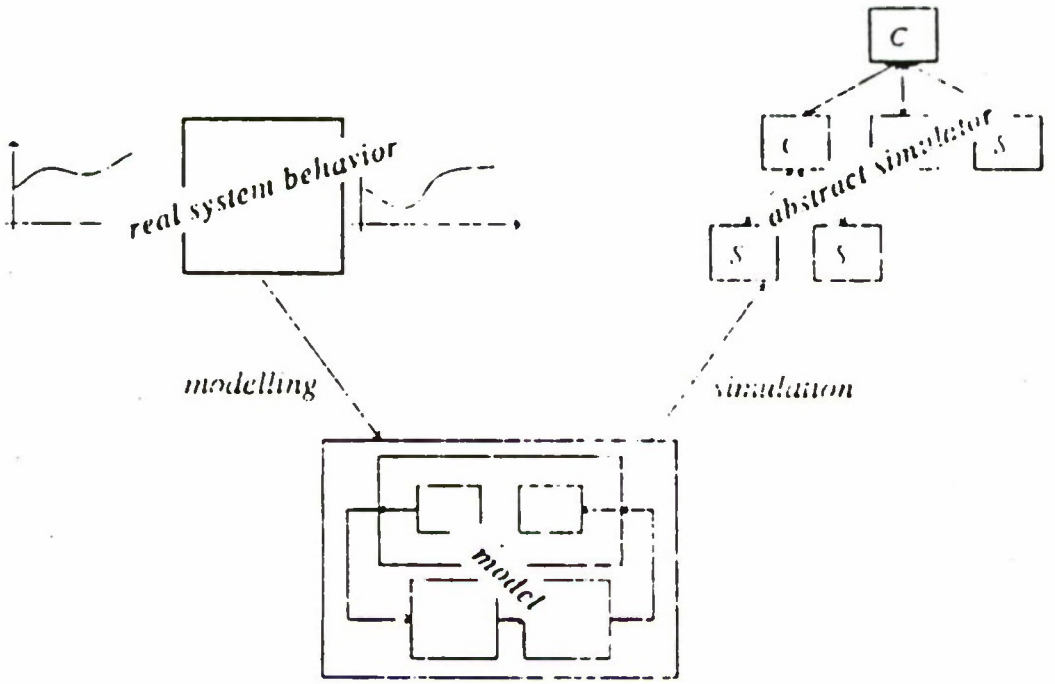


Figure 1.10 Real system, model and computer from the systems theory point of view

Level 0: Observation Frame $O = <T, X, Y>$ Using an observation frame O we just define the system boundary. The set X is the input interface in form of a *set of inputs* and Y is the output interface in form of a *set of outputs*. The set T is called the *time base*. It is used to order events and represents the observation times, i.e., points in time when we are able to define system behavior and system dynamic. Usual time bases are the real numbers - the *continuous time base* - or the integers - a *discrete time base*.

Level 1: Relation Observation IORO = $<T, X, W, Y, R>$ The relation observation *IORO* defines the behavior of the system using a relation of possible input values at particular times and their possible output values. Hence there is not an unequivocal mapping of input values to outputs but for a particular input at a particular time there can be several different outputs.

Level 2: Function Observation IOFO = $<T, X, W, Y, F>$ This level is equal to level 1 with the only difference being that the relation R is partitioned into a set F of functions f and each function defines a unique output response for a particular input segment. Hence, when we have knowledge of the function f, we have knowledge of the "initial state" to get a unique output response.

Level 3: Input / Output System for dynamic system IOS = $<T, X, W, Q, Y, \Delta, \Lambda>$ Whereas in level 0 to 2 it only was possible to represent the system boundary and the system behavior, in this level we are able now to model the internal of the system. For that we use the set of states Q. The *global state transition function* Δ represents the dynamic of the system. The output function Λ is used to model how the current state manifests at the output interface. This level incorporates our most important modelling concept for simulation modelling and therefore we also denote an input / output system *dynamic system*.

Level 4: Structured Input / Output System SIOS = $<T, X, W, Q, Y, \Delta, \Lambda>$ A structured input / output system is an input / output system where the set of inputs, outputs, and states are multi-variable sets, i.e., that we are able to identify several input, output and state variables.

Level 5: Coupled System CS = $<T, X, Y, Components, Coupling>$ In a coupled system the interior of the system is represented by identifying several component systems and couplings between them. The coupling defines how the coupled system inputs and component system outputs are mapped into component system inputs and coupled system outputs. The state of the coupled system is built up by the states of all component systems, system dynamic is built up by the dynamic of the components and the coupling scheme. A coupled system also has an input / output interface and is a system itself, it can appear as component in a bigger coupled system.

formal / *levels*	*differential equation*	*discrete time*	*discrete event*
coupled system	*DESN*	*DTSN*	*DEVN*
structured system			
input output system	*DESS*	*DTSS*	*DEVS*
function observation			
relation observation			
observation frame			

Figure 1.6 [Zeigler 84a]
Orthogonality of levels of system description and system formalisms

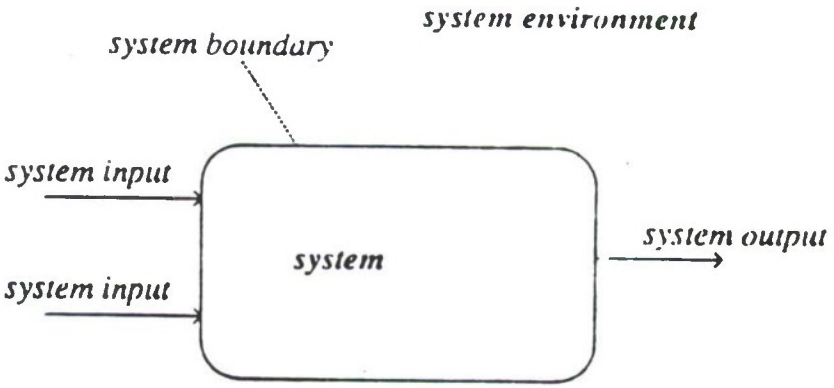


Figure 1.4 System concepts

$S = <T, X, Y, \Omega, Q, \Delta, \Lambda>$

with

T is a time base;

X is a *set of inputs*;

Y is a *set of outputs*;

Ω is the *set of admissible input segments* $\omega: <t_1, t_2> \rightarrow X$ over T and X and Ω is closed under concatenation as well as under left segmentation;

Q is a set, the *set of internal states*;

$\Delta : Q \times \Omega \rightarrow Q$ is the *global state transition function* which has to fulfill the following constraint (semigroup property):

$\forall\, \omega: <t_1, t_2> \rightarrow \Lambda \in \Omega, t \in <t_1, t_2> : \Delta(q, \omega) = \Delta(\Delta(q, \omega_{<t}), \omega_{>t})$;

$\Lambda : Q \times X \rightarrow Y$ is the *output function*.

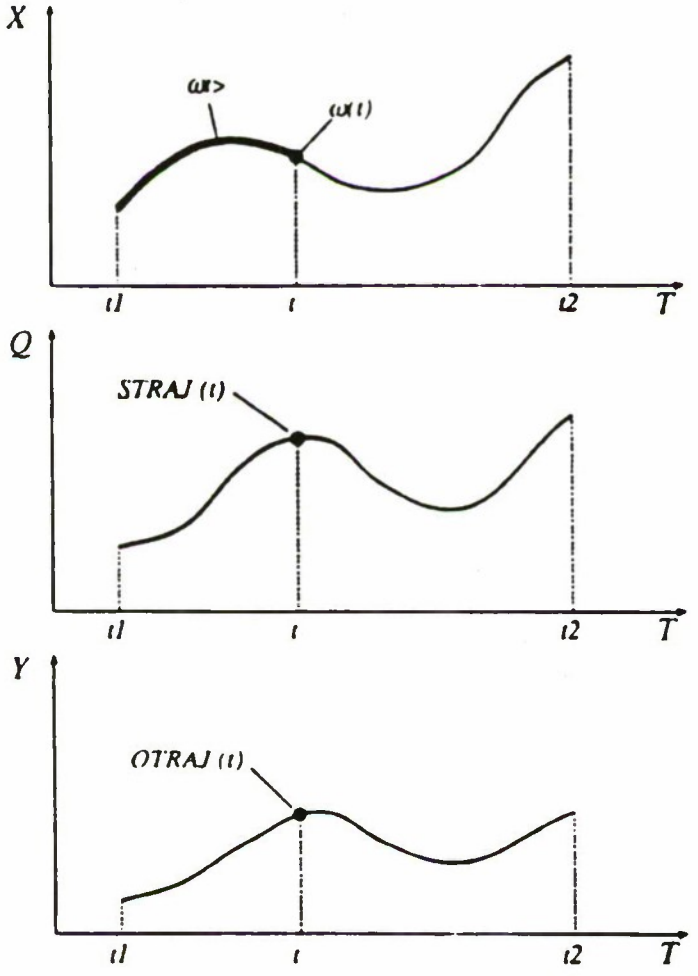


Figure 2.1 [Zeigler 76] ω, $STRAJ_{q,\omega}$ and $OTRAJ_{q,\omega}$ of the general dynamic system

	differential equation	*discrete time*	*discrete event*
time base T	*continuous reals*	*discrete integers*	*continuous reals*
basic sets X, Y, Q	*real vector space*	*arbitrary*	*arbitrary*
input segments	*piecewise continuous*	*sequences*	*discrete event segments*
state trajectory	*continuous*	*sequences*	*piecewise constant*
output trajectory	*continuous*	*sequences*	*discrete event segment*

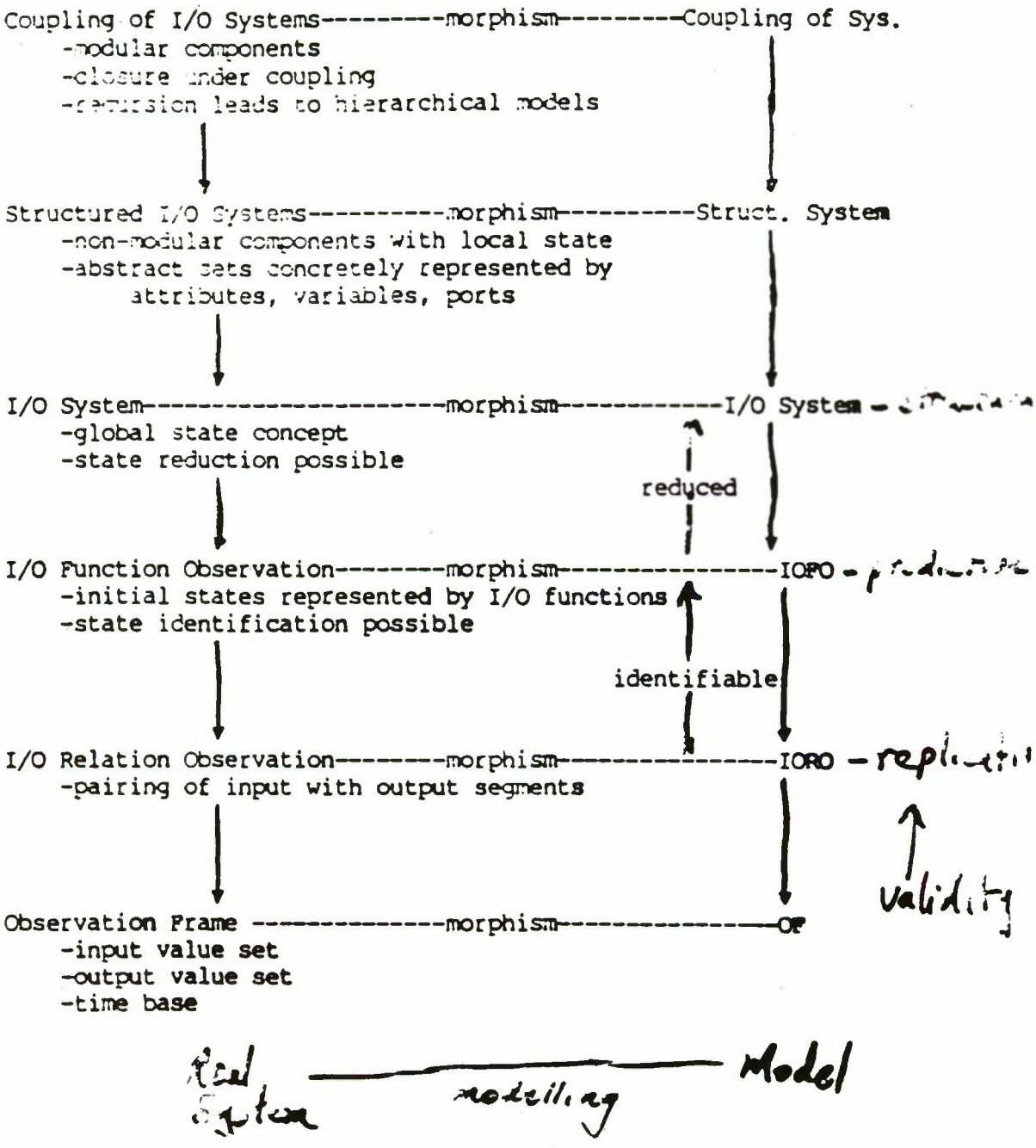
Figure 1.7 [Zeigler 84a]

Constraints imposed by system formalisms at the input / output system level

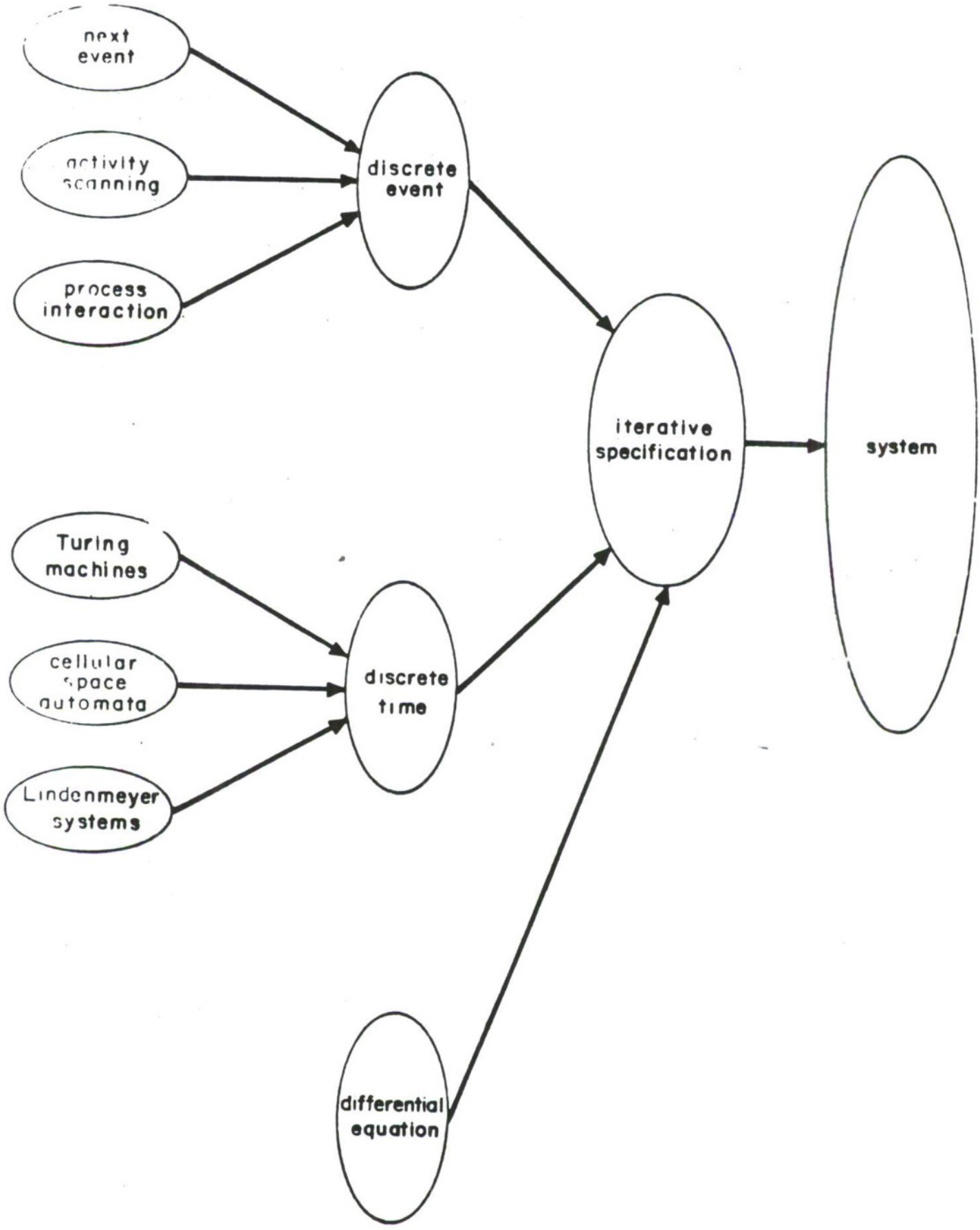
	differential equation	*discrete time*	*discrete event*
time base T	*continuous reals*	*discrete integers*	*continuous reals*
basic sets X, Y	*real vector space*	*arbitrary*	*arbitrary*
input segments	*piecewise continuous*	*sequences*	*discrete event segments*
components	*differential equ systems*	*discrete time systems*	*discrete event systems*
couplings	*no algebraic cycles*	*no algebraic cycles*	*no direct feedbacks*

Figure 1.8

Constraints imposed by system formalisms at the coupled system level



Levels of system specification
(epistemological levels)


next event
activity scanning
process interaction
discrete event
Turing machines
cellular space automata
Lindenmeyer systems
discrete time
differential equation
iterative specification
system

BACKGROUND LITERATURE

Object-Oriented Modeling and Discrete-Event Simulation[1]

BERNARD P. ZEIGLER

AI-Simulation Research Group
Department of Electrical and Computer Engineering
University of Arizona
Tucson, Arizona

1. Introduction 68
   1.1 Origins of Modeling and Simulation 69
   1.2 Definitions of Modeling and Simulation 70
   1.3 Activities Involved in Simulation Modeling 71
2. Discrete-Event Dynamic Systems 71
   2.1 The DEVS Approach 72
   2.2 DEVS in Relation to Other Formal Approaches 73
3. Brief Review of the DEVS Formalism 74
   3.1 Basic Models 76
   3.2 A Simple Processor: Example of a Basic Model 77
   3.3 Coupled Models 79
   3.4 A Multiserver: Example of a Coupled Model 80
   3.5 Expressing a Coupled Model as a Basic Model 81
   3.6 State Trajectories and Performance Indexes 82
   3.7 Hierarchical, Modular Models 84
   3.8 Simulation World Views 85
   3.9 Summary: DEVS Formalism and Its Properties 86
4. Object-Oriented System Concepts 86
   4.1 Objects 88
   4.2 Object Classes and Inheritance 89
5. The DEVS and Object-Oriented Paradigms 92
   5.1 Communication and Message Passing 93
   5.2 Implementation of DEVS in Object-Oriented Systems 93
   5.3 The Simulation Process 95
6. Concurrent Object-Oriented Systems 97
   6.1 Par-sets: A Parallel Generalization of List Structures 98
   6.2 Decentralization and Parallelism 99
   6.3 An Object-Oriented Computation Model 100

[1] Research supported by NASA-Ames Co-operative Agreement No. NCC 2-525, "A Simulation Environment for Laboratory Management by Robot Organization," and by McDonnell Douglas Space Systems Grant for Knowledge-Based Model Management. Portions of this chapter are adapted with permission from my book, *Object-Oriented Simulation of Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems* (Academic Press, Cambridge, Mass., 1990). Material was also adapted with permission from my article, "Simulation Methodology and Model Manipulation," in *Concise Encyclopedia of Information Processing Systems and Organizations*, edited by A. P. Sage (Pergamon Press PLC, 1990), pp. 414–19.

ADVANCES IN COMPUTERS, VOL. 33

Copyright © 1991 by Academic Press, Inc.
All rights of reproduction in any form reserved.
ISBN 0-12-012133-6

7. Distributed Simulation on Concurrent Object-Oriented Systems 101
7.1 Model Specification and Distributed Simulation 102
7.2 Hierarchical Simulation 103
7.3 Hierarchical Distributed Simulation Hybrids 104
7.4 The Need for Parallel Architectures to Design Parallel Systems 107
8. Conclusion 108
References 110

1. Introduction

The design of complex computer-based systems will depend ever more heavily on discrete-event simulation studies based on realistic models. By computer-based systems we mean systems relating to computer-integrated and flexible manufacturing, automation, and robotics as well as large communication networks, computerized process control, advanced computer architectures, and so on (Manivannan, 1989; Reddy *et al.*, 1986; Klahr, 1986; O'Keefe, 1986; Ehr and Wnuk, 1985). Object-oriented programming concepts are exerting a strong influence on the development of computer-based tools for complex-system design. More than a programming style, object-based concepts are coming to be regarded as a powerful paradigm for the modeling of large, complexly interacting systems such as flexible manufacturing systems, communication systems, and massively parallel computer architectures. Discrete-event simulation is very much allied to the object-oriented paradigm. (Indeed, the first object-oriented language, Simula67, was a discrete event simulation language.) Simulation provides the critical ability to study the dynamic (time-based) behavior of models that are defined with object-oriented means. This chapter reviews the fundamental concepts of object-oriented modeling and discrete-event simulation. Besides providing a historical context, it also looks ahead to some of the issues in need of much research, especially issues involving the use of parallel processing systems for simulation.

Current simulation practice is greatly limited in the size and realism that can be accommodated in models due to the extremely long times necessary to simulate them. Such long turnaround times severely impact the design cycle and lead to suboptimal designs. Ability to simulate large models in reasonable time is the motivation for seeking speed advantages offered by parallel processing systems. We shall review advances in distributed simulation, especially as they relate to concurrent object-oriented computing. We shall then discuss research areas that will contribute to attaining significantly higher levels of computer simulation capability including: (a) formalisms for discrete-event model specification that exploit the advantages of object-oriented

paradigms, (b) concurrent and parallel object-oriented computing systems to provide the platforms for distributed simulation, and (c) distributed simulation strategies to maximally exploit the parallelism in models and objects and fully utilize the capabilities of the underlying computing platform. Closing the loop, we shall show how distributed simulation is needed to design more powerful computer architectures—which in turn, will be needed to support distributed simulation of complex computer-based systems.

1.1 Origins of Modeling and Simulation

Modeling and simulation is an emerging field whose boundaries are not well defined. Model building can be traced back at least as far as the Newtonian era, but the tremendous impetus it received with the advent of the electronic computer is of course a relatively recent phenomenon. Moreover, there are at least two main sources of approach and technique—from physical science and operations research—that are still in the process of confluence and unification.

Physical scientists, especially in the applied and engineering branches, are faced with increasingly complex equations—combinations of general laws and empirical relations—for which analytic solutions are of limited use. In response, automatic solvers of differential equations were developed, whose operation was based on the integration capabilities of some particular natural medium. The early differential analyzers, developed in the 1920s by Bush and others, were based on mechanical motion. These were soon replaced by the faster and more reliable electronic analog computers, in which the integration is performed by capacitors and signals are normalized by high-grain amplifiers.

Analog computers saw heavy and significant use in the chemical and aerospace industries, among others, but limitations on problem size, stability, and accuracy of computation led to harnessing of the emerging electronic digital computers to achieve equivalent capabilities. The latter perform integration numerically, using principles that originated with long-known manual approximation methods. However, what gave digital computers eventual primacy was their information-processing abilities: simulation programming languages could be designed that would provide for convenient specification, processing, and manipulation of differential-equation models. Analog computation survives nowadays in the form of hybrid computers that couple analog integration with digital information-processing and control.

The second source of approach and technique lay in operations research with its desire to ameliorate industrial processing networks plagued by congestion, unproductive delays, and underutilization of resources. New concepts such as "event" and "activity" were developed which (in the

beginning) had little to do with the classical modeling concepts. An associated development was the incorporation of direct experimentation subject to chance outcomes within the computation, originally known as Monte Carlo methods. Tools were being developed for discrete-event modeling before there was adequate practical experience or theory to support them.

1.2 Definitions of Modeling and Simulation

As the field matures, the emphasis is shifting from simulation, as a set of computational techniques, to modeling, whether it be in a continuous or discrete form (or indeed, in forms that combine the two). Limitations are better appreciated, but so are the enormous potentials.

Definitions of modeling and simulation abound (Pritsker, 1979), partly reflecting the many origins of the area. Perhaps the most representative is that of Shannon (1975):

> Simulation is the process of designing a model of a real system and conducting experiments with this model with the purpose of either understanding the behavior of the system or of evaluating various strategies (within the limits imposed by a criterion or set of criteria) for the operation of the system.

Shannon emphasizes the experimental orientation of simulation techniques but widens the term *simulation* to include modeling and design activities that are not usually considered simulation-related. Other definitions try to characterize simulation narrowly to distinguish it from other computational techniques. One such definition is that adopted by Pritsker (1979):

> Simulation modelling assumes that we can describe a system in terms acceptable to a computing system. In this regard, a key concept is the system state description. If a system can be characterized by a set of variables, with each combination of variable values representing a unique state or condition of the system, then manipulation of the variable values simulates movements of the system by moving it from state to state in accordance with well defined operating rules.

It is characteristic of simulation tools that they facilitate a (hypothetical) description of the internal structure of a real system to the level of detail that the modeler perceives in reality. This power of representation distinguishes simulation from analytical techniques but also places a new burden on the modeler such as the choice of the level of detail compatible with the objectives of the modeling effort, the real-system data available, and the computational

and human resources at one's disposal. To write a detailed description of a system (i.e., a model) is one thing; to verify that it reflects one's intentions and then to validate it as a true description of the real system is another.

1.3 Activities Involved in Simulation Modeling

With the recognition that informal definitions can go only so far in conveying concepts and methods, a formal framework that founds a structure of definitions and theorems upon a set-theoretic basis has been developed (Zeigler, 1976). A comprehensive structuring of the activities involved in good practice delineates the following categories (Ören, 1987).

- Model generation and referencing: the generation of models (i.e., system descriptions) either by construction from scratch or by employing models retrieved from a model repository as components to be coupled together.
- Model processing: the manipulation of model texts (e.g., to produce documentation, to check consistency) and the generation of model behavior, of which simulation, restricted to mean computerized experimentation with models, is a predominant form.
- Behavior processing: the analysis and display of behavior in static (i.e., equilibrium), dynamic (i.e., concerning state trajectories), or structural (i.e., concerning changes in model structure) modes.
- Real-system experimentation: the gathering and organized storage of behavioral data from the real system or any of its components of interest.
- Model-quality assurance: the verification of the simulation program or device as correctly implementing the intended model, the validation of the model as an adequate representation of the real system, and the testing of other relations in which models participate.

This framework forms the basis for organization of articles on simulation methodology and model manipulation in Singh (1987). It also provides a useful perspective to keep in mind for the following discussion.

2. Discrete-Event Dynamic Systems

This chapter focuses on the discrete-event modeling and simulation approach that emerged from operations-research concerns. Discrete-event modeling is finding ever more application to analysis and design of complex manufacturing, communication, and computer systems among others. Long overdue recognition of the importance of the field emerged with the publication of a special issue on DEDS (discrete-event dynamic systems) edited by

Yu-chi Ho (1989). Powerful languages and workstations been developed for describing such models for computer simulation. (See Garzia *et al.*, 1986, for a general review.) Yet, general understanding of the nature of discrete-event systems *per se* (as distinct from their computer representations) is still in relative infancy compared to that of continuous systems.

Differential equations employed to describe continuous systems have a long history of development whose mathematical formalization came well before the advent of the computer. In contrast, discrete-event simulations were made possible by, and evolved with, the growing computational power of computers. The prime requirement for conducting such simulation was to be able to program a computer appropriately. Because they seemed not to be of immediate utility, computer-independent model description formalisms for discrete-event systems, which would parallel the differential equations for continuous systems, were late in coming. Yet, it is now being recognized that our understanding of complex systems may be greatly enhanced with such mathematically based formalisms.

Since the early 1970s work has been proceeding on a mathematical formalism for modeling discrete-event systems. One approach, inspired by the systems-theory concepts of Zadeh and Desoer (1963), Wymore (1967), Mesarovic and Takahara (1975), and Arbib and Padulo (1974), attempted to cast both continuous- and discrete-event models within a common systems-modeling framework. This approach was elaborated in a number of publications primarily summarized in books (Zeigler 1976, 1984) and is reviewed in (Zeigler 1985a). Systems-modeling concepts were an important facet in a movement to develop a methodology under which simulation could be performed in a more principled and secure manner. (See, for example Ören *et al*, 1984.) The recent advent of high-performance artificial-intelligence software and hardware has facilitated the transfer of this simulation methodology from research to practice (Elzas *et al.*, 1986).

2.1 The DEVS Approach

The Discrete Event System Specification (DEVS) formalism introduced by Zeigler (1976) provides a means of specifying a mathematical object called a system. Basically, a system has a time base, inputs, states, and outputs, and functions for determining next states and outputs given current states and inputs (Zeigler, 1984). Discrete-event systems represent certain constellations of such parameters just as continuous systems do. For example, the inputs in discrete-event systems occur at arbitrarily spaced moments, while those in continuous systems are piecewise continuous functions of time. The insight provided by the DEVS formalism is in the simple way that it characterizes how discrete-event simulation languages specify discrete-event system param-

eters. Having this abstraction, we can design new simulation languages with sound semantics that are easier to understand. Indeed, the DEVS-Scheme environment (Zeigler, 1990) is an implementation of the DEVS formalism in Scheme (a Lisp dialect) which enables the modeler to specify models directly in its terms.

DEVS-Scheme supports building models in a hierarchical, modular manner described previously. This is a systems-oriented approach not possible in popular commercial simulation languages such as Simscript, Simula, GASP, SLAM, and SIMAN (all of which are discrete-event–based) or CSMP and ACSL (which are for continuous models).

The DEVS formalism is more than just a means of constructing simulation models. It provides a formal representation of discrete-event systems capable of mathematical manipulation just as differential equations serve this role for continuous systems. Such manipulation includes behavioral analysis whereby properties of the behavior of a system are deduced by examining its structure. Although this is an area of intense investigation, such analysis is difficult—we return to this thought in a moment. Therefore, direct computer simulation will remain a primary means of generating and studying model behavior. However, other kinds of processing are equally important: mathematical representations may be compared, transformed into other forms, simplified, decomposed, and reconstituted in a great variety of ways (Ören, 1987; Mittleman and Praehofer, 1990).

2.2 DEVS in Relation to Other Formal Approaches

A number of other approaches to modeling discrete-event dynamic systems are brought together in the previously mentioned special issue (Ho, 1989). Many are algebraic or graphic in character and do not include the time element that DEVS inherits from its system theoretic origins. The most closely related formalisms are those emerging under the framework of generalized semi-Markov processes (GSMPs) in which we can include the stochastic generalizations of Petri Nets (Sanders, 1988; Meyer *et al.*, 1985). GSMPs, as formulated by Glynn (1989) and Cassandras and Strickland (1989), attempt to formalize discrete-event–simulation models as Markov processes with countable state sets that are amenable to mathematical analysis. The relationship between DEVS and GSMPs needs to be explored. However, DEVS appears to the more powerful formalism, trading mathematical tractability for expressive power (Zeigler, 1990).

A more history-oriented formalism for discrete-event–model specification has recently been proposed (Narain and Rothenberg, 1989). The availability of Prolog-like simulation engines makes such approaches feasible, but many practical and theoretical issues are raised.

3. Brief Review of the DEVS Formalism

Figure 1 depicts the conceptual framework underlying the DEVS formalism (Zeigler, 1976). The modeling and simulation enterprise concerns three basic objects:

- The *real system*, in existence or proposed, which is regarded as fundamentally a source of data.
- The model, which is a set of instructions for generating data comparable to that observable in the real system. The structure of the model is its set of instructions. The behavior of the model is the set of all possible data that can be generated by faithfully executing the model instructions.
- The *simulator*, which exercises the model's instructions to actually generate its behavior.

The basic objects are related by two relations:

- The *modeling relation*, linking real system and model, defines how well the model represents the system or entity being modeled. In general terms a model can be considered valid if the data generated by the model agree with the data produced by the real system in an experimental frame of interest.
- The *simulation relation*, linking model and simulator, represents how faithfully the simulator is able to carry out the instructions of the model.

There is a crucial element that has been brought into this picture: the experimental frame. This captures how the modeler's objectives impact model construction, experimentation, and validation. The interpretation of the experimental frame concept in simulation languages is still evolving.

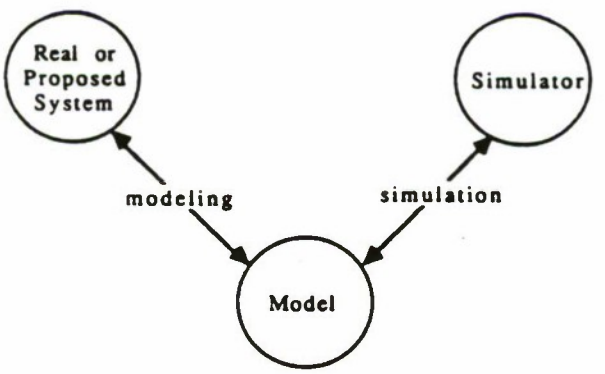
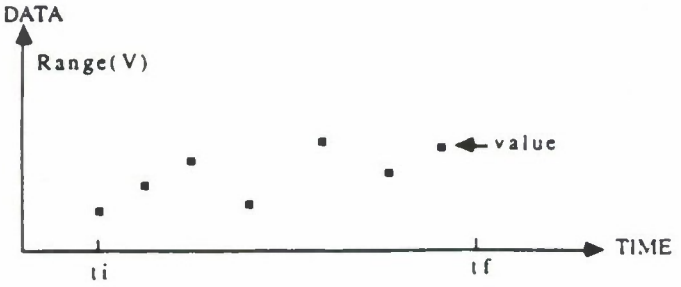


FIG. 1. Entities and relations in simulation.



Range(V): set of values that V can assume

ti: initial (starting) time

tf: final (terminating) time

FIG. 2. Generalized data segment produced by a system or model.

:ST (Ören, 1984) was the first conceptual language to sharply distinguish model and experiment specifications. SIMAN (Pedgen, 1983) was the first commercial language to incorporate a modicum of separation between model and experimental frame along the lines suggested by Zeigler (1976) and Ören and Zeigler (1979). The uniform treatment of experimental frame objects and model objects in DEVS-Scheme (Zeigler, 1990) implements a more recent formalization (Zeigler, 1984). In DEVS-Scheme, experimental frames are formulated as model objects in the same manner as the models of primary interest. In this way, model/experimental frame pairs form coupled model objects with the same properties as other objects of this kind.

The basic items of data produced by a system or model are *time segments*. These time segments are mappings from intervals defined over a specified time base to values in the ranges of one or more variables. The variables can be either observed or measured. An example of a data segment is shown in Fig. 2.

The structure of a model may be expressed in a mathematical language called a *formalism*. The discrete-event formalism focuses on the changes of variable values and generates time segments that are piecewise constant. Thus, an event is a change in a variable value that occurs instantaneously as shown in Fig. 3. We distinguish events, which are changes in value, from event-generating mechanisms. The latter are simulation constructs (often called "event routines") that at certain (scheduled) times determine whether an event actually occurs and what new values for variables are established. In essence the formalism defines how to generate new values for variables and the times the new values should be take effect. An important aspect of the formalism is that the time intervals between event occurrences are variable (in constrast to discrete-time models, where the time step is a fixed number).

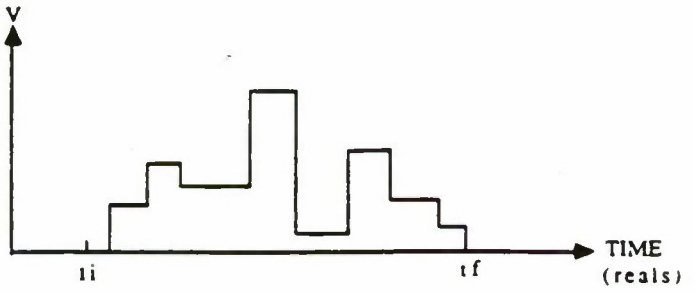


FIG. 3. Discrete-event time segment.

3.1 Basic Models

In the DEVS formalism, one must specify (1) basic models from which larger ones are built and (2) how these models are connected together in hierarchical fashion. In this formalism basic models are defined as follows.

A DEVS (discrete-event-system specification) is a structure:

$M = \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$.

X is the set of external (input) event types.

S is the sequential state set.

Y is the output set.

$\delta_{int}: S \to S$, the internal transition function.

$\delta_{ext}: Q \times S \to S$, the external transition function where Q is the total state set $= \{(s, e) \mid s \in S, 0 <= e <= ta(s)\}$.

$ta: S \to R^{+}_{0,\infty}$, the time-advance function.

$\lambda: S \to Y$, the output function.

To specify modular discrete-event models requires that we adopt a different view than that fostered by traditional simulation languages. As with modular specification in general, we must view a model as possessing input and output ports through which all interaction with the environment is mediated. In the discrete-event case, events determine values appearing on such ports. More specifically, when external events, arising outside the model, are received on its input ports, the model description must determine how it responds to them. Also, internal events arising within the model change its state, as well as manifesting themselves as events on the output ports to be transmitted to other model components.

A basic model contains the following information:

- The set of input ports through whicn external events are received.
- The set of output ports through which external events are sent.

- The set of state variables and parameters. Two state variables are usually present: *phase* and *sigma*. (In the absence of external events, the system stays in the current *phase* for the time given by *sigma*.)
- The time-advance function, which controls the timing of internal transitions. When the *sigma* state variable is present, this function just returns the value of *sigma*.
- The internal transition function, which specifies to which next state the system will transit after the time given by the time-advance function has elapsed.
- The external transition function, which specifies how the system changes state when an input is received. The effect is to place the system in a new *phase* and *sigma*, thus scheduling it for a next internal transition: the next state is computed on the basis of the present state, input port, value of the external event, and time that has elapsed in the current state.
- The output function, which generates an external output just before an internal transition takes place.

3.2 A Simple Processor: Example of a Basic Model

A pseudocode facilitates such model specification and its expression within DEVS-Scheme. Each input port requires specification of an external transition, in the form of a

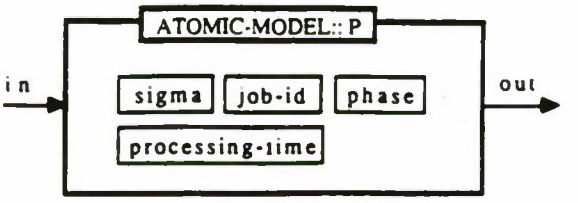
when receive x on input port p ...

phrase.

The internal transition function can be specified in the form of a process-like description with phases and their transitions. The output function uses phrases of the form

send y to output port p.

To illustrate, we consider a rather simplistic model of a processor that processes computer jobs or problems (Fig. 4). Expressed in pseudocode, it takes the form of a basic model called P. Basically, we represent only the time it takes to complete a job or solve a problem, not the detailed manner in which such processing is done. Thus, if the processor is idle, i.e., in *phase* 'passive, when a job arrives on the input port 'in, it stores the *job-id* (a distinct name for the job) and goes to work. This is achieved by the phrase *hold-in busy processing-time*, which sets the *phase* to 'busy and *sigma* (the time-left state variable) to *processing-time*. Such handling of incoming jobs is represented in the external transition function. Since this processor has no buffering capability, when a job arrives while the processor is busy, it simply ignores it.



ATOMIC-MODEL: P

state variables: sigma = inf
phase = passive
job-id = ()

parameters: processing-time = 5

external transition function:

```
case input-port
    in: case phase
            passive: store job-id
                     hold-in busy processing-time
            busy: continue
    else: error
```

internal transition function:

```
case phase
    busy: passive
    passive: (does not arise)
```

output function:

```
send job-id to port out
```

FIG. 4. A DEVS model of a simple processor.

This is achieved by the "continue" phrase which updates *sigma* to reflect passage of elasped time, but otherwise leaves the state unchanged.

When the processor has finished processing, it places the job identity on port 'out and returns to the 'passive *phase*. Sending of the job is done by the output function which is called just before the state transition function. The latter contains the phrase "passivate" which returns the model to the idle state in which the *phase* is 'passive and *sigma* is ∞.

Note that P has two state variables, *job-id* and *processing-time*, in addition to the standard ones, *sigma* and *phase*. Since *processing-time*, once initialized, does not change during the run, it is actually a *parameter* (fixed characteristic) of the model.

Simple as this processor is, we can combine it with other components

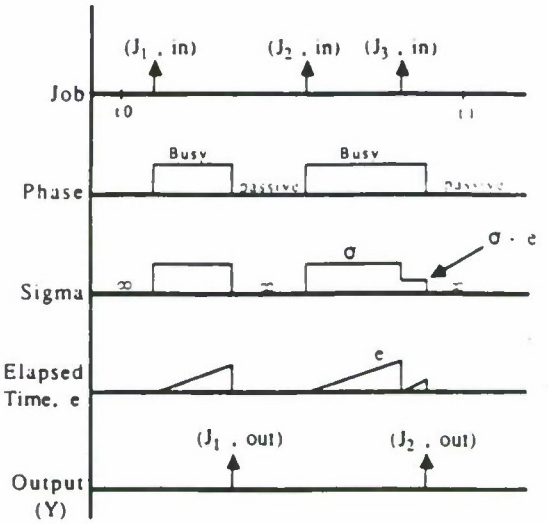


FIG. 5. Model-generated trajectories of simple processor.

create models of computer architectures that provide some insight into their performance. The basic model can also be refined to represent more complex aspects of computer operation.

It is important to note here that there is no way to generate an output directly from an external input event. An output can only occur just before an internal transition. To have an external event cause an output without delay, we have it "schedule" an internal state with a hold time of zero. The relationships between external transitions, internal transitions, and outputs are as shown in Fig. 5.

3.3 Coupled Models

Basic models may be coupled in the DEVS formalism to form a coupled model which is defined by the structure:

$$DN = \langle D, \{M_i\}, \{I_i\}, \{Z_{ij}\}, \text{select} \rangle$$

where D is a set of component names; for each i in D:

M_i is a component basic model
I_i is a set, the influencees of i; and for each j in I_i,
Z_{ij} is a function, the i-to-j output translation, and
select is a function, the tie-breaking selector.

A coupled model tells how to couple (connect) several component models together to form a new model. This latter model can itself be employed as a component in a larger coupled model, thus giving rise to hierarchical construction. A coupled model contains the following information:

- The set of components
- For each component, its influencees
- The set of input ports through which external events are received
- The set of output ports through which external events are sent
- The coupling specification consisting of:

  The *external input coupling* which connects the input ports of the coupled to model one or more of the input ports of the components. This directs inputs received by the coupled model to designated component models.

  The *external output coupling* which connects output ports of components to output ports of the coupled model. Thus, when an output is generated by a component, it may be sent to a designated output port of the coupled model and thus be transmitted externally.

  The *internal coupling* which connects output ports of components to input ports of other components. When an input is generated by a component, it may be sent to the input ports of designated components (in addition to being sent to an output port of the coupled model).

- The *select* function which embodies the rules employed to choose which of the imminent components (those having the minimum time of next event) is allowed to carry out its next event.

3.4 A Multiserver: Example of a Coupled Model

Various multiprocessing configurations can be modeled, each having a coordinator that sends problems (jobs) to some subordinate processors and receives solutions from them. As an example, in the *multiserver* architecture, the coordinator reroutes incoming problems to whichever processor is free at the time. Solved problems return to the coordinator and emerge from it. We study the throughput (number of jobs released per second) and turnaround-time (time taken for a job be processed) performance measures of such an architecture. (See Sauer and Chandy, 1980, for a discussion of performance evaluation.) To obtain these performance measures we couple an experimental frame component, EF, to the model. (See Zeigler (1990) for more details on experimental frames.)

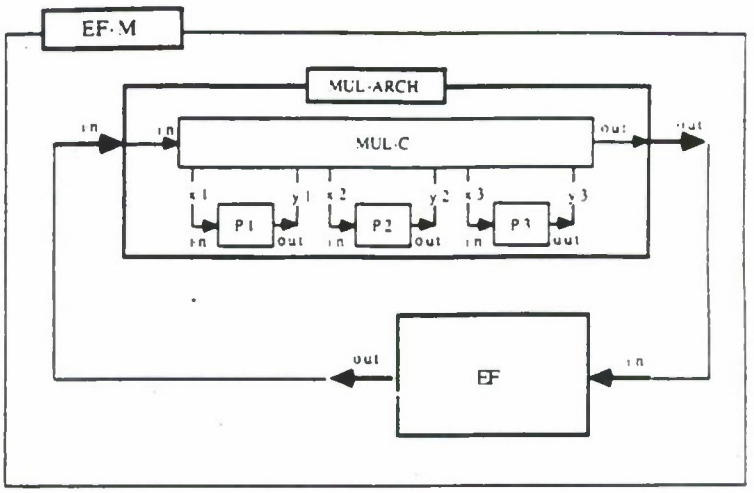


FIG. 6. The multiserver coupled-model architecture.

Figure 6 describes the coupled model, MUL-ARCH that implements the multiserver architecture. Note how the external-input coupling connects the 'in port of MUL-ARCH to the 'in port of the coordinator, MUL-C, while the external-output coupling similarly connects the 'out ports together. The internal coupling connects the sending and receiving ports of MUL-C to corresponding ports of the subordinate processors.

3.5 Expressing a Coupled Model as a Basic Model

A coupled model *DN* can be expressed as an equivalent basic model in the DEVS formalism (Zeigler, 1984). Such a basic model can itself be employed in a larger coupled model. This shows that the formalism is closed under coupling as required for hierarchical-model construction. Expressing a coupled model DN as an equivalent basic model captures the means by which the components interact to yield the overall behavior.

At any event time t, each component i is in a state s_i and has been there for an elapsed time e_i. The time advance in state s_i is $ta_i(s_i)$ so that component i is scheduled for an internal event at time $t + [ta_i(s_i) - e_i]$. The next event in the system will occur at a time that is the minimum of these scheduled times, namely, at time $t + \sigma$, where σ is the minimum of the residual times, $[ta_i(s_i) - e_i]$, over the components i in D. Of those components whose remaining times $[ta_i(s_i) - e_i]$ are equal to the minimum, we choose one using the tie-breaking *select* function. Let i^* be this selected, or *imminent*,

component. At time $t + \sigma$, just before i^* changes state, it computes its output $y^* = \lambda_{i^*}(s_{i^*})$. This output is sent to each of the influences of i^* in the form of a translated input: for influencee j, the input, $x_{i^*,j}$ is $Z_{i^*,j}(y^*)$. The elapsed time at any component i at time $t + \sigma$ is just $e_i' = e_i + \sigma$. An influencee, j, responds to the external event generated by i^* by applying its external transition function, to obtain the next state $s_j' = \delta_{ext}(s_j, e_j', x_{i^*,j})$ and to reset its elapsed time to 0. Other components not in the influencee set are unaffected by the activation of i^* except that their elapsed time clock is incremented by σ as just described. Finally, the imminent component i^* executes its internal transition by going to state $s_{i^*}' = \delta_{int}(s_{i^*})$ and resetting its elapsed time to 0.

Let the state of the basic DEVS model M, representing the overall system, be the vector of states $s = \langle (s_i, e_i) \rangle$ of the components. The preceding describes how M's time advance and internal transition functions work. Namely, the time advance in state s, $ta(s) = \sigma$, the smallest of the residual times of each of the components. At the next event, M's internal transition function transforms the given state to a new vector $\langle (s_i', e_i') \rangle$ computed according to the preceding recipe. We can similarly follow the effect of an external input event's arrival to some of the components and thereby derive the external transition function of the basic model.

3.6 State Trajectories and Performance Indexes

Let us trace a typical state trajectory to illustrate the operation of coupled models in general and the multiserver architecture in Fig. 6 in particular. We start in an initial state in which the multiserver coordinator and all subordinate processors are idle. The experimental frame will send jobs to arrive on port 'in of MUL-ARCH. Figure 7 shows how we can represent the time behavior for a coupled model. The incoming job stream is represented by the segment of external input events shown on the top horizontal axis. Each of the three processors is assigned its own time axis.

Following the course of the first job arrival, JI, on port 'in of MUL-ARCH, the external input coupling scheme will send JI to port 'in of the coordinator, MUL-C (not shown in Fig. 7). Having received JI and being passive, MUL-C goes into state BUSY (dictated by its external transition function). After waiting there for a very short time (actually zero), the coordinator puts JI on port 'x1 (as dictated by the output function) and immediately returns to the passive phase (due to the internal transition function).

The internal coupling of MUL-ARCH then causes J1 to be appear on port 'in of processor P1. Since the latter is idle, it accepts the job and enters the BUSY phase for a time given by its processing-time parameter. (Recall the

time by $p/3$ so that the throughput is $3/p$. Since the processors are always kept busy, there is no way to produce a higher rate of job completions. Clearly, each job still takes p units of time to be processed, so that the average turnaround time is p.

3.7 Hierarchical, Modular Models

The fact that the DEVS formalism is closed under coupling leads to a very important structuring property: the ability to construct hierarchical models (Simon, 1969) in a step-by-step manner.

A *hierarchical* model is inductively defined:

- An atomic model is a hierarchical model.
- A coupled model whose components are hierarchical models is a hierarchical model.
- Nothing else is a hierarchical model. The only way to get hierarchical models is by following the inductive process defined in the first two clauses.

The structure of a hierarchical model is exhibited in a *composition tree* such as that in Fig. 8 for the multiserver architecture. The components of the outermost coupled model, or root model, are shown as its children. (MUL-ARCH and EF are children of EF-M.) A child, which is also a coupled model, has its components descending from it as children. Children that are atomic models become leaves in the tree. The coupling specification needed to construct a coupled model is attached to a vertical line descending from the parent to its children. In other words, the coupling specification is asso-

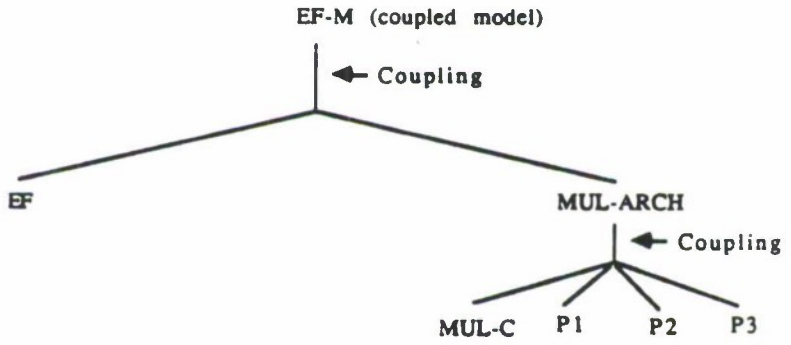


FIG. 8. Composition tree for multiserver architecture.

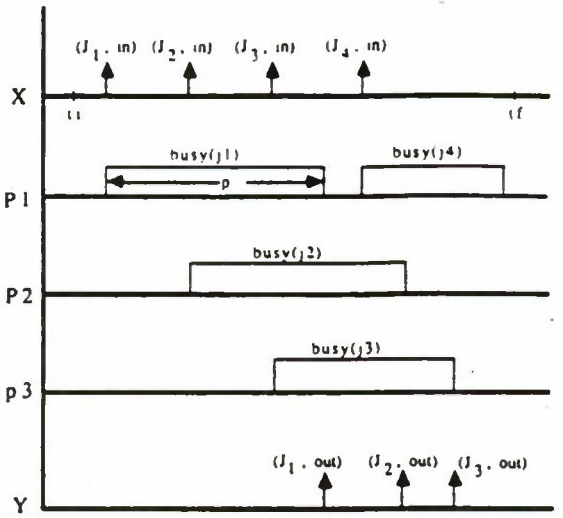


FIG. 7. Sketch of trajectory for multiserver architecture.

description of the simple processor P of which P1 is an isomorphic copy.) Let p represent the value of the processing time. For simplicity in the sequel, we shall assume that p is a constant and the same for all processors. After time p has elasped, P1 will place J1 on port 'out. The external output coupling now determines that J1 appears on port 'out of MUL-ARCH and leaves the architecture as a processed job as illustrated in Fig. 7.

Now let a second job, J2, arrive T time units after J1's arrival. If T is bigger than p, then P1 will be passive by the time J2 arrives and will start processing it. However, if T is smaller than p, then P1 will be busy when J2 arrives. Rather than losing J2 as was the case for the simple processor, here the multiserver coordinator comes into play. Knowing that P1 is busy, MUL-C sends J2 to the next free processor, which happens to be P2. More truthfully, MUL-C places J2 on its output port 'x2, which is coupled by the internal coupling of MUL-ARCH to P2's input port 'in. J2 will be sent out of MUL-ARCH p units later in a manner similar to J1's processing.

A third job, J3, arriving while both P1 and P2 are busy, will be sent to P3. However, a fourth job that arrives while all processors are busy will be lost. As illustrated in Fig. 7, if the job interarrival time, T, is a constant, equal to $p/3$, then the fourth and subsequent jobs arrive just after a (exactly one) processor has finished its work. The figure makes clear that this is an arrival pattern in which processors are always kept busy. The emerging jobs are separated in

ciated with a decomposition of the parent into its children. Thus, a composition tree represents the information needed to construct a particular hierarchical model.

3.8 Simulation World Views

The three standard so-called world views for discrete-event model specification—event scheduling, activity scanning, and process interaction (Franta, 1977; Hooper, 1986)—provide alternative means of model representation. Each may have conceptual and efficiency advantages for representing systems of particular kinds (Zeigler, 1976, 1984; Overstreet and Nance 1986; Balci, 1988).

Event-scheduling languages provide means for scheduling events. The underlying world view of these languages may be characterized as event-driven. That is, it is event routines that make things happen to objects—but the events need not be directly linked to actions of objects. A simulation of chemical reactions at the molecular level provides an example. Here vast numbers of collisions are the events that trigger changes in states of colliding molecules. In writing such a model, one concentrates on how collisions schedule other collisions to occur, thus advancing the model through time.

Activity-scanning languages view a system as consisting of a concurrent set of activities, each with its own conditions for activation and its own working time. An example is that of a shipping port in which boats and machines are involved in a number of activities including docking and undocking, loading and unloading, and entering and leaving the port. Events in such models are directly associated with objects. Indeed, the initiation and termination of activities would be characterized by events in the event-scheduling approach.

Process-interaction languages view a system as composed of processes, which are sequences, or cycles, of activities carried out by objects. This view would treat the shipping port as containing a collection of ships that each engage in a process or sequence of activities such as enter the port, dock, unload the cargo, load new cargo, and leave the port.

These traditional worldviews are nonmodular in that they allow components of a model unrestricted access to each other's states. The basic model specification of hierarchical, modular DEVS is most closely identifiable with the process formalism but endows it with the modularity property that conventional process-based languages do not support. Cota and Sargent (1990) introduce an extension to the traditional process-interaction world view called "new world view" and show that it is structurally equivalent to the hierarchical, modular DEVS. Since Zeigler (1984) showed that all world views are behaviorally equivalent, the advantage of the new world view does not lie in its ability to express new model behaviors. Rather, the hierarchical, modular

properties support independent model-component definition and make it possible to reliably construct models of large, complex systems. Moreover, the new world view accords better with that adopted by researchers in distributed simulation, which is discussed later.

Another form of model specification equivalent to the new world view is the rule-based approach supported in *class forward models* in DEVS-Scheme (Zeigler, 1990). This is reminiscent of conventional activity scanning—rules and activities superficially look very much alike. However, the forward-models paradigm affords the modularity not supported in the traditional activity-scanning world view. Forward models also support the class and inheritance features of the object-oriented paradigm to be discussed next.

3.9 Summary: DEVS Formalism and Its Properties

The formal properties of the DEVS formalism and their importance for model-based simulation are summarized as:

- *Modularity:* Model specifications are self-contained and have input and output ports through which all interaction with the external world must take place. Ports provide a level of delayed binding that needs to be resolved only when models are coupled together.
- *Closure under coupling:* Models may be connected together by coupling of input and output ports to create larger coupled models having the same interface properties as the components.
- *Hierarchical construction* follows as a consequence of modularity and closure under coupling. Successively more complex models can be built by using as building blocks the coupled models already constructed.
- *Stand-alone and bottom-up testability:* Due to input/output modularity, models are independently verifiable at every stage of hierarchical construction. This fosters secure and incremental bottom-up synthesis of complex models.
- *Experimental frame/model separation:* Experimental frames are independently realized as models of special kinds: generators, transducers, acceptors. Having input/output ports, they can be coupled to models to which they are applicable.

4. Object-Oriented System Concepts

Object-oriented programming, in which software is structured as a collection of interacting objects, has emerged as the common basis for a host of diverse applications including: (1) computer simulation of discrete-event systems, (2) software-engineering techniques such as abstract data types, encap-

sulation, information hiding developed to enhance modularity, correctness, testability, extensibility, reusability, and maintainability, (3) operating-system mechanisms to protect resources and security, and (4) representation schemes in artifical intelligence, such as rules, frames, demons, and blackboard coupled specialists, that express knowledge as aggregations of modular chunks that can communicate when needed (Hayes, 1981).

These varied applications, and the remarkable fact of the convergence to the object concept, lend strong support to the contention that object-oriented programming will become universally accepted as a new standard for software development in general. A variety of *object-oriented programming languages* have been developed: Smalltalk (Goldberg and David, 1983), LOOPS (Bobrow and Stefik, 1983), Flavors (Weinreb *et al.*, 1983), and Eifel (Meyer, 1988). Already object-oriented versions of widely used programming languages have emerged and are finding commercial acceptance: C++ (Stroustrup, 1986), Classic-Ada (Bach, 1989), and CLOS (Common Lisp Object System; Keene, 1988).[1]

Interestingly, the connection between discrete-event simulation and object-oriented programming is quite old. The discrete-event–simulation language Simula (Dahl and Nygaard, 1966) introduced class inheritance and association of both procedures and data structures with class instances. Simula allowed not only static entities to be represented as objects but also dynamic entities such as events and processes. However, in conventional simulation, a distinct paradigm for decomposing problems into object classes and associated methods did not emerge. Conventional simulation languages offer no special support for such a programming style. As a result, new languages have emerged for object-oriented simulation (Klahr, 1986; Middleton and Zancanato, 1986; Ruiz-Mier and Talavage, 1989; Bryan, 1989; Lomow and Baezner, 1989). However, more than just a simulation tool, the object-oriented perspective also lends itself to the higher level of systems design in which complex processes and systems may be usefully represented.

In object-oriented paradigms, and object is a conglomerate of data structures and associated operations, usually representing a real-world counterpart. *Objects* are usually given generic descriptions so that *classes* of objects are defined and individual instances of such classes may be generated at will. Classes of objects form a *taxonomical hierarchy* in which they are arranged according to their degree of generality. Special kinds of objects have *slots* for both attributes and methods that are unique as well as those that are *inherited* from more general classes (Adelsberger 1986). *Methods* (procedures)

[1] Languages such as Ada that feature abstract data typing (encapsulation, information hiding, and abstraction) but not inheritance and dynamic binding are sometimes referred to as "object-based" as opposed to "object-oriented." Classic-Ada is a preprocessor that adds the latter facilities to Ada.

can perform operations on the global object state (the ensemble of its slots) and invoke each other in a manner called *message passing*. The massage-passing paradigm differs from subroutine invocation in allowing a much greater degree of discretion to the receiver in interpreting a message. A subroutine call is a command sent to by an onmiscient master to a completely subservient slave; a message is a polite request by one peer to another to perform an action that the latter may or may not be able, or choose, to do. Slots can have so-called *active values*. These are procedures that are triggered when a slot is accessed or modified. Active values can be likened to *demons* that watch over slots to determine their values and propagate information to other slots.

Conventional software systems tend to consist of collections of subprograms based on functional decomposition techniques that concentrate on algorithmic abstractions. In conventional programming languages such as LISP or Pascal, we tend to write programs around a main routine that calls on other routines to work at appropriate times. Decision-making power is concentrated in the main routine. The other routines play supporting roles, coming alive only when the flow of control passes through them. In contrast, the object-oriented paradigm encourages a much more decentralized style of decision making by creating objects whose existence may continue throughout the life of the program. We can make such objects act as experts in their own task assignments by providing them with the appropriate knowledge. Such distribution and localization of knowledge has important implications for parallel execution, as we shall see.

4.1 Objects

As just indicated, and object-oriented system contains components called *objects*. Each object has its own variables and procedures to manipulate these variables called *methods*. Only the methods owned by the object can access and change the values of its variables. The values originally assigned to variables of an object will persist indefinitely throughout its lifetime—unless changed by some method. They will stay this way until a subsequent change is brought about by some—the same or another—method. Thus, the variables collectively constitute the state of the object. In these terms, only the methods of an object can alter its state.

Objects can communicate with each other, and with higher levels of control, to cause changes in their states, by a process called message passing. The general form of a message is:

send
to object: *O*
apply method: *m*
..ith arguments: *a*1, ... ,*an*

specialized methods are compatible (same input and output) as the general ones. Thus, extensibility and ease of system evolution are inherent in the object-oriented approach.

The more general form of organization in which classes may inherit from several classes, called *multiple inheritance*, provides additional flexibility in certain situations but is harder to keep track of. The recent standardization of object-oriented approaches in CLOS takes things one step further. It supports a very flexible scheme of associating a generic method call with a particular method based on the classes of its arguments. Ordinary inheritance is then the simple case in which only one argument determines how a generic method is to be interpreted.

While the outlines of the object-oriented paradigm have achieved a certain degree of general acceptance, many variations on the main themes have been proposed. CLOS is intended as an environment to explore existing alternatives and to introduce new ones. It provides metalevel object-definition capabilities for this purpose. Theoretical and practical questions are open. For example: what is the best class-inheritance scheme for a given set of classes? Indeed, even the question of how to characterize optimality, e.g., minimal redundancy, is open (Mittelmann and Praehoffer, 1990).

Some of the major features of the object-oriented paradigm are summarized as follows:

- class definition:
  - —objects in a class have same structure (slots) and behavior (methods)
  - —means for specifying templates for object construction
  - —membership predicates: testing for membership of an object in a class
  - —class taxonomies, object hierarchies
- instantiation protocols:
  - —object creation, replication, destruction
  - —associated equality predicates: testing for equality of objects
  - —more generally, pattern matching: testing whether objects match patterns
- existence modes:
  - —everything an object (or only user-defined objects on top of preobjects) persistence: does not disappear between invocations
  - —passive/active status (whether object is processing or not)

- inheritance mechanisms (assignment of structure and behavior):

  —compile time:
  —run time: semantic nets
  —generic function approach of CLOS

- message-passing protocols:

  —dynamic binding
  —broadcasting
  —multicasting
  —fixed directed graph
  —reconfigurability

- multiprocessor architectures:

  —assignment of objects to processors (fixed, dynamic)
  —distributed control (each object stores its own methods)
  —centralization (CLOS generic function processor)

5. The DEVS and Object-Oriented Paradigms

Model specifications arising from mathematical system theory (Ören, 1984; Zeigler, 1976, 1984; Futo, 1985) bear a resemblance to concepts of object-oriented modeling. Both objects and system models share a concept of *internal state*. However, *modular-system models* operate on a *time base* while object behavior is not indexed by time. Moreover, modular-system models have recognized input and output ports through which all interaction with the environment occurs. On the other hand, while objects pass messages to each other, the sender must have an explicitly given set of receivers called *acquaintances*. These objects must exist for the transmission to be well defined (Agha, 1987). Thus, an object cannot be studied in isolation apart from its acquaintances. However, the coupling concept in modular systems provides a level of *delayed binding*—a system model places a value on one of its ports, the actual destination of this output is not determined until the model becomes a component in a larger system and a coupling scheme is specified. Clearly, a system model may place an output on a port that has no receiver coupled to it. It can therefore (a) be developed and tested as a stand-alone unit, (b) be placed in a model base and reloaded at will (an object must have its acquaintances

This represents a message sent to object *O* telling it to apply its method named *m* with the argument values *a*1, ... ,*an*. Carrying out the orders of this message may result in the method changing the state of the object and/or producing an output in response to the message.

One of the most useful concepts afforded by the object-oriented paradigm is that different objects can have variables or methods having the same name. Such methods may exhibit a basic similarity in purpose despite a difference in the detailed manner in which this purpose is achieved. The user of a method relies only on the fact that any object that has that method can be expected to exhibit a certain behavior and leaves the implementation of this behavior to the designer of the object. The process of enclosing data and methods within an object is called *encapsulation*. Shielding the user of an object from its internal structure is called *information hiding*, and revealing only its behavior is *abstraction*. We shall see that such abstraction has an important consequence, called *extensibility*, the ability to extend a software system by adding in new definitions without modifying earlier ones.

4.2 Object Classes and Inheritance

In object-oriented systems, objects are usually not defined individually. Instead, a class definition provides a template for generating any number of instances, each one an identical copy of a basic prototype. The basic form for such a class definition is:

Define class: name (name to be used for the class)
class variables (variables describing the class per se)
instance variables (variables owned by each instance)
methods (methods owned by each instance)
constructor (procedure for creating instances)
destructor (method for destroying instance)
inheritance (other classes from which to inherit definitions)

Often objects can be organized into a family of homogeneous classes, which are mutually distinct, but do have certain fundamental properties in common. The object-oriented paradigm provides a natural way to exploit such situations to afford an economy of definition as well as the extensibility just referred to. A class definition can specify from which classes the new class will inherit, i.e., automatically obtain all their definitions. This saves having to copy or rewrite such common variables and methods and helps maintain consistency in definitions when code modifications are made.

In the most straightforward case, a class inherits only from only one other class, which is called its *parent*. In this case, the classes form a tree structure, called a *specialization hierarchy under a root class*. The root class

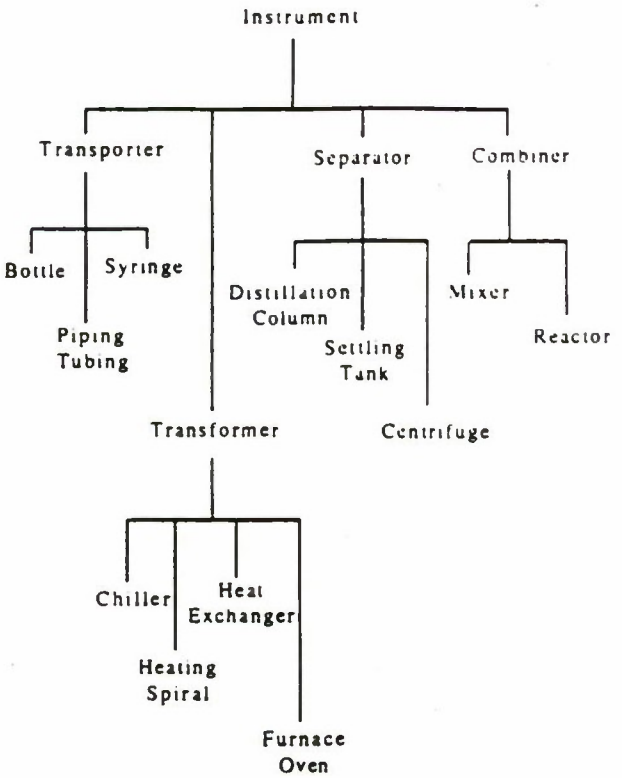


FIG. 9. Class specification hierarchy for laboratory instruments.

is the most general class. Its children inherit all its features and are more specialized, in that they may have additional features as well. Their children may be even more specialized, inheriting from the parents (and hence from the grandparent). An example is shown in Fig. 9, where laboratory instruments are organized into a specialization hierarchy.

The root class generally provides general definitions for methods of general utility. The more specialized classes may override such general definitions by providing methods of their own with the same name. The term *polymorphism* is used to refer to such multivalued definition: object-oriented systems obviously must interpret the meaning of a method name in its proper context, a process called *dynamic binding*. Polymorphism and dynamic binding are important features that distinguish message passing from ordinary procedure calls.

The system may evolve by adding more and more specialized classes, while the higher-level procedures need not change so long as the newly introduced

specified), and (c) be reused in any applications context in which its behavior is appropriate and coupling to other components makes sense.

5.1 Communication and Message Passing

Communication in a modular-systems paradigm involves a more complex *communication protocol* than that of a conventional object-oriented one. In conventional object paradigms, a component A can send a message to a component B, expecting a value to be returned immediately as in a subroutine call. In coupled-system models (as well as in concurrent-object-oriented systems, which are soon to be discussed), objects are simultaneously active and message transmission is more complex: an arbitrary time will elapse between A's sending to B and B's responding to A. During this interval, both A and B may also be engaging in their individual activities. Indeed, B may be busy when A's message arrives and may ignore it or buffer it for later examination. A concurrent-object-oriented environment must clearly specify the details of the message-handling protocols so that the user can design object behaviors accordingly.

The modular-systems paradigm on the other hand does not impose a specific message-passing protocol. Perhaps, it would be fairer to say that it imposes only a minimal one. The modeler is therefore completely free to specify, as an integral part of his or her models, how communication is handled. Note that information flow is only one of many interactions that may be modeled; for example, material flow may be present as well. Finally, since system models operate on a time base, times involved in communication and other interaction are explicitly modeled. In contrast, in concurrent object-oriented systems, the underlying computational model must shield the users, who do not concern themselves with time explicitly, from indeterminacies that could arise due to transmission delays, buffering, etc.

5.2 Implementation of DEVS in Object-Oriented Systems

Although object-oriented systems feature a limited form of communication protocol in relation to modular systems, they can serve well as a basis to implement discrete-event and other system-modeling formalisms (Zeigler, 1990; Thomasma and Ulgen, 1988; Kim, 1988; Praehofer and Mittlemann, 1990; Ehr and Wnuk, 1985). In this section, we briefly review how the DEVS formalism may be implemented using the primitives provided by object-oriented systems.

The implementation of the hierarchical, modular DEVS formalism over an object-oriented substrate is based on the abstract simulator concepts (Concepcion and Zeigler, 1988). Since such a scheme is naturally implemented

by multiprocessor architectures, models developed in this form are readily transportable to the distributed simulation systems discussed later.

Classes *models* and *processors* provide the basic constructs needed for modeling and simulation. *Models* is further specialized into the major classes *atomic-models* and *coupled-models*, which in turn are specialized into more specific clases, a process that may be continued indefinitely as the user builds up a family of models for a specific application domain. Class *processors*, on the other hand, has three specializations: *simulators*, *coordinators*, and *root-coordinators*, which serve to handle all the simulation needs.

As class atomic-models realizes the DEVS basic model formalism, it has variables corresponding to each of the parts of this formalism. For example atomic-models has instance variables *int-transfn*, *ext-transfn*, *outputfn*, and *time-advancefn*, which specify a model's internal transition function, external transition function, output function, and time-advance function, respectively. These functions are applied to the state of the model by the methods *int-transition*, *ext-transition*, *output*?, and *time-advance*? In DEVS-Scheme classes *forward-models* and *table-models* are examples of specialized classes of atomic-models and provide for rule-based and table-based model specification, respectively.

Coupled-models is the major class that embodies the hierarchical model composition constructs of the DEVS formalism. It can be further specialized to provide subclasses for specific kinds of coupling schemes. A coupled model is defined by specifying its component models, called its *children* and the coupling relations that establish the desired communication links. Accordingly, any specialization of coupled-models is expected to supply the methods:

- *get-children*, which returns the list of components
- *get-influencees*, which determines those siblings to which the output of the imminent component will be sent
- *get-receivers*, which determines which subcomponents will receive an external-event input to the parent coupled model
- *translate*, which provides port-to-port translation.

For example, the class *kernel-models* is a subclass of coupled-models whose specializations represent various forms of network interconnects. The *children*, *influencees*, and *receivers* are uniquely determined by the particular specialization of kernel-models. For example, in *broadcast-models*, all subcomponents of a model communicate directly with each other and with the outside world. *Controlled-models* provides a means for representing centrally controlled systems. *Hypercube-models* and *cellular-models* provide for coupling of components via a geometrically based neighborhood relation.

5.3 The Simulation Process

The *simulators*, *coordinators*, and *root-coordinators* specialization classes of processors carry out the simulation of DEVS models by implementing the abstract simulator principles developed as part of the DEVS theory (Zeigler, 1984; Concepcion and Zeigler, 1988). Simulators and coordinators are assigned to handle atomic-models and coupled-models in a one-to-one manner, respectively (see Fig. 10). A root coordinator manages the overall simulation and is linked to the coordinator of the outermost coupled model. Simulation proceeds by means of messages passed among the processors that carry information concerning internal and external events as well as data need for synchronization. Messages have fields for *source* of origination, *time* (carrying local or global time stamps, depending on the use), and *content* consisting of a *port* designation and a *value*, which are both determined by atomic-model output functions. There are four types of messages: *, *x*, *y*, and *done*.

A processor receives and sends several types of messages. An *x-message* represents the arrival of an external event to a processor's assigned model; it bears the global model time and comes from its parent. A coordinator transmits this message to the processors of its assigned model's receivers,

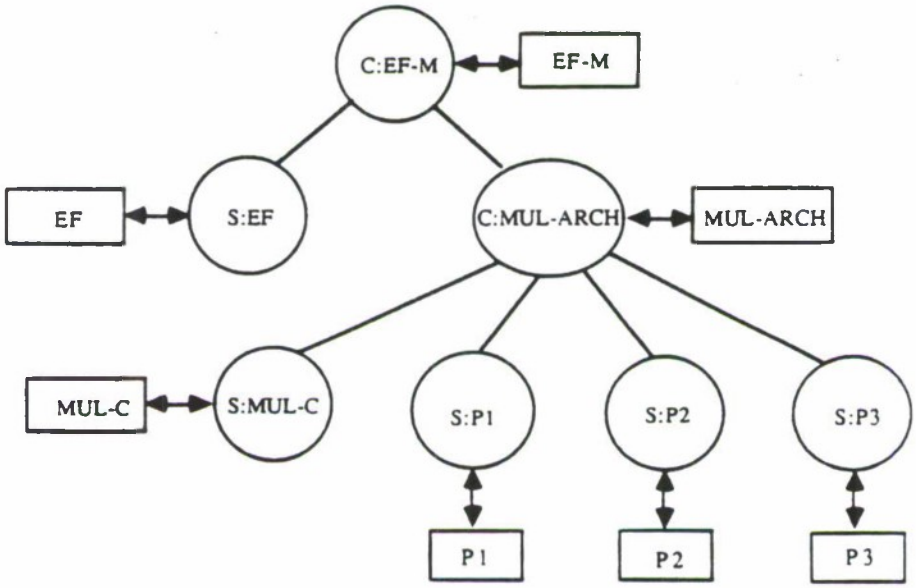


FIG. 10. The abstract simulator structure formed by assigning processors to the model components.

using its *get-receivers* and *translate* methods. When a simulator receives an *x-message*, it calls the external transition function of its assigned model (using the *ext-transition* method) and then responds with a *done-message*. The latter indicates to the parent that the state transition has been carried out and carries with it the model time at which the next internal event of its assigned model is scheduled (obtained by calling the *time-advance*? method).

A **-message* arriving at a processor indicates that the next internal event is to be carried out within its scope. Thus, a coordinator responds to a **-message* by transmiting it to its imminent child, the child with minimum *time-of-next-event* (or selected by tie-breaking rules embodied in the *selectfn*, if more than one has the minimum *time of next event*).

A simulator processes the **-message* by calling the internal transition function of its assigned model (which is the imminent atomic model) and responding with a *y-message* followed by a *done-message*. The former message carries as content the *port* and *value* obtained by calling the output function of its atomic model. The latter *done-message* indicates that the state transition has been carried out and provides the new *time of next event*.

When a coordinator receives a *y-message* from its imminent child, it consults the external output coupling scheme to see whether it should be transmitted to its parent, and its internal coupling scheme to obtain the children and their respective input ports to which the message should be sent. This processing uses the *get-influencees* and *translate* methods of its coupled model.

When a coordinator has received the *done-messages* from all the inflencees (in the ascending *y-message* case) or receivers (in the descending *x-message* case), it computes the minimum of its *tN-children* (maintained list of times of next event) and determines its new immient child for use upon receiving the next **-message*. Also it sends this new minimum as the time of its own next internal event in a *done-message* to its parent.

Simulation is begun by initializing the states of the atomic-models, thereby also determining each one's *time-of-next-event*. These times are propagated upward by *done-messages* and thus set up a path of imminent subcomponents from the outermost coupled model to an innermost atomic model. When the root coordinator receives a *done-message* from its child (the coordinator of the outermost coupled-model), it returns a **-message* to it bearing its *time of next event*. This starts the simulation, since the **-message* will be transmitted down the imminent path to the imminent simulator. There will result an upward wave of *y-messages*, a downward wave of *x-messages*, and an upward wave of *done-messages*, the last of which, transmitted to the root coordinator, initiates the next round of simulation (processing of the next internal event).

The simulation process has been shown to be correct by Zeigler (1984), i.e., it will generate the behavior of the equivalent basic model associated with a coupled model (Section 3.5).

Such an implementation provides for message tracing as well as standard execution. In the *pause* mode, the simulation pauses with each message receipt and the contents of the received message are displayed in the window of the affected component. In the *run* mode, the simulation advances without interruption and only the states of the atomic-models are displayed in their respective windows. In *pause* mode, a simulation run can be terminated anywhere during the root coordinator's cycle. This leaves the model in a partial state, which may well manifest the source of an error. In *run* mode, however, a request for termination can be issued at any time, but the simulation will stop only when the current cycle is complete. This leaves the model in a completed state from which the simulation can be validly continued. The run can be restarted from the resulting state after any desired modification, whether to the model or to the state.

6. Concurrent Object-Oriented Systems

Discrete-event simulations are acknowledged to be among the most computationally expensive tasks. In many cases the sequential execution of large-scale simulations make take hours or days of processor time, and stochastic models require many runs to determine output distributions and confidence levels. One way to employ parallel processing to speed up such simulations is to map the object-oriented implementation of a discrete-event model directly onto a parallel processing system. Such an approach seeks to exploit the parallelism visible at the object level but not at the DEVS model level. We first consider how parallelism and concurrency may be exploited at the object level; later we return to consider how it may be exploited at the model level. We shall also show how discrete-event simulation is necessary for design of powerful new computer architectures for supporting object-oriented computing.

Objects, as self-contained data-code packages with orderly interaction supported by a unified communication protocol, provide "an almost perfect ground for concurrent programming" (Yonezawa and Tokoro, 1987). Thus, there is interest in developing constructs to maximally exploit parallelism and concurrency in object-oriented computing. Constructs that have been suggested include:

- *Future:* The futures construct allows computation to proceed using a stand-in for a yet-to-be-received value rather than waiting for that value to arrive (Lieberman, 1987).
- *Message multicasting:* Several messages may be sent as a group and the replies processed as they arrive (Yonezawa *et al.*, 1987). In multicasting

such messages are directed to a designated subset of objects: broadcasting is a special case in which the designated set is the full set of objects.

- *Asynchronous message sending:* The sender proceeds with other activities while waiting for a reply to a message. This is similar to the futures concept. In either case, there may be point at which nothing further can be done before receiving the reply. Asynchronous message sending requires the user to specify such synchronization points explicitly; the futures construct does this automatically.
- *Self-replication:* An object may replicate itself in order to avoid causing a bottleneck for services it provides (Agha and Hewitt, 1987). In this case, the object becomes a router of incoming messages to its offspring, which perform its original work.
- *Parallel generalization of list structure:* Lists can be formulated as a class of objects. In static terms, list objects are organizers of other objects (including list objects, leading to hierarchical list structures). Taking into account the processing capabilities of objects, list objects can be regarded as coordinators of processing activities of other objects. Moreover, the sequential ordering of lists is an accident of sequential processing and (unordered) parallel sets are more appropriate to parallel processing (Hillis, 1985). One generalization of the list structure of LISP is that embodied in the RACE actor which creates sequences of objects to be computed in parallel and ordered by their time of completion (Lieberman, 1987).

6.1 Par-sets: A Parallel Generalization of List Structures

Another generalization of list structures is called *par-sets* (parallel sets). A par-set object has one instance variable, members, which is capable of holding a finite set of object names. (Some of these names may be those of other par-sets.) Methods for such a class include those for construction (inserting a member, uniting two sets, deleting a member) and querying (membership, inclusion). However, the important methods concern those for coordination of member object behavior. Corresponding to LISP's *map* and *mapcar* forms are the methods: *tell-all* and *ask-all*. When a tell-all message is sent to a par-set, the par-set retransmits the message arguments, the first of which is a method identifier, to all its members, causing them to execute the message operation in parallel. An ask-all method is similar except that the transmitted message is a query and the par-set must package the returned values into a newly created par-set which is returned as its response to the ask-all query. Other methods are naturally provided to par-sets. For example, a variation of ask-all is *match-all* in which the members matching a given pattern are returned as a par-set. "AND" and "OR" parallelism may be exploited using corresponding methods

that wait for the first false reply and the first true reply, respectively, to a Boolean expression to each member to evaluate.

6.2 Decentralization and Parallelism

As noted earlier, the object-oriented modeling paradigm encourages a decentralized form of decision making. This style of representation, whose motivation originates with its beneficial software-engineering attributes, also turns out to have a high affinity for the par-set approach to parallelism. Often, however, the decision algorithm must be reformulated to fit the required form. Consider, for example, a decision ("expert") system for selecting an instrument most suitable to a given task from the leaf classes in Fig. 9. A typical decision algorithm might look like this:

if the material is a solid and needs to be cooled
then use a CHILLER
if the material is a liquid and needs to be heated
then use a HEATING SPIRAL
If the material has liquid constituents that need to be separated
then use a DISTILLATION COLUMN
etc.

Whether written in "soft" form as a set of rules or "hard" coded in a case enumeration statement, this form of decision making is not in the spirit of object-orientation—even though its outcome is an object in an underlying class-inheritance structure. To make it conform to true object-oriented style, first note that it should not have to be rewritten every time a new class (i.e., type of instrument) is added to the system. Fortunately, object-oriented programming systems usually provide the ability to query for the set of leaf classes in a given hierarchy, thus relieving modelers of the responsibility of maintaining their own lists to keep track of the existing classes. Second, the decision must be decentralized so that objects independently apply their expert knowledge and then report back the results. To do this, we distribute the decision rules within polymorphic methods for each class. For example,

CHILLER is given the method *test-applicability*? with definition: Is the material a solid that needs to be cooled?

HEATING SPIRAL is given the method, *test-applicability*? with definition: Is the material a liquid that needs to be heated?

DISTILLATION COLUMN is given the method *test-applicability*? with definition: Does the material have liquid constituents that need to be separated?
etc.

Note that when a new class is added to the system, the modeler must decide whether to allow it to inherit its *test-applicability*? method from a parent or whether to supply it with its own specialized version.

The decision now takes the form:

For each leaf class,
  send its representative object *test-applicability*?
  Select one from those that respond with a "yes" answer.

The parallelism in such a decentralized decision is then readily exploited by the par-set mechanism. We create a par-set whose members list contains a representative object from each leaf class. (This can be performed automatically upon initiation.) To execute the decision algorithm, we send this par-set an ask-all message, asking its members to report back the results of applying their *test-applicability*? method. The result is that all applicability tests are performed in parallel rather than sequentially, as would be the case in the original algorithm.

The par-set concept therefore embodies both the decentralization spirit of object-oriented modeling as well as exploiting the potential parallelism of such decentralization.

6.3 An Object-Oriented Computation Model

The par-set concept may be employed in a yet more fundamental way to represent the structure of objects in order to exploit parallelism in object computations. Recall that the state of an object (its instance variable values) is shared by its methods. Both the instance variables and methods may be organized by par-sets. The instance variables of an object are represented by a par-set whose members are called *attr-val objects* and the methods by a par-set whose members are called *rule-objects*. *Attr-val objects* have two instance variables: *attr* and *val*, where *attr* is the name of an instance variable (attribute) *val* is an object. Methods of *attr-val objects* need only compare attributes and values. *Rule-objects* have (at least) two instance variables: *condition* and *action*. The *condition* is a Boolean predicate on the object state. The *action* is a mapping of the state to itself.

At the high level, computation and control are governed by message-passing protocols. Objects have processing ability and communicate by passing messages that transmit operations and queries. When an object receives a message, it enters the following cycle:

- By means of the ask-all methods, each rule-object tests its condition against the object state.

- The conflict set (rule-objects with satisfied conditions) is resolved to a single rule-object.
- The selected rule-object executes its action resulting in a change in object state. If the incoming message is a query, then a value is returned to the originator of the message. (One or more messages may be sent to other objects in the course of testing conditions and executing actions.)
- These steps are repeated until the conflict set is empty. Then the object goes idle.

Object matching is a ubiquitous process executed in object methods that tests whether an object matches a template. Basic forms of such matching are testing identity and equality of pairs of objects. These are formulated as methods that compare, in parallel, *the attr-val* objects in the par-sets of each object. Such comparison leads to recursion since values of an object's instance variables are also objects in need of equality testing. Thus, parallelism is exploited by the par-set construct at every level in the object structure.

Implementation of the par-set class must rely on a powerful underlying communication system that supports message multicasting. Optics has been proposed as a medium that could supply the high-bandwidth, high-connectivity communication that is required (Louri, 1990a,b). Interestingly, computer-architecture designs that implement new computing models such as this can be studied using discrete-event simulation prior to physical realization (Kim and Zeigler, 1989; Lee, 1990). Indeed, we shall soon argue that such distributed discrete-event simulation is imperative for this enterprise.

7. Distributed Simulation on Concurrent Object-Oriented Systems

Direct mapping of discrete-event models onto concurrent object-oriented processors may achieve only limited success by not taking into account the unique time-based behaviors of such models. However, distributed simulation, the use of parallel and concurrent processing for simulation, is singularly difficult. This is due to the constraint of time ordering of events—the event list maintained by most simulation languages is the major source of sequential bottleneck in this class of programs. Several approaches to distributed simulation of discrete-event models have been developed and experimental systems implemented (Peacock *et al.*, 1979; Hontalas *et al.*, 1989; Bagrodia *et al.*, 1987). The exploitation of the natural parallelism found in multicomponent models must overcome the bottleneck of time-ordered events.

Researchers such as Misra (1986) and Jefferson (1985) represent a distributed simulation as a set of *logical processes*, each with its own clock, which communicate via message passing. Commonly, synchronization aand intercommunication mechanisms are required in distributed simulations in order to maintain strict synchronization of local and global clocks. However, in many multicomponent models a strict synchronization of clocks does not permit much in the way of speedup over sequential (uniprocessor) simulation since not many events can occur simultaneously. (Although this is accepted as a truism by many researchers in distributed simulation, we shall argue that there is parallelism of this kind to exploit.) Thus, strategies have been developed that attempt to decouple logical processes to some extent, i.e., to allow them to advance their local clocks independently to some degree before synchronizations.

Conservative strategies such as Chandy and Misra's (1981) attempt to allow clock advance only to the extent permitted for a correct simulation. This requires continual lookahead to assure that a logical processor does not advance its clock too far—lest messages arrive that bear earlier time stamps. Such an "out-of-synch" situation is construed as deadlock. Conservative strategies therefore attempt to prevent such deadlock. Alternatively, *optimistic* strategies allow processors to advance at their own speeds and employ "rollback" mechanisms to compensate for synchronization violations (Jefferson and Sowizral, 1985). Such mechanisms detect the arrival of out-of-date messages (with time stamps marked earlier then the local clock) and cause the processor to roll back to the state that it would have been in had such a message arrived on time. The processor can resume simulation advance from the recovered state. However, the messages that it sent after the reset clock time are all invalid so that logical processes that received such messages have to be rolled back as well.

7.1 Model Specification and Distributed Simulation

It should be apparent that there may be considerable overhead involved in attempting to exploit model concurrency through distributed simulation mechanisms. Though some simulation data has been obtained to measure the performance of these mechanisms (e.g., Lomow *et al.*, 1988), little is known about the theoretical limits of speedup made possible with distributed simulations. Also, little has been learned about the relation of these limits to properties of discrete event models.

Unfortunately, research in distributed simulation as we have just outlined does not distinguish between a model and a simulator—a key distinction stressed earlier in this chapter (Fig. 1). Both are lumped together in a logical process. As a consequence, expositions are more muddled than necessary.

Indeed, even the concept of simulation correctness is ill defined since there is no independently given model to use in a simulation relation. Moreover, as Cota and Sargent (1990) show, model structure is not well exploited. For example, consider the definition of *influencers* of a control state—the components of a model that can affect a given component's activation condition in that state (Zeigler, 1976). It can be seen that a simulator cannot advance its clock beyond those of its *influencing* simulators (the simulators of the influencers of its model's current control state) but can advance its clock beyond those of all other simulators. Cota and Sargent show how this concept can be exploited to do "internal lookahead" in conservative simulation and to minimize rollbacks and saved states in optimistic simulation. They also show how component model structure—given its modular nature—may be analyzed to make predictions in conservative simulation using "external lookahead."

Another failing of the logical-process concept, due once more to its lack of recognition of the model-simulator distinction, is its failure to distinguish between internal and external events. Recall that there are two fundamental types of events recognized by a DEVS component model: external and internal. An external event results from an input from outside of the component that occurs at an unpredictable time. By contrast, an internal event is determined by the model itself. Now in the logical-process concept, every state change is brought about by a message. Thus, internal events must be effected as messages sent by a component to itself. The only problem with this approach is that interruption of a component's current activity, occasioned by an external event, must be implemented as a cancellation of the current internal-event self-message. Such interruption, a fundamental concept of the modular DEVS formalism, is handled in straightforward fashion by DEVS and by the associated "new worldview" of Cota and Sargent (1990). However, the premier environment for optimistic distributed simulation, the Time Warp environment (Hontalas *et al.*, 1989), fails to recognize the importance of such interruption and to support the necessary self-message cancellation.

7.2 Hierarchical Simulation

Earlier, we mentioned that there may be significant parallelism to exploit in discrete-event simulation apart from the concurrency resulting from overlapping logical processes. The DEVS formalism reveals inherent parallelism in a model through external events. An external event may be sent to one or more model components simultaneously. The arriving external events then may be processed in parallel if the models are each assigned to distinct physical processors. It should be recognized that external events result from internal events. A single internal event in a model may produce an output that

is sent as an external event to one or more model components and may propogate through all levels of the hierarchical structure. The hierarchical and modular construction characteristics of DEVS helps to identify the parallelism is a natural and explicit manner. In the best case, as a hierarchical model is decomposed, the parallelism that is identified in this manner may grow exponentially with each hierarchical level (Zhang and Zeigler, 1989; Zeigler and Zhang, 1990). Examples of models where external-event parallelism may be highly significant are those of decentralized massively parallel computer architectures. Particularly good candidates are those employing broadcasting and multicasting communications, such as those based on the par-set model described previously.

Exploitation of external-event parallelism is one motivation for an approach to use of parallel computer systems called *hierarchical simulation* (Baik and Zeigler, 1985; Wang, 1987; Concepcion and Zeigler, 1988; Zeigler, 1990). Hierarchical simulation is a form of distributed simulation in which hierarchical model structure is preserved. With hierarchical simulation, model structure and behavior can be more easily observed and understood in relation to the real system being modeled in a "one-one analogy" (Dekker, 1984).

Theoretical analysis based on the DEVS hierarchical, modular formalism has indicated that significant speedup, of an exponential nature, is possible by employing hierarchical multiprocessor architectures (Concepcion, 1985; Zeigler and Zhang, 1990). The DEVS-Scheme environment (Zeigler, 1990) provides a vehicle to develop methods for collecting simulation-execution data, and using this data to evaluate alternative hierarchical structures for model simulation. Indeed, the simulation strategy employed in DEVS-Scheme actually realizes a "virtual" multiprocessor architecture. Were each of the processor objects realized by a physical processor, the simulation architecture would represent one possible implementation of a DEVS model on a multiprocessor system. This architecture, called the fully distributed assignment, is one member of a family of possibilities of assignments of model components to processors. Other possibilities exist in which models are mapped to processors in a many-to-one fashion. Such assignments are of interest since theoretical analysis suggests that communication overhead grows linearly with the height of the composition tree in the fully distributed assignment. Thus, using a smaller number of processors than the number of models in the composition tree may result in faster execution time.

7.3 Hierarchical Distributed Simulation Hybrids

The Time Warp Operating System (TWOS) is a special-purpose object-oriented operating system that implements the Time Warp optimistic simulation strategy (Bellenot, 1987; Hontalas *et al.*, 1989). TWOS is a single-

user system and runs a single simulation at a time as a batch process, executing concurrently on as many processors of a distributed system as are allocated. The model to be simulated must be decomposed into objects (logical processes) that interact by passing time-stamped messages. Objects may be mapped onto processors in a one-to-one or many-to-one manner based upon processor and memory limitations. Note that when a many-to-one mapping of objects to processors is used, those objects residing on a single physical processor are executed sequentially. This raises the question of how to assign objects to processors in a time-optimal manner. The hierarchical, modular advantages of the DEVS formalism provides a basis for investigating this and related problems. As already indicated, decomposition of a basic model into a coupled model can expose external events that appear as internal events within the basic model. Parallelism that arises due to simultaneous external events may or may not be worth exploiting depending on the additional overhead required to do so. Since all events in Time Warp are treated as external events, the approach to predicting optimality of assignments given by Zeigler and Zhang (1990) may be directly applicable.

The individual advantages of the DEVS formalism and of Time Warp suggests an approach that combines the DEVS formalism for hierarchical, modular modeling with the optimistic simulation strategy of Time Warp. A hybrid approach of this sort has been implemented in Classic-Ada (Christensen, 1990). The hybrid uses the DEVS formalism for hierarchical, modular model specification and the DEVS abstract-simulator concepts for simulation management on the physically distinct processors. The Time Warp mechanism is implemented using Ada tasking and manages the global distributed simulation process involving communications and synchronization among the DEVS simulators.

In somewhat more detail, a hierarchical DEVS model is decomposed at the top level with each of its components being assigned to a distinct physical processor. Each such component is managed by a hierarchical simulator in the same manner as implemented in DEVS-Scheme (Section 5.3). Such simulators require modification to support the Time Warp mechanism, especially its rollback process. The atomic-model simulator must be modified to save its state at every transaction. The saved states must be stored in a manner that facilitates rapid retrieval. Also, the simulator must have a means to discard old states in an efficient manner.

The root coordinators, which control the simulation process on each physical processor, must be interfaced with the Time Warp manager to execute the distributed optimistic strategy. To interface with Time Warp, the root coordinator must be able to receive inputs from, and send outputs to, other root coordinators. The modified root coordinator will be referred to as a *distributed coordinator*. The external events in the distributed coordinator are of four types: (1) receipt of an external message from another distributed

coordinator, (2) receipt of a rollback request, (3) receipt of a self-scheduling message, and (4) receipt of a request for fossil collection (release of memory used by discardable messages).

A distributed coordinator and its DEVS substructure are encapsulated in an Ada task. A minimum of one distributed coordinator is assigned to each physical processor allocated to the distributed simulation. The size of the DEVS structure which is managed by a distributed coordinator is completely arbitrary, ranging from a single atomic model to a large hierarchical structure. This facilitates the study of optimal model-to-processor mappings as just mentioned.

The port and coupling concepts in DEVS govern the passing of external messages between the distributed coordinators. The internal coupling specification associated with the top-level coupled model is distributed to each distributed coordinator. (Each one need only have the couplings involving its own component.) When a distributed coordinator receives a y-message (output) from its subordinate coordinator, it employs the coupling specification to send it to the appropriate distributed coordinator as an x-message. The receiver will then pass the x-message to its subordinate coordinator. Figure 11a illustrates the passing of external-event messages between coordinators. Figure 11b shows how this approach distributes the work of the top-level root coordinator and its subordinate coordinator that would be employed in a (nonoptimistic) hierarchical simulator.

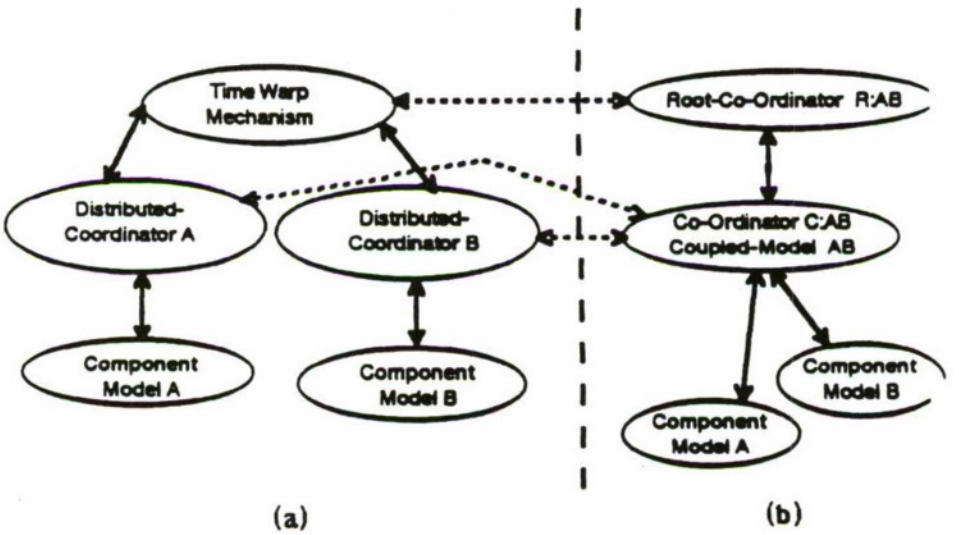


FIG. 11. External message passing between distributed coordinators.

The ability to pass messages among distributed coordinators is facilitated by the use of input and output message buffers. These input and output buffers are implemented as Ada tasks so that all communications between the distributed coordinators may occur asynchronously. The distributed coordinator sends self-scheduling messages directly to its own input buffer; all other messages are sent to its output buffer. A self-scheduling message will be placed in the input buffer only if there are no external event messages in the input buffer with a receive time earlier than the self-scheduling message. This and other event-cancellation mechanisms are made possible by the fundamental DEVS distinction between internal and external events.

The general significance of the hybrid approach is summarized as follows:

- A sound model-specification formalism, such as DEVS, can serve as a firm basis for implementation of models in both sequential and distributed simulation media.
- Parallelism and concurrency may be achieved by exploiting model structure made explicit in such language-independent models.
- Closure under coupling facilitates the mapping of hierarchical models onto multiple processors and concomitant mapping optimality studies.
- The DEVS-Ada handling of the Time Warp manager can be seen as a decentralization of the functions performed centrally by the TWOS. As argued before, such decentralization can be expected to result in increased parallelism and processing speed.

Let us note that the hybrid approach just discussed can be seen as a one-level distributed architecture. By applying the same approach to the model components assigned to physical processors, a two-level architecture is obtained; and by recursion, arbitrary multiple-level architectures may be studied. Since all interfaces have been well defined, no further complexity need be introduced in a multilevel approach.

7.4 The Need for Parallel Architectures to Design Parallel Systems

Parallel computing systems are essential prerequisites for the simulation-based design of complex computer-based systems including computer architectures. Conventional sequential computers are insufficient for two reasons. First, as we have indicated before, simulation run times place severe limitations on the complexity of systems that can be studied. This is especially true for simulation of parallel architectures whose very design is antithetical

to sequential processing—i.e., the techniques employed to maximize parallelism in distributed systems usually place extreme demands on sequential simulations of such systems. For example, recall that the decentralization of decision making encouraged by the object-oriented paradigm brings about parallelism by down-loading as much of the decision making to the objects as possible. Thus, to simulate message multicasting by the par-set object in a straightforward manner on a uniprocessor, each receiver must be examined to see how it responds to the message—a calculation likely to be N times more complex (where N is the number of par-set members) than the original centralized decision. In some cases, one can reduce the sequential simulation time by maintaining information that will predict how receivers respond. For example, if the problem is that of message acceptance, this information can be used to predict which receivers will in fact accept the message, thereby reducing the examination to them. However, in this case, the correspondence between the model and real architecture becomes problematic. It is much better to use a parallel machine as simulation platform, for which a better, albeit not perfect, approximation can be made to the parallel architecture under study. For example, broadcasting can be simulated with lower time cost with a hypercube architecture because at least some, though not all, transmissions will be simultaneous.

The second limitation of simulating parallel systems on uniprocessors relates less to simulation time and more to the ultimate validity of sequential simulation of parallel processes. Since models are only approximations to the reality they represent, results of sequential simulation can never be conclusive and must be validated against the behavior of a real parallel machine. The nature of the machine has a profound effect on the algorithm design and software. Simulation results obtained on a different machine may therefore be misleading. A parallel and distributed machine allows one to gain much more insight into the behavior of parallel architectures—an insight that will feed back to the development of better models of parallel processing. Thus, existing parallel computing systems are essential to bootstrap our way to yet faster parallel systems in the future.

8. Conclusion

Object-oriented modeling and discrete-event simulation provide a powerful basis for design and analysis of computer-based systems in such applications as flexible manufacturing, automation, and robotics. The object-oriented approach provides powerful modeling concepts to support computer-based tools for complex system design. Discrete-event simulation has a long history

of association with the object-oriented paradigm and provides the critical ability to study the dynamic behavior of models that are defined with object-oriented means.

To fully appreciate the future role of discrete-event simulation in the design of computer-based systems requires that we extend the context of the discussion to that of knowledge-based design of systems. As mentioned earlier, object-oriented modeling bears a direct relation to the frame-based knowledge-representation schemes of artificial intelligence (AI). The incorporation of AI knowledge-representation schemes within simulation models results in knowledge-based simulation systems (Reddy *et al.*, 1986; Fox and Smith, 1984). Such schemes can be used not only to organize information about the nature of the objects involved in the simulation, but also within components themselves so as to model intelligent agents.

A use of AI techniques with great potential is the application to modeling and simulation methodology. Expert systems, the spearheads of AI commercial application, are software systems that incorporate significant components of human knowledge and expertise in a limited problem domain. Since modeling and simulation is a difficult, labor-intensive process, simulation researchers have been looking for ways in which expert systems could aid in it. Such systems could lessen the need for modelers to be experts in simulation programming, advise on selection of models or their components for specific purposes, interpret simulation results with expert statistical judgment, etc.

While such potential is extremely attractive in expanding the user-friendliness and range of applicability of simulation, progress is not likely to be rapid. The fact is that to formalize the knowledge needed to conduct a meaningful simulation study is extremely difficult. Knowledge cannot be entered as a discrete set of independent units (rules) gleaned from observation of an expert solving a problem in a limited domain. (The domain of modeling and simulation is vast and there are no experts in all its facets.) Rather, knowledge must be coded adhering to coherent systematization derived from a sound conceptual framework. The DEVS theory of discrete-event systems described in this chapter holds promise for providing such a framework.

Modeling and simulation are usually performed within the context of system design, control, or management. Although early computer-aided design (CAD) and decision-support systems had little in the way of simulation tools, research has begun to integrate these tools within such systems. (See, for example, Rozenblit *et al.* (1989); Rozenblit and Huang (1991).)

As a backdrop for design and decision making, a simulation environment must be able to support rapid development of various models at different levels of abstraction/aggregation and oriented toward diverse objectives. To

obviate having to start from scratch each time a model is needed, models may be kept in an organized library called a *model base* (Zeigler, 1990). Object-oriented knowledge-representation schemes will be increasingly employed to organize models in such model bases and to enhance reusability of models, model-base integration, and evolvability.

Reusability of models requires that the model base be populated by models in modular form enabling hierarchical assembly and disassembly. As we have seen, conventional discrete-event–simulation languages are not well suited to these demands. Although it is possible (with some difficulty) to adapt such languages, a new generation of environments is being developed to support such hierarchical modular model construction. Once again, formalisms such as DEVS should provide a firm foundation for such next-generation environments.

To summarize, we have reviewed the fundamental concepts of object-oriented modeling and discrete-event simulation. To advance the application of these techniques, research is needed in a number of interrelated areas:

- Formalisms for discrete-event model specification that exploit the advantages of object-oriented paradigms and extend their dynamic modeling capability.
- Concurrent and parallel object-oriented computing systems to provide the platforms for distributed simulation.
- Distributed simulation strategies to maximally exploit the parallelism in models and objects and fully utilize the capabilities of the underlying computing platform.
- Use of existing parallel computer-simulation platforms to bootstrap the development of yet more powerful parallel computer architectures.

Progress in these areas will bring the use of computer simulation to significantly new levels of capability to support the design of the complex computer-based systems under worldwide development.

References

Adelsberger, H. H., (1986). Rule Based Object Oriented Simulation Systems. *In* "Intelligent Simulation Environments" (P. A. Luker and H. H. Adelsberger, eds.). Simulation Series, vol. 17. Society of Computer Simulation, San Diego, California.

Agha, G., and Hewitt, C. (1987). Concurrent Programming Using Actors. *In* "Object-Oriented Concurrent Programming" (A. Yonezawa and M. Tokoro, eds.). MIT Press, Cambridge, Massachusetts.

Arbib, X., Michael, A., and Padulo, L. (1974). Systems Theory: A Unified State Space Approach to Continuous and Discrete Systems, W. B. Saunders, Philadelphia, Philadelphia.

Bach, W. W. (1989). Ada, An Object-Oriented Language. *J. Pascal, Ada, Modula-2* (March-April), 19–25.

Baik, D. K., and Zeigler, B. P. (1985). Performance Evaluation of Hierarchical Distributed Simulators. *Proc. Winter Simulation Conf.*

Balci, O. (1988). The Implementation of Four Conceptual Frameworks for Simulation Modeling in High-Level Languages. *Proc. Winter Simulation Conf.* Society of Computer Simulation, San Diego, California.

Bellenot (1987). Distributed Simulation and the Time Warp Operating System. *ACM Operating Systems Review*, 77–93.

Bobrow, D. G., and Stefik, M. J. (1983). "The LOOPS Manual." Xerox, Palo Alto, California.

Bagrodia, R. L., Chandy, K. M., and Misra, J. (1987). A Message-Based Approach to Discrete Event Simulation. *IEEE Trans. Soft. Eng.*, vol. SE-13, no. 6, pp. 654–665.

Bryan, O. F. (1989). MODSIM II—An Object-Oriented Simulation Language for Sequential and Parallel Processors. *Proc. Winter Simulation Conf.*, pp. 205–210.

Cassandras, C. G., and Strickland, S. G. (1989). Sample Path Properties of Timed Discrete Event Systems. *Proceedings of the IEEE* 77, no. 1 (January), 59–71.

Chandy, K. M., and Misra, J. (1981). Asynchronous Distributed Simulation via a Sequence of Parallel Computations. *Communications of the ACM* **24**, no. 11 (April), 198–206.

Christensen, E., and Zeigler, B. P. (1990). Distributed Discrete Event Simulation: Combining DEVS and Time Warp. *Proc. AI and Sim. Eastern Multiconf.* Society of Computer Simulation, San Diego, California.

Concepcion, A. I. (1985). Mapping Distributed Simulators onto the Hierarchical Multibus Multiprocessor Architecture." *In* "Distributed Simulation 1985" (P. Reynolds, ed.). Society of Computer Simulation, San Diego, California.

Concepcion, A. I., and Zeigler, B. P. (1988). DEVS Formalism: A Framework for Hierarchical Model Development. *IEEE Transactions on Software Engineering*, **14**, no. 2 (February), 228–241.

Cota, B. A., and Sargent, R. G. (1990). A New Version of the Process World View for Simulation Modelling. CASE Center Tech. Rept. no. 9003, Syracuse University, Syracuse, New York.

Dahl, O. J., and Nygaard, K. (1966). Simula: An Algol-Based Simulation Language. *CACM* 9, 671–688.

Dekker, L. (1984). Concepts for an Advanced Parallel Simulation Architecture. *In* "Simulation and Model-Based Methodologies: An Integrative View" (T. I. Ören, M. S. Elzas, and B. P. Zeigler, eds.), pp. 235–280. Springer-Verlag, New York.

Delaney, W., and Vaccari, E. (1989). "Dynamic Models and Discrete Event Simulation." Marcel Dekker, New York.

Ehr, W., and Wnuk, A. (1985). Discrete Event Simulation of a Model Family with Boris. *Proc. 11 IMAC World Congress*, Oslo, Norway.

Elzas, M. S., Ören, T. I., and Zeigler, B. P. (1986). "Modelling and Simulation Methodology in the Artificial Intelligence Era." North-Holland, Amsterdam.

Fox, M. S., and Smith, S. F. (1984). ISIS: A Knowledge Based System for Factory Scheduling. *Expert Systems*, 1, 25–49.

Franta, W. R. (1977). "The Process View of Simulation." North-Holland, Amsterdam.

Futo, I. (1985). Combined Discrete/Continuous Modeling and Problem Solving. *In* "AI, Graphics and Simulation" (G. Birtwistle, ed.). Society of Computer Simulation, San Diego, California.

Garzia, R. F., Garzia, M. R., and Zeigler, B. P. (1986). Discrete Event Simulation. *IEEE Spectrum* (December), 32–36.

Goldberg, A., and David, R. (1983). "Smalltalk-80: The Language and its Application." Addison-Wesley, Reading Massachusetts.

Glynn, P. W. (1989). A GSMP Formalism for Discrete Event Systems. *Proceedings of the IEEE* 77, no. 1 (January), 14–23.

Hayes, P. J. (1981). The Logic of Frames. *In* "Readings in Artificial Intelligence" (B. L. Weber and N. J. Nilsson, eds.), pp. 451–458.

Hillis, D. (1985). "The Connection Machine." MIT Press, Cambridge, Massachusetts.

Ho, Y. (1989). Editors Introduction. *Special Issue on Dynamics of Discrete Event Systems. Proceedings of the IEEE* 77, (1).

Hontalas, P., Jefferson, D., and Presley, M. (1989). Time Warp Operating System Version 2.0 User's Manual. Pasadena, California, Jet Propulsion Laboratory.

Hooper, J. W. (1986). "Strategy-Related Characteristics of Discrete Event Languages and Models." *Simulation*, **46** (4), 153–159.

Jefferson, D. R. (1985). Virtual Time. *ACM Trans. Prog. Lang. Sys.* 7 (3), 198–206.

Jefferson, D., and Sowizral, H. (1985). Fast Concurrent Simulation Using the Time Warp Mechanism. *In* "Distributed Simulation 1985" (P. Reynolds, ed.). Society of Computer Simulation, San Diego, California.

Keene, S. E. (1988). "Programming in Common Lisp Object-Oriented System." Addison-Wesley, Massachusetts.

Kim, T. (1988). A Knowledge-Based Environment for Hierarchical Modelling and Simulation. Ph.D. dissertation, University of Arizona, Tucson.

Kim, T. G., and Zeigler, B. P. (1989). A Knowledge-Based Environment for Investigating Multicomputer Architectures. *Information and Software Technology*, **31** (10), 512–520.

Klahr, P. (1986). Expressibility in ROSS, an Object-Oriented Simulation System. *In* "Artificial Intelligence in Simulation" (G. C. Vansteenkiste, E. J. H. Kerckhoffs, and B. P. Zeigler, eds.). Society of Computer Simulation, San Diego, California.

Lee, C. (1990). A Hierarchical, Modular Modelling and Simulation Environment for AI Multicomputer Design. Ph.D. dissertation, Department of Electrical and Computer Engineering, University of Arizona, Tucson.

Lieberman, H. (1987). Concurrent Object-Oriented Programming in Act 1. *In* "Object-Oriented Concurrent Programming" (A. Yonezawa and M. Tokoro, eds.). MIT Press, Cambridge, Massachusetts.

Lomow, G., and Baezner, D. (1989). A Tutorial Introduction to Object-Oriented Simulation and SIM++. *Proc. Winter Simulation Conf.*, pp. 140–146.

Lomow, G., Cleary, J., Unger, N., and West, D. (1988). A Performance Study of Time Warp. *Proc. Distributed Simulation '88*, San Diego, California.

Louri, A. (1990a). A Preliminary Version of an Optical Dataflow Multiprocessing System. *Proc. Hawaii Intl. Conf. on Sys. Sci.—23*, pp. 121–130.

Louri, A. (1990b). Massively Parallel Computing with Symbolic Substitution. IEEE Trans. on Par. and Distr. Comp. (to appear).

Manivannan, S. (1989). Just-in-Time Simulation Using Artificial Intelligence. *Proc. Winter Simulation Conf.* Society of Computer Simulation, San Diego, California.

Mesarovic, M. D., and Takahara, Y. (1975). "General Systems Theory: Mathematical Foundations." Academic Press, New York.

Meyer, B. (1988). "Object-Oriented Software Construction." Prentice-Hall, Englewood Cliffs, New Jersey.

Meyer, J. F., Movaghar, A., and Sanders, W. H. (1985). Stochastic Activity Networks: Structure, Behavior and Application. *Proc. Int. Workshop on Timed Petri Nets*, Torino, Italy, pp. 106–115.

Middleton, S., and Zanconato, R. (1986). BLOBS: An Object-Oriented Language for Simulation and Reasoning. *In* "Artificial Intelligence in Simulation" (G. C. Vansteenkiste, E. J. H. Kerckhoffs, and B. P. Zeigler, eds.), pp. 130–135. Society of Computer Simulation, San Diego, California.

Misra, J. (1986). Distributed Discrete Event Simulation. *ACM Computing Surveys*, **18** (1), 39–65.

Mittelmann, R. (1990). Object-Oriented Design of CAST Systems. *In* "Computer Aided Systems

Theory—Eurocast '89" (F. Pichler and R. Moreno-Diaz, eds.), pp. 69–75. Lecture Notes in Computer Science. Springer Verlag, Berlin.

Mittelmann, R., and Praehofer, H. (1990). Design of an Object-Oriented Kernel for CAST. *In* "Computer Aided Systems Theory—Eurocast '89" (F. Pichler and R. Moreno-Diaz, eds.), pp. 76–85. Lecture Notes in Computer Science. Springer Verlag, Berlin.

Narain, S., and Rothenberg, J. (1989). A History-Oriented Calculus for Simulating Dynamic Systems. *Proceedings of Fourth AAAI Workshop on AI and Simulation*, pp. 78–81.

O'Keefe, R. (1986). Simulation and Expert Systems—A Taxonomy and Some Examples. *Simulation* **46** (1), 10–16.

Ören, T. I. (1984). GEST—A Modeling and Simulation Language Based on System Theoretic Concepts. *In* "Simulation and Model-Based Methodologies: An Integrative View" (T. I. Ören, B. P. Zeigler, and M. S. Elzas, eds.), pp. 3–40. North-Holland, Amsterdam.

Ören, T. I. (1987). Taxonomy of Simulation Model Processing. *In* "Encyclopedia of Systems and Control" (M. Singh, ed.). Pergamon.

Ören, T. I., and Zeigler, B. P. (1979). Concepts for Advanced Simulation Methodologies. *Simulation* **32** (3), pp. 69–82.

Ören, T. I., Elzas, M. S., and Zeigler, B. P. (1984). "Simulation and Model-Based Methodologies: An Integrated View." Springer-Verlag, New York.

Overstreet, C. M., and Nance, R. E. (1986). World View Based Discrete Event Model Simplification. *In* "Modeling and Simulation Methodology in the AI Era" (M. S. Elzas, T. I. Ören, and B. P. Zeigler, eds.), pp. 165–170. North-Holland.

Padulo, L., and Arbib, M. A. (1974). "System Theory." Saunders, Philadelphia.

Peacock, J., Wong, H. W., and Manning, E. (1979). Distributed Simulation Using a Network of Processors. *Computer Networks* **3** (1).

Pedgen, C. D. (1983). Introduction to SIMAN. *Proc. Winter Simulation Conf.* Society of Computer Simulation.

Peterson, J. L. (1981). "Petri Net Theory and Modeling of Systems." Prentice Hall, Englewood Cliffs, New Jersey.

Pritsker, A. A. B. (1979). Compilation of Definitions of Simulation. *Simulation* **33**, 61–63.

Reddy, Y. V., Fox, M. S., Husain, N., and McRoberts, M. (1986). The Knowledge-Based Simulation System. *IEEE Software* (March), 26–37.

Rozenblit, J. W., Hu, J., and Huang, Y. (1989). An Integrated, Entity-Based Knowledge Representation Scheme for System Design. Proc. NSF Design *Res. Conf.*, pp. 393–408, Amherst, Mass. June 1989.

Rozenblit, J. W., and Huang, Y. (1991). Rule-Based Generation of Model Structures in Multi-faceted System Modelling and System Design. *ORSA J. on Computing* (Vol. 3 No. 4).

Ruiz-Mier, S., and Talavage, J. A Hybrid Paradigm for Modeling of Complex Systems. *In* "Artificial Intelligence, Simulation and Modelling" (L. A. Widman, K. A. Loparo, and N. Nielsen, eds.), pp. 381–395. Wiley, New York.

Sanders, W. H. (1988). Construction and Solution of Performability Models Based on Stochastic Activity Networks. Ph.D. diss. University of Michigan, Ann Arbor.

Sanders, W. H., and Meyer, J. F. (1989). Reduced Base Model Construction Methods for Stochastic Activity Networks. *Proc. Third Int. Workshop on Petri Nets and Performance Models*, Kyoto, Japan.

Sauer, C. H., and Chandy, K. M. (1980). "Computer Systems Performance Modelling." Prentice Hall, Englewood Cliffs, New Jersey.

Sevinc, S., and Zeigler, B. P. (1988). Entity Structure Based Design Methodology: A LAN Protocol Example. *IEEE Transactions on Software Engineering* **14**, no. 3 (March), 375–383.

Shannon, R. E. (1975). "Systems Simulation: The Art and the Science." Prentice-Hall, Englewood Cliffs, New Jersey.

Simon, H. A. (1969). "The Sciences of the Artificial." MIT Press, Cambridge, Massachusetts.

Singh, M. G. (1987). "Systems and Control Encyclopedia." Pergamon, Oxford, England.

Stroustrup, B. (1986)." The C++ Programming Language." Addison-Wesley, Reading, Massachusetts.

Thomasma, T., and Ulgen, O. M. (1988). Hierarchical, Modular Simulation Modelling in Icon-Based Simulation Program Generators for Manufacturing. *Proc. Winter Simulation Conf.*, San Diego, pp. 254–262.

Weinreb, D., Moon, D. and Stallman, R. (1983). Lisp Machine Manual, MIT, Cambridge, Massachusetts.

Wymore, A. W. (1967). "A Mathematical Theory of Systems Engineering: The Elements." Wiley, New York.

Yonezawa, A., and Tokoro, M. (1987). "Object-Oriented Concurrent Programming." MIT Press, Cambridge, Massachusetts.

Yonezawa, A., Shibayama, E., Takada, T., and Honda, Y. (1987). Modelling and Programming in an Object-Oriented Concurrent Language ABCL/1. *In* "Object-Oriented Concurrent Programming" (A. Yonezawa and M. Tokoro, eds.). MIT Press, Cambridge, Massachusetts.

Zadeh, L. A., and Desoer, C. A. (1963). "Linear System Theory: The State Space Approach." McGraw-Hill, New York.

Zeigler, B. P. (1976). "Theory of Modelling and Simulation." Wiley, New York (reissued by Krieger, Malabar, Florida, 1985).

Zeigler, B. P. (1984). "Multifaceted Modelling and Discrete Event Simulation." Academic Press, London and Orlando, Florida.

Zeigler, B. P. (1985a). System-Theoretic Representation of Simulation Models. *IIE Transactions* 16 (1), 19–34.

Zeigler, B. P. (1985b). Discrete Event Formalism for Model Based Distributed Simulation. *Proc. of the Conference on Distributed Simulation*, pp. 3–7.

Zeigler, B. P. (1990). "Object-Oriented Simulation with Hierarchical Modular Models: Intelligent Agents and Endomorphic Systems." Academic Press, Boston.

Zeigler, B. P. and Zhang, G. (1990). Mapping Hierarchical Discrete Event Models to Multiprocessor Systems: Algorithm, Analysis, and Simulation. *J. Parallel and Distributed Computers* (in press).

Zhang, G., and Zeigler, B. P. (1989). DEVS-Scheme Supported Mapping of Hierarchical Models onto Multiple Processor Systems. *Society of Computer Simulation Multiconference on Distributed Simulation*, Tampa, Florida, pp. 64–69.

IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, VOL. 20, NO. 5, SEPTEMBER/OCTOBER 1990

System Entity Structuring and Model Base Management

T. G. KIM, MEMBER, IEEE, C. LEE, STUDENT MEMBER, IEEE, E. R. CHRISTENSEN, MEMBER, IEEE, AND B. P. ZEIGLER, SENIOR MEMBER, IEEE

Abstract—System entity structure (SES) is a structural knowledge representation scheme that contains knowledge of decomposition, taxonomy, and coupling of a system. Formally, the SES is a labeled tree with attached variable types that satisfy certain axioms. Described is a realization of the SES in Scheme (a Lisp dialect) called ESP-Scheme. The computer representation of SES and main operations on SES are presented, and then facilities provided by ESP-Scheme are described. Two examples of application are discussed: a parallel processor model and a simulation study of a university phone registration system. ESP-Scheme acts as a model base management system in DEVS-Scheme, a knowledge-based simulation environment. It supports specification of the structure of a family of models, pruning the structure to a reduced version, and transforming the latter to a simulation model by synthesizing component models in the model base.

I. Introduction

THE NEED for greatly enhanced support of simulation modeling activities is increasingly being recognized. Model development methodology, state-of-the-art software technology, and artificial intelligence, especially knowledge-based concepts [6], [8], [14]–[16], [18], [23], [26] have been suggested as means to achieve advanced simulation environments [1], [2], [10], [17].

To cite one application context, with the new importance given to simulation in military training and analysis, there is strong demand for a simulation workbench, i.e., an evolving architecture providing an integrated collection of standards, protocols, and tools to enable analysts ... to build, understand, use and reuse simulations ..." [9]. Such a workbench would have to provide a sharable repository of models and a means of assisting users to browse through, and understand the content of, stored models to confidently select those relevant to the current analytic task. Although this need for model base management is recognized, there are few concrete proposals to meet it [9].

The model base management system discussed in this paper, based on the System Entity Structure, is the only one, to the authors' knowledge, that provides a working software system capable of addressing the needs of model repositories just stated. We note that "model management" here refers to the storage, cataloging, and reuse of component models to synthesize simulation models to meet current objectives. A second sense of the term, related though distinct, is used in decision support systems, where "models" refer to more generalized mathematical and analytic tools for operating on input data to produce output helpful in decision making [25].

Our approach to model base management stems from the need to develop computerized support for multifaceted modeling methodology [33]. This methodology recognizes that most systems of interest require a multiplicity of models to be developed since a single all-encompassing model, however desirable as a conceptual goal, is not a practical object. By decomposing questions and objectives into an ordered structure of experimental frames (specifications of experimentation conditions), useful partial models may be constructed, validated, and employed, each attuned to a limited set of frames.

A multifaceted model base may contain a variety of models and associated formalisms [34] that are organized with the help of the system entity structure (SES), which directs the synthesis of models from components in the mdoel base. The SES is a knowledge representation scheme that combines decomposition, taxonomic, and coupling relationships. Knowledge representation is generally accepted to be a key ingredient in designing artificial intelligence software. Previous work identified the need for representing the structure and behavior of systems in a declarative scheme related to frame-theoretic and object-based formalisms [32], [8], [27]. The elements represented are motivated, on the one hand, by systems theory [28], [29] concepts of decomposition (i.e., how a system is hierarchically broken down into components) and coupling (i.e., how these components may be interconnected to reconstitute the original system). On the other hand, systems theory has not focused on taxonomic relations, as represented, for example, in frame-hierarchy knowledge representation schemes [8], [27]. In the SES scheme, such representation concerns the admissible variants of components in decompositions and the further specializations of such variants. The interaction of decomposition, coupling, and taxonomic relations in an SES affords a compact specification of a family models for a given domain.

Manuscript received December 2, 1989; revised March 24, 1990.

T. G. Kim is with the Department of Electrical and Computer Engineering, University of Kansas, Lawrence, KS 66045.

C. Lee, E. R. Christensen, and B. P. Ziegler are with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ 85721.

IEEE Log Number 9036358.

0018-9472/90/0900-1013$01.00 ©1990 IEEE

One application of this framework is to the design of systems. Here the SES serves as a means of organizing the possible configurations of a system to be designed which may be extracted with a pruning process [11], [13], [33]. Pruning is a goal-directed process, where the goal is formulated by the modeler (designer) to meet the system design requirements. Pruning reduces the set of candidate models to those suitable for the problem under study.

The implementation described here adheres to the formal characterization of the SES given in [35]. One main difference with earlier implementations [3], [7] is that the present implementation is fully integrated with the DEVS-Scheme environment that provides the underlying model synthesis and simulation layer. In addition, several powerful features to assist in combining structures to form larger ones have been added. Although space precludes a full exposition of SES concepts, the reader will find a brief review followed later by examples that illustrate the major features.

DEVS-Scheme realizes Zeigler's DEVS (discrete event system specification) formalism in Scheme (a Lisp dialect) [12], [13], [31]. The environment supports building models in a hierarchical, modular manner, a systems-oriented approach not possible in conventional languages [5]. To organize the complex hierarchical structures of models developed using DEVS-Scheme, a model base management system is highly desirable. The system entity structuring formalism developed by Zeigler [33] is one such tool for the model base management. ESP-Scheme is a realization of the system entity structuring formalism in the Scheme environment. ESP-Scheme supports hierarchical specification of the structure of a family of models, pruning the structure to a reduced version, and transforming the latter to a simulation model by synthesizing component models in the model base developed using DEVS-Scheme.

This paper first reviews the system entity structuring formalism and then describes some overall features of the ESP-Scheme, including representation and operations of the system entity structure. It also presents an outline of a knowledge-base framework for modeling and simulation, based on the entity structure and model base. The paper illustrates the concepts presented with two examples: a parallel processor model and a simulation study of a university phone registration system.

II. The DEVS Formalism and DEVS-Scheme

The DEVS (discrete event system specification) formalism developed by Zeigler supports specification of discrete event systems in hierarchical, modular fashion. In the formalism, one must specify 1) basic models from which larger ones are built, and 2) how these models are connected together in hierarchical fashion. Definitions for a basic model, called an atomic DEVS, and a general form of model, called a coupled DEVS, can be found in [33].

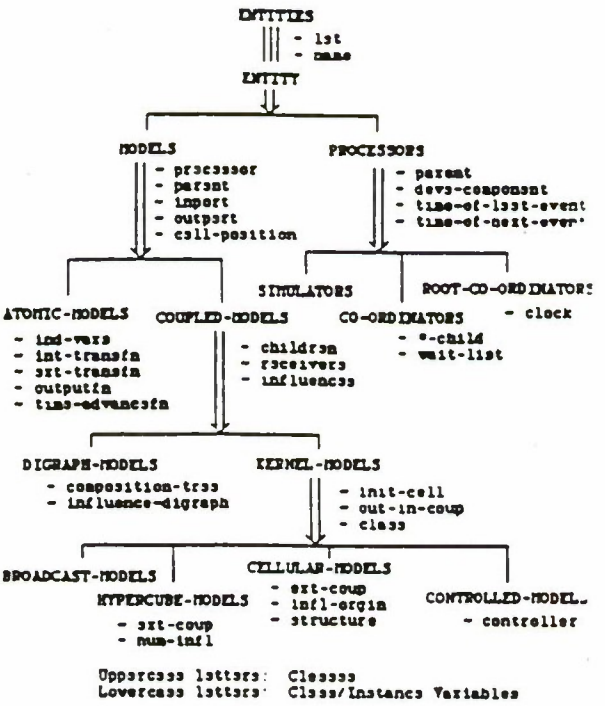


Fig. 1. Class hierarchy of DEVS-Scheme.

DEVS-Scheme, a general-purpose modeling and sir lation environment, is an implementation of DEVS malism in a LISP-based, object-oriented program system [31], [13]. DEVS-Scheme is written in the Scheme language, which runs on DOS compatible mic computers and under a Scheme interpreter for the T Instruments Explorer. DEVS-Scheme is implemente a shell that sits upon PC-Scheme in such a way that al the underlying Lisp-based and object-oriented prog ming language features are available to the user. result is a powerful basis for combining AI and simulat techniques.

The class specialization hierarchy in DEVS-Scher shown in Fig. 1. All classes in DEVS-Scheme are s classes of the universal "class entities" that provide to for manipulating objects in these classes (these object hereafter called entities). The inheritance mechanism sures that such general facilities need only be defin once and for all.

Models and processors, the main subclasses of ent provide the basic constructs needed for modeling a simulation. "Models" is further specialized into the t classes "atomic-models" and "coupled-models," whic turn are specialized into more specific cases, a proc that may be continued indefinitely as the user builds m specific model base. The class definitions for atomic-m els and coupled-models closely parallel those for aton DEVS and coupled DEVS, respectively. Cl "processors," on the other hand, have three specia tions: simulators, coordinators, and root coordinators. simulators, coordinators, and root coordinators carry o the simulation of a model in a manner following hierarchical abstract simulator concepts in [33]. Thus

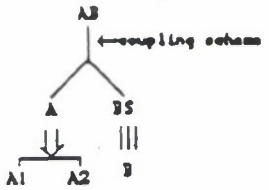


Fig. 2. System entity structure.

[EVS-Scheme environment

- supports modular hierarchical model construction,
- allows independent testing of components models,
- separates models from experimental frames,
- supports distributed simulation.

A detailed description of all classes in DEVS-Scheme is available in [34] and [13]. Examples will be provided to illustrate salient features.

III. System Entity Structuring Formalism: A Review

A system entity structure (SES) is a knowledge representation scheme that contains the decomposition, coupling, and taxonomy information necessary to direct model synthesis [33], [19]. Formally, the SES is a labeled tree with attached variable types that satisfy five axioms: alternating mode, uniformity, strict hierarchy, valid brothers, and attached variables. A detailed discussion of the axioms is available in [34].

A. Three Relationships in SES

There are three types of nodes in the SES—entity, aspect, and specialization—which represent three types of knowledge about the structure of systems. The entity node, having several "aspects" and/or "specializations," corresponds to a real-world object. The aspect node (a single vertical line in the labeled tree of Fig. 2) represents one "decomposition," out of many possible, of an entity. Thus the children of an aspect node are entities, distinct components of the decomposition. The specification node (double vertical arrow in the labeled tree of Fig. 2) represents a way in which a "general" entity can be categorized into "special" entities. As shown in Fig. 2, attached to an aspect node is a coupling scheme, which specifies external input, external output, and internal couplings of a system and its components.

A "multiple entity" is a special entity that consists of a collection of homogenous components. We call such components a multiple decomposition of the multiple entity. The aspect of such a multiple entity is called multiple aspect (triple vertical lines in the labeled tree of Fig. 2). The representation of such a multiple entity is as follows. A multiple entity BS and its components B are represented by BS, three vertical lines, and B from the top down. Note that instead of presenting all B's for BS's components, only one B is placed in the labeled tree. The number of B's is specified by a variable, which is attached to the multiple aspect node.

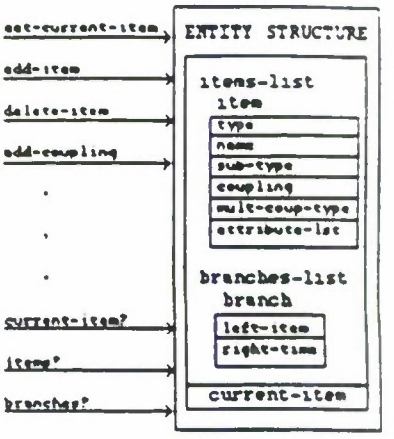


Fig. 3. System entity structure module.

B. Operations on SES

An SES represented by a labeled tree consists of branches and nodes (nodes are also called items). An item in the SES is one of the three types: entity, aspect, or specialization. Some of the operations performed on the SES are adding an item to the SES, deleting an item from the SES, attaching variables to items in the SES, deleting variables from items in the SES, pruning the SES, and transforming the pruned SES into a model.

The construction of a SES is a sequence of adding new items—entities, aspects or specializations—to the SES. The deletion operation, which deletes entities and associated branches from the SES, can be applied only to those entities with no aspects. The pruning operation extracts a substructure of the SES by selecting one aspect and/or one specialization for each entity in the SES. The pruning operation ultimately reduces the SES to a composition tree that contains all the information about the structure of the reduced version of the model. The transform operation synthesizes the reduced version of the model in a hierarchical fashion from components in the model base.

IV. Implementation of ESP-Scheme

A. Representation of SES in ESP-Scheme

The SES is implemented by a module called "entity-structure"—a package of hidden variables and associated operations—as shown in Fig. 3. Lists of items and branches are main variables representing a tree structure for the SES. The variable "current-item" points to the current item in the SES, under which new items can be added. Each item in the items-list is represented by a structure type called item, the fields of which include "type," "name," "coupling," "multi-coup-type," and "attributes-lst." Each branch in the branches-list is represented by another structure type called branch, which maintains an ordered pair of two items, left- and right-items, in the SES.

The field "type" in item structure represents the type of an entity in an entity structure whose range is {entity, aspect, specialization}. The field "name" is used to identify an entity by its name. The field "coupling" is used to specify a coupling scheme of a model specified by a system entity structure. The coupling scheme is a collection of three coupling specifications: external input, external output, and internal coupling. Each of three coupling specifications is represented by a set of ordered pairs of ports. The representation of the coupling scheme is compatible with that of DEVS-Scheme. The field "sub-type" with range {multiple-entity, multiple-aspect, multiple-children} represents information on multiple entities and multiple decomposition. The field "mult-coup-type" with range {broadcast, hypercube, cellular, controlled} is used to specify the coupling scheme associated with the corresponding subclass of kernel models in DEVS-Scheme [11], [13], [34]. Items in SES may have attributes that characterize their features. The field "attributes-lst" maintains a list of such attributes, each of which is a pair of variables and its value.

The main operations on the SES are "set-current-item," "add-item," "add-mult," "delete-item," "add-coupling," "prune," and "transform." The construction of a system BUF-PROC SES will be used to illustrate the operations. BUF-PROC is a processing element containing a buffer cascaded with a processor. The type of buffer is assumed to be either FIFO (first-in first-out) or LIFO (last-in first-out), and will be selected by the user in the pruning process. Once the SES of the BUF-PROC is built, we prune the BUF-PROC entity structure and transform the pruned BUF-PROC into the desired model. The first step in the creation of an SES is defining the root entity. Line (1) of Fig. 4(a) illustrates the creation of an entity structure named BUF-PROC. Once the SES with the root entity BUF-PROC has been created, items are added to the SES. However, a sequence of adding items should be such that the resulting SES satisfies the axiom of "alternating mode." SES axioms are automatically checked by the entity structure module as the operations are processed. Since the next items to be added are either aspects or specializations, we add an aspect called comp-dec under the root entity BUF-PROC (line (2) of Fig. 4(a)). To add other items under the aspect comp-dec requires setting the current item to the aspect comp-dec (line (3) of Fig. 4(a)). After the current item is set, two components, a buffer and a processor, are added one by one (lines (4) and (5) of Fig. 4(a)). Note that the current item of the SES is still at the aspect comp-dec. When an item with type specialization is added under the entity BUFFER, the current item must be set to the BUFFER (line (6) of Fig. 4(a)). Then line (7) adds a specialization, "buf-type", under the entity BUFFER. Similarly, lines (8), (9), and (10) add two items, FIFO-BUF and LIFO-BUF, under the specialization "buf-type."

The coupling scheme of the BUF-PROC system, which is attached to the aspect comp-dec, can be specified by the operation "add-coupling." This operation requires the

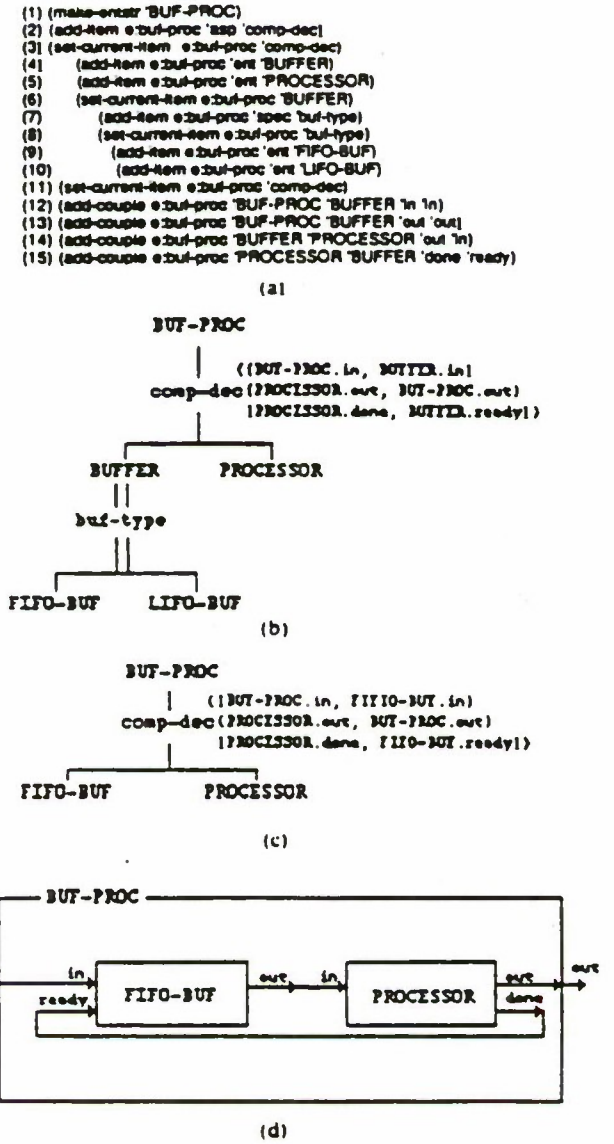


Fig. 4. Model structure representation using systems entity structure. (a) ESP-Scheme code. (b) System entity structure. (c) Pruned entity structure. (d) Transformed model.

specification of two entity names and their respective port names. The operation "add-coupling" specifies both internal and external coupling of the BUF-PROC system. Lines (12) and (13) of Fig. 3(a) specify the external coupling scheme, while lines (14) and (15) of Fig. 4(a) specify the internal coupling scheme of the BUF-PROC. The resulting SES for the BUF-PROC system is shown in Fig. 4(b).

A "pure" entity structure is one having no specializations and, at most, one aspect hanging from every entity. The result of pruning is a pruned entity structure, which contains fewer aspects and specializations than the original and therefore specifies a smaller family of alternative models than the latter. Ultimately, pruning terminates in a pure entity structure that specifies the synthesis of a particular hierarchical model. An example of a pruned entity structure is shown in Fig. 4(c), where FIFO-BUF, corresponding to the first-in first-out queueing discipline

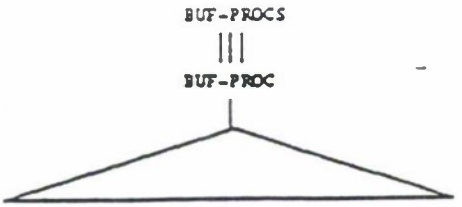


Fig. 5. System entity structure multiple entity. (Same substructure as in Fig. 4(b).)

has been selected as the type of buffer desired. Note that the specialization FIFO-BUF replaces its parent, BUFFER, in all occurrences of the latter in coupling (and other) specifications.

To automatically construct a simulation model in DEVS-Scheme, we apply the "transform" operation to a pruned entity structure. Transform retrieves from the model base those models that correspond to the entities in the pruned entity structure, and then synthesizes them into a simulation model for the BUF-PROC (Fig. 4(d)). The atomic DEVS models for the FIFO-BUF, LIFO-BUF, and PROCESSOR must be stored in the model base prior to the transform operation. Details of the transform operation will be described in Section VI-C.

B. Multiple Entity

The ability to specify multiple entities and multiple decompositions provides a powerful means for representing massively parallel computer architectures which may have different connection topologies—such as broadcast, hypercube, controlled, or cellular. The specification of a parallel processor will be used to illustrate the use and power of multiple entities and multiple decompositions. Assume that the parallel processor is a collection of processing elements with one of the interconnection topologies given in the preceding, each element being a copy of the BUF-PROC that has already been specified. The parallel processor is named BUF-PROCS, which means it is a collection of BUF-PROC's.

We can build the SES for BUF-PROCS by modifying the SES shown in Fig. 4(a). Line (1) of Fig. 4(a) is replaced by a new root entity BUF-PROCS of "multiple entity" type. A multiple aspect is added under the BUF-PROCS, and BUF-PROC is added under the multiple aspect by the operation "add-mult." Once the three items are added, lines (2)–(15) of Fig. 4(a) can be reused without change for the BUF-PROCS specification. The operation "add-mult-couple" specifies the internal coupling scheme for the kernel models in DEVS-Scheme in contrast to "add-coupling" for digraph models. It sets the slot "mult-coup-type" of the item of type "multiple aspect" to one of the subclasses of kernel models [11], [13], [34] in DEVS-Scheme. The resulting SES, which has a multiple entity, is shown in Fig. 5. If broadcast coupling is selected for the multiple decomposition in the pruning process, the operation "transform" will create a broadcast model of BUF-PROCS containing the number of atomic BUF-PROC models specified during the pruning process.



Fig. 6. Construction of multilevel multiple entities.

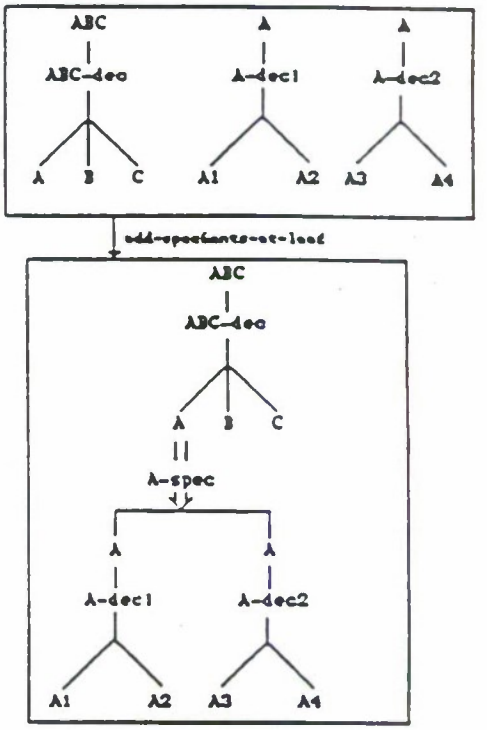


Fig. 7. Addition of pruned entity structure.

C. Hierarchical Model Structuring Operations

Some hierarchical model structuring operations will be discussed to show the power of the SES hierarchical model structuring formalism. Other operations can be found in [13].

The operation "add-sub-entstr" is an extension of the operation "add-item." This operation adds an entity structure under the current item of type "aspect" in the original entity structure. Similarly, the operation "delete-sub-entstr" is an extension of the operation "delete-item." The operation "delete-sub-entstr" deletes the subentity structure under the specified item.

The operation "add-mult" is extended by making arbitrary the number of hierarchy levels for the multiple entities to be added. The operation "add-mult-mult" allows us to specify a hierarchical construction of different kernel models. Fig. 6 shows the operation that results in three levels of hierarchy of the multiple entities AS, BS, and CS. To specify a different coupling type for different kernel models, we use the operation "set-mult-coup-type." An application of the operation "add-mult-mult" to the modeling of a multilevel hypercube architecture can be

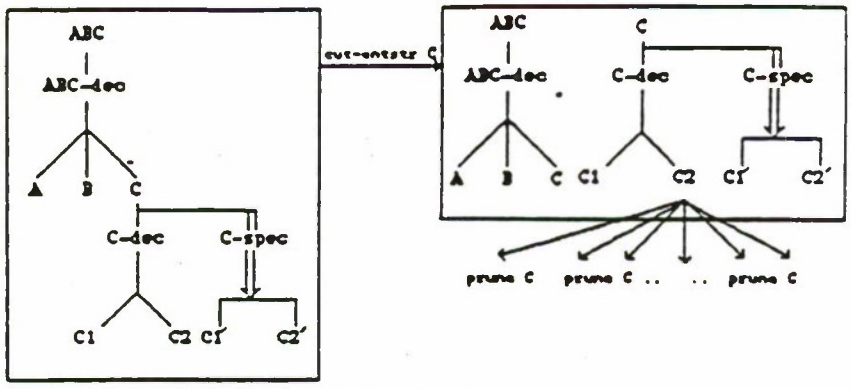


Fig. 8. Cutting subentity structures.

found in [11]. The operation "attach-num-mult-children" attaches the number of components under a multiple entity to the multiple entity.

D. Reuse of Pruned Entity Structures

The entity structure module provides several operations for reusing pruned entity structures. The operation "add-spec&ents-at-leaf" searches the entity structure base to find pruned entity structures whose root names are the same as that of a leaf entity in an entity structure. If any are found, the operation adds a specialization under the leaf entity and adds the pruned entities under the specialization (Fig. 7). The operation "mult-asp → asp" changes a multiple aspect in an entity structure to an aspect by specifying the number of children attached under the aspect.

The operation "cut-entstr" makes a nonleaf entity into a leaf entity by removing the subentity structure beneath the nonleaf entity, constructing a new entity structure and saving it in the entity structure base (we shall describe the entity structure base in Section V-A). The new entity structure has as its root the original nonleaf entity and contains the original nonleaf entity subentity structure (as shown in Fig. 8).

An entity structure for a system can be constructed in a hierarchically distributed manner so that it contains only leaf entities. Each leaf entity in the higher level entity structure may then be a root entity in a lower level entity structure. Hierarchical entity structures may be merged into a single entity structure using the operation "merge-entstr." Merge-entstr searches for entity structures in the entity structure base with the same root names as the leaf entities of the specified entity structure. If such entity structures are present in the entity structure base, they replace the respective leaf entities in the specified entity structure.

V. Facilities in ESP-Scheme

SES construction, copying, and other facilities are provided by ESP-Scheme. The facility "make-entstr" creates an entity structure whose name is the same as its root name except for a prefix "e:". For example, (make-entstr 'system) creates an entity structure e:system with root name system. Since the entity structure has only the root entity system, items can be added as required to construct the desired entity structure. The facility "delete-entstr" deletes an existing entity structure.

The facility "copy-entstr" copies one entity structure to another entity structure. For example, (copy-entstr e:system 'new-system) creates an entity structure named e:new-system that has the same structure (e:system). The facility copies a list of items and a list of branches from the original entity structure and constructs a new entity structure.

A list of all entity structures is maintained by an entity structure manager (described in Section VI-B). All facilities that create or delete entity structures must report to the manager the creation and/or deletion of entity structures. For example, the facility "rename-entstr" asks the manager to delete the original entity structure from, and add the renamed entity structure to, the entity list.

VI. Entity Structure Base/Model Base Management

System entity structures represent structural knowledge about systems. The system entity structures are saved in a entity structure base (ENBASE) for later use. To do so, we store the entity structure created in the current Scheme environment in a directory called ENBASE on an external storage device such as a disk for later use.

A. Entity Structure Base

We have implemented ESP-Scheme so that it can save entity structures in, and retrieve them from, the ENBASE directory by using two facilities, "save-entstr" and "load-entstr." The facility "save-entstr" saves an entity structure or a pruned entity structure into the ENBASE directory by storing a pair consisting of a list of items and a list of branches for the entity structure in the form of a disk file. A file name in the ENBASE directory, corresponding to the entity structure, is the same as its root entity name except for the extension of the file name, which can be either ".e" for the entity structure or ".p" for the pruned entity structure. The facility "load-entstr" searches for a

file corresponding to an entity structure in the ENBASE directory, retrieves the items list and branches list for the corresponding entity structure, and constructs the entity structure.

B. Entity Structure Manager

Entity structure manager (ESM) module manages all of the system entity structures in the current environment and/or in the ENBASE directory. The ESM module has three local variables. The first is a list of entity structures either in the ENBASE or in the current environment. The second is a list of pruned entity structures either in the ENBASE directory or in the current environment. The third is a list of both entity structures and pruned entity structures in the current environment. The operations in ESM include showing entity structures, adding entity structures, and deleting entity structures.

The operation "show-all-ens" shows all entity structures in the current environment and/or in the ENBASE directory. The operation shows the entity structures and pruned entity structures in the three separate lists described in the preceding. The operation "add-entstr" adds an entity structure to the ESM list whenever an entity structure is created or renamed by the facilities described in Section V. The "delete-entstr" operation deletes an entity structure from the list in the ESM.

The initialization routine of ESP-Scheme initializes the ESM when ESP-Scheme is loaded. The initialization includes searching the ENBASE directory and building entity structures lists (described previously) so that the user may use the entity structures present in the current environment or retrieve some from the ENBASE directory as required. The current ENBASE directory can be moved from one place to the other as requested by the user by using "change-dir." Any change in the ENBASE directory results in the reinitialization of the ESM.

C. Transforming into DEVS Models

A pruned entity structure is synthesized into a simulation model by the transform operation. As transform visits each entity in the pruned entity structure, it calls upon a retrieval process that searches the model base (MBASE) directory for a model corresponding to the current entity. If one is found, it is used, and transformation of the entity subtree is aborted. The retrieval process proceeds by evaluating retrieval rules. The retrieval rules consist of condition/action productions and conflict resolution rules.

Prior to searching for a model, the name of the current entity is examined. If the current entity name is segmentable into a base name and a nontrivial extension (the extension must start with numbers or "&"), the base name is used as the entity name for the retrieval process.

One rule for searching for a model that corresponds to the current entity is to first look in the working memory, then in the MBASE directory, and finally, if the current entity is a leaf, in the ENBASE directory. If more than one rule condition is satisfied when evaluated, conflict resolution is used to fire only one rule. The conflict resolution strategy is context specificity—the rule with the most specific condition(s) is fired. Details of retrieval rules and conflict resolution rules are available in [13].

If a pruned entity structure is found in the ENBASE directory during the searching process, a transform is invoked and executed in a separate Scheme environment so as not to interfere with the current environment. Each recursive invocation can occur in a leaf entity only.

The integration of the ESP-Scheme model base management system into the DEVS-Scheme simulation environment is demonstrated by the following simulation study.

VII. Simulation of the Registration System via Phone (RSVP) System at the University of Arizona using DEVS-Scheme

A registration system via phone (RSVP) has been recently installed at the University of Arizona. There have been many complaints and concerns as to whether the RSVP system in its current configuration is adequate to handle the needs of the University of Arizona. Additionally, there has been an expressed desire to incorporate some form of advertising such as student yearbooks, student health insurance, and others into the system [4].

The following provides an overview of the simulation study of the RSVP system using the discrete event system specification (DEVS)-Scheme object-oriented modeling and simulation environment described earlier.

A. Objective of the Simulation Study

The objective of the project was to assist the University Center for Computing and Information Technology by providing a realistic simulation of the existing RSVP system and then investigate the impact of adding advertising to the system.

B. Specific Issues Addressed by the Study

1) Number of telephone lines required to support the system. Currently there are 32 lines by which students can access the system. The University is considering a request to increase the number of lines to improve the system's student registration throughput.
2) Effects of the length of time to perform a transaction. This issue will address the effect of adding advertising onto the system after a student's transaction is completed.

C. Measures of Performance

The primary performance measures are the grade of service provided, utilization rate of the lines, and the student registration throughput. The grade of service is a ratio of call completions to call attempts. The utilization rate is a ratio of the time the lines are being used to the

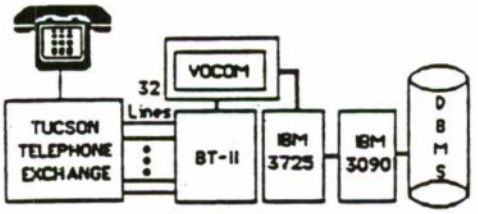


Fig. 9. RSVP block diagram.

total time available for use. The throughput is measured by the number of students who can be registered per unit of time.

VIII. RSVP System Description

The RSVP system consists of a Perceptron VOCOM-1 Package, an IBM 3725 communications processor which serves as a terminal controller, and an IBM 3090 mainframe computer. The VOCOM-1 package consists of two major components: a BT-II unit and a VOCOM unit. The BT-II unit, based upon a PDP-11 minicomputer with a real-time operating system, answers the phones (up to 32 at a time), provides the voice feedback to the student, and provides a translation from the touchtone phone inputs into an ASCII format. The VOCOM unit is primarily a VT-100 terminal with a tape drive which displays system operating status and statistics and provides a translation of the ASCII to EBCDIC. The IBM 3725 communications processor acts as a terminal controller and treats the VOCOM-1 package as an IBM 3271 terminal. The RSVP system software runs on the IBM 3090 mainframe computer. A block diagram of the system is shown in Fig. 9.

IX. RSVP Simulation Study Procedures

The experimental procedures followed during the course of this study were as follows: data collection, data analysis, development of assumptions, model construction, model validation, simulation experiments, and simulation output analysis. These are the same procedures that should be used in any simulation project. This simulation example will focus on the assumptions, model construction, model validation, simulation experimental procedures, and the simulation output analysis.

A. Assumptions

The assumptions made during the simulation study are as follows.

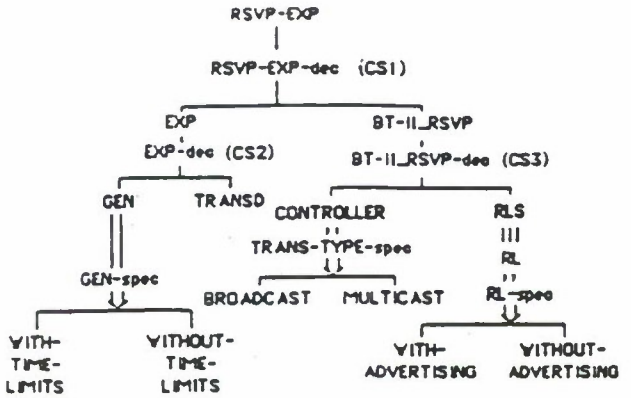
1) The response time of the IBM terminal controller and mainframe are not a consideration, since the system has submillisecond response time. Additionally, the system is designed to support approximately 5000 terminals in the transaction mode, and only 300 terminals are connected at the present time.
2) System failure was not considered, since there is only one system. Thus, if the system fails, it is the same as not having the system.
3) Single-line failure was also not considered, there was not any available data indicating the vidual line failure rate.
4) All calls will be at least 30 s in length. The B does not detect a disconnection unless the c terminated by entering the transaction terminat code or times out. If a transaction termination c is not received within 15 s of the last input, system will generate a message and then wait other 15 s. The system will then disconnect the l after the second 15-s wait if no input is receive
5) Each registration line simulated requires a sep random number stream. This provides a better r resentation of the randomness in the call le generation.
6) The data for calls being answered will be collec upon completion of the call rather than at the t the call was answered. This assumption will a the reduction of messages passed between the m els.
7) The call lengths will be determined by each rather than by the call generator. This saves sim tion run time, since a call length will not need to determined for the call attempts that go u swered.
8) Advertising will increase each call length by 15 The length of the advertising is based upon experience of a broadcast engineer.

B. Model Construction

Model construction in the DEVS-Scheme/ESP-Sche environment consists of three subactivities: specificati of the model structure, specification of the model be ior, and synthesis of a simulation model [13]. The st ture of the system is represented by an SES. The behavi of the system is represented by the models in the m base. The simulation model synthesis is accomplished the transform operation described earlier.

1) Model Structure: To generate inputs to the RS system and to measure its performance, an experime frame was used. An experimental frame is coupled mod composed of atomic-models that are used to generat inputs, observe outputs, and provide control in ac dance with the desired experimental conditions. The S coupling the RSVP and experimental frame is shown Fig. 10. There were two specializations for the contro implementation: broadcast and multicast. The broadc specialization allows all registration lines to receive th input message. The multicast specialization specifie designates which registration line is to receive the inp The coupling schemes designated by {CS1}, {CS2}, an {CS3} in Fig. 10 show the external input, external outpu and internal port connections of the models.

Our study focused on the performance of the BT registration system, with and without advertising and als with a varied number of registration lines. To obtain particular model structure for simulation required the



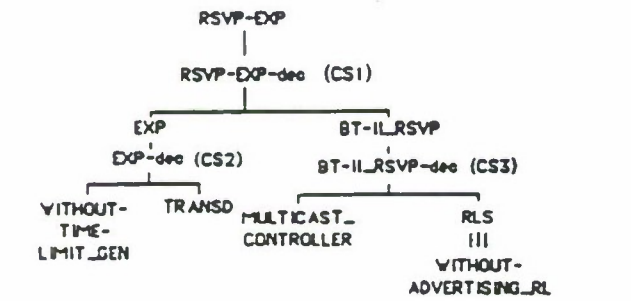
(CS1) = ((EXP.out, BT-IL_RSVP.in)
(BT-IL_RSVP.out, EXP.in))

(CS2) = ((EXP.in, TRANSD.done)
(GEN.out, EXP.out)
(GEN.out, TRANSD.attempt))

(CS3) = ((BT-IL_RSVP.in, CONTROLLER.in)
(CONTROLLER.assign, RLS.assign)
(RLS.done, CONTROLLER.done)
(CONTROLLER.done, BT-IL_RSVP.out))

GEN = Generator
TRANSD = Transducer
RLS = Registration Lines
RL = Registration Line

Fig. 10. Experimental frame RSVP system entity structure.



(CS1) = ((EXP.out, BT-IL_RSVP.in)
(BT-IL_RSVP.out, EXP.in))

(CS2) = ((EXP.in, TRANSD.done)
(WITHOUT-TIME-LIMIT_GEN.out, EXP.out)
(WITHOUT-TIME-LIMIT_GEN.out, TRANSD.attempt))

(CS3) = ((BT-II_RSVP.in, MULTICAST_CONTROLLER.in)
(MULTICAST_CONTROLLER.assign, RLS.assign)
(RLS.done, MULTICAST_CONTROLLER.done)
(MULTICAST_CONTROLLER.done, BT-IL_RSVP.out))

GEN = Generator
TRANSD = Transducer
RLS = Registration Lines
RL = Registration Line

Fig. 11. RSVP-EXP pruned entity structure.

of the pruning operation. Our initial simulation model was constructed by selecting WITHOUT-TIME-LIMITS specialization from GEN-spec, the MULTICAST specialization from TRANS-TYPE-spec, and WITHOUT-ADVERTISING specialization from the RL-spec during the pruning operation. The number of registration lines for a particular simulation model was specified during the running process. Fig. 11 illustrates the pruned SES.

2) Model Behavior: To model the behavior of the RSVP system required the implementation of four atomic models. Two of the models were required for the experimental frame (GEN and TRANSD), and two models were

```
;;Creates an atomic model ADV_RL and an attached simulator s:ADV_RL
(make-pair atomic-models 'ADV_RL)
(send ADV_RL def-state '(
                                ;;state-variables
                                 ta-time  ; length of call
                               extra) )      ; advertising time
;;Establishes the state variables for ADV_RL, it already has state
;;variables sigma and phase by default

(send ADV_RL set-s
      (make-state
                                ;; initialization of state variables
                                    'sigma 'inf
                           'phase 'passive
                           'ta-time '()
                                    'extra 0))  ; without advertising
(define (ext-ADV_RL s e x)
  (let (
        (call-length  (content-value x))  ; value of incoming message
                                                    ; is the call length
        )
    (if (< call-length 0.5)
        (set! call-length 0.5))
    (set! (state-ta-time s) (+ call-length (state-extra s)))
    (hold-in 'busy  (state-ta-time s))
                                ;; (set! (state-sigma s) state-ta-time s)
                                ;; (set! (state-phase s) 'busy)
  ); let
)
(define (int-ADV_RL s)
  (passivate)
)
(define (out-ADV_RL s)
  (make-content 'port 'done
          'value (list (state-name s)
                       (state-ta-time s)) )
)

;; Attach external and internal transition functions and output
;; function to the model ADV_RL
(send ADV_RL set-ext-transfn ext-ADV_RL)
(send ADV_RL set-int-transfn int-ADV_RL)
(send ADV_RL set-outputfn out-ADV_RL
```

Fig. 12. ADV_RL atomic-model DEVS-scheme code.

required for the RSVP system (RL and MULTI-CAST_CONTROLLER).

GEN and TRANSD were required to generate the call attempts and to collect output data, respectively. The atomic model RL represents the behavior of a single telephone registration line and the atomic model MULTICAST_CONTROLLER represents the behavior of the BT-II when selecting an available telephone registration line. The DEVS-Scheme implementation of the atomic model RL without advertising is shown in Fig. 12. More detail on model construction and simulation can be found in [34].

The simulation of the registration procedure is initiated by the generator outputting a call attempt to the BT-II_RSVP system and TRANSD. The BT-II_RSVP system outputs the call attempt to the MULTICAST_CONTROLLER, which in turn passes the call attempt to a registration line if available, otherwise the call attempt is ignored. When a call is completed, the registration line sends a call completion notice to the MULTICAST_CONTROLLER. The MULTICAST_CONTROLLER then passes the completion notice to the BT-II_RSVP system, which in turn sends the completion notice to EXP. EXP completes the simulation of one registration by sending the completion notice to the TRANSD.

Each atomic model constructed was tested independently using the facilities provided by DEVS-Scheme prior

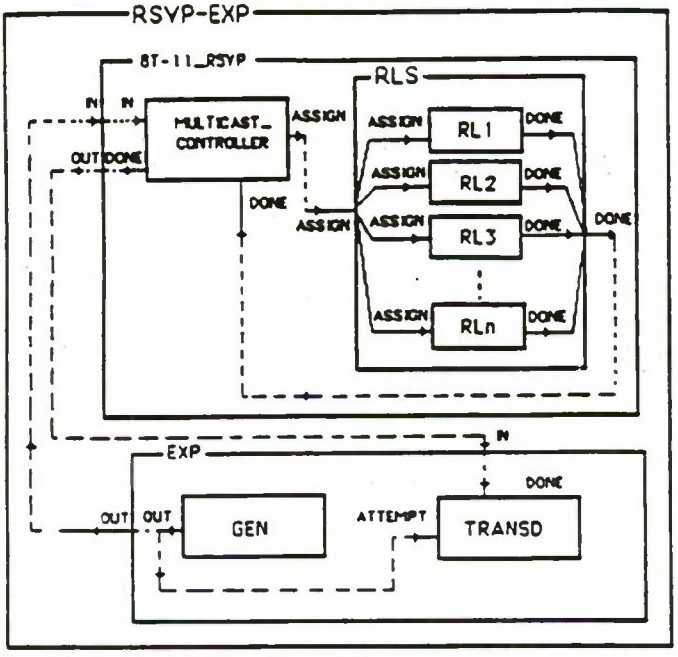


Fig. 13. Transformed RSVP model.

to storing the atomic model in the model base. This facilitates the debugging of the atomic models and any coupled model of which they are a component.

3) Simulation Model Synthesis: The simulation model of the RSVP system was synthesized using the transform operation described in Section VI-C of this paper. The transformed RSVP system model is as shown in Fig. 13.

C. Model Validation

The models were validated by comparing the output generated using 23 lines, with a stationary Poisson arrival rate of 396 attempts per hour, the call lengths normally distributed ($m = 3.0667$ min, $s = 0.55375$ min) to the predicted data developed by U.S. West Communications for CCIT. The attempts per hour and mean call length used for the validation run were the same attempts per hour and mean call length used by U.S. West in their calculations. The standard deviation was obtained from analyzing the data from August 12–27, 1988. This data included the data used by U.S. West to calculate the predicted performance of the RSVP system. The predicted grade of service was 90%; our simulation runs had an average grade of service of 87.35%.

The simulation runs were also replicated with 32 lines, using the same arrival rate and call lengths and compared to actual system data. The actual system performance indicated that the grade of service was 100%. Our simulation runs had an average grade of service of 97.95%. Simulation output for two of the validation runs is shown in Fig. 14.

If the initial 30 minutes are discarded, the grade of service for 23 lines and 32 lines is 89.04% and 100%. The low completion rates for the first 30 minutes is a result of collecting call-answered data upon completion of the call

23 Lines Time	% Calls Completed	32 Lines Time	%Calls Completed
Run # 1		Run # 1	
30	0.809	30	0.882
60	0.859	60	0.994
90	0.949	90	1.000
120	0.916	120	1.015
150	0.844	150	1.004
Run # 2		Run # 2	
30	0.802	30	0.888
60	0.825	60	0.973
90	0.897	90	1.026
120	0.973	120	0.970
150	0.860	150	1.043

Fig. 14. Model validation data.

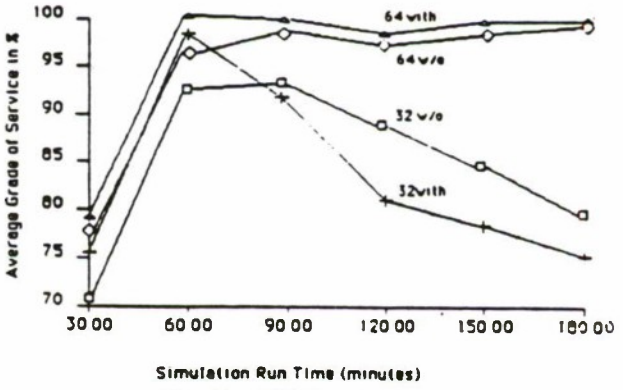


Fig. 15. RSVP performance.

rather than at the time the call was answered.

In either case, the difference between the simulation output, the predicted performance, and actual syster performance was less than 3%. Thus our model is consid ered to be a valid model of the University of Arizona RSVP system.

D. Simulation Experimental Procedure

The call attempt rate was modeled using a nonstation ary Poisson process to reflect the change in the cal attempts per hour. The simulation experimental proce dure consisted of replicating each simulation run a mini mum of three times from 7:00 AM to 10:00 AM. Thi allowed the system to reach the maximum call attempt rate, thus testing the system under maximum load. The call lengths were modeled using a normal distribution with $m = 3.56945$ and $s = 1.23502$ minutes.

Four different system configurations were simulated: the current 32-line system, a 32-line system with advertis ing, a 64-line system without advertising, and a 64-line system with advertising.

Each replication of a particular configuration used a different set of random number streams. However, each configuration used the same set of random number streams for performance comparison purposes.

E. Simulation Output Data

A representative sample of the simulation data obtained from the RSVP model is as shown in Fig. 15. Data

was collected at 30-min intervals for a simulation period of 180 minutes for each replication.

F. Output Data Analysis

The current 32-line configuration of the RSVP system exhibited an acceptable grade of service between 60 and 90 min of simulation time and then deteriorated rapidly as the call attempt rate began to increase. When a 15-s advertisement was added to the system, the grade of service deteriorated from a high of 98% at 60 min of simulation to a low point of 77% at 180 minutes.

G. RSVP Simulation Conclusions and Recommendations

The current system configuration is not adequate to meet the current registration demand as it did not sustain a grade of service above 90%. If advertising is to be added to the current 32-line configuration, the grade of service will deteriorate even more. The 64-line configuration had a grade of service above 90%, and if advertising were to be added, there would be no appreciable change in the grade of service.

X. Conclusion

We have described an implementation of system entity structure in Scheme, ESP-Scheme, which serves as a model base management system for DEVS models. Also, we illustrated the use of these concepts by modeling and simulating the University of Arizona registration system via phone (RSVP).

The utility of ESP-Scheme also has been demonstrated in the construction of several complex hierarchical models for computer networks [24], advanced computer architectures [13], and multirobot systems [30], [34].

The system entity structure construct has thus been shown to provide a workable foundation for model base management in simulation environments and workbenches. This is the first demonstration, to the authors' knowledge, of a knowledge-based system that addresses the needs of a model repository as expressed in [9], i.e., to provide a sharable repository of models and a means of assisting users to synthesize models to satisfy the objectives of the current analytic task. ESP-Scheme provides a system-theory-based structuring of the model base and a goal-based means of constructing models from reusable components. The hierarchical, modular simulation modeling capability of the underlying DEVS-Scheme environment is critical to the workability of this approach.

The aforementioned DEVS-Scheme/ESP-Scheme knowledge-based framework serves as our vehicle for research in knowledge-based system design using variant families of design models [19]–[22].

Research is continuing to further explore, and respond to, the needs of model base management. Currently, our efforts are concentrating on maintaining model base coherence in the face of the multiplicity of related models expressed in various formalisms at various levels of abstraction.

References

[1] O. Balci, "Requirements for model development environments," *Comp. Oper. Res.*, vol. 13, no. 1, pp. 53–67, 1986.

[2] D. W. Balmer and P. J. Paul, "CASM—The right environment for simulation," *J. Oper. Res. Soc.*, no. 37, pp. 443–452, 1986.

[3] D. Belogus, "Multifaceted modelling and simulation: a software engineering implementation," Ph.D. dissertation, Weizmann Inst. of Science, Rehovot, Israel, 1985.

[4] E. R. Christensen, C. Lee, and F. DiMaggio, "DEVS-Scheme simulation of the University of Arizona registration system via phone (RSVP) system," class project for MIS 521a, University of Arizona, Tucson, AZ, 1988.

[5] A. I. Concepcion and B. P. Zeigler, "DEVS formalism: A framework for hierarchical model development," *IEEE Trans. Software Engr.*, vol. SE-14, no. 2, pp. 228–241, Feb. 1988.

[6] P. K. Davis, "Applying artificial intelligence techniques to strategic-level gaming and simulation," in *Modelling and Simulation Methodology in the Artifical Intelligence Era*, M. S. Elzas, T. I. Oren, and B. P. Zeigler, Eds. Amsterdam: North Holland, 1986.

[7] M. S. Elzas, "The applicability of artificial intelligence techniques to knowledge representation in modelling and simulation," in *Modelling and Simulation Methodology in the Artificial Intelligence Era*, M. S. Elzas, T. I. Oren, and B. P. Zeigler, Eds. Amsterdam: North Holland, 1986, pp. 19–40.

[8] ———, "Relations between artificial intelligence environments and modelling & simulation support systems," in *Modelling and Simulation Methodology in the Artificial Intelligence Era*, M. S. Elzas, T. I. Oren, and B. P. Zeigler, Eds. Amsterdam: North Holland, pp. 61–78, 1986.

[9] J. B. Gilmer and I. Kameny, "SIMTECH 97: Report of the Workbench Working Group," *Pinduor*, vol. 22, no. 4, 1989.

[10] J. O. Henriksen, "The integrated simulation environment," *Oper. Res.*, vol. 31, pp. 1053–1073, 1983.

[11] T. G. Kim and B. P. Zeigler, "The class kernel-models in DEVS-Scheme: A hypercube architecture example," *ACM SIMULETTER*, vol. 19, no. 2, June 1988.

[12] ———, "The DEVS formalism: Hierarchical, modular system specification in an object-oriented framework," in *Proc. 1987 Winter Simulation Conf.*, Atlanta, GA, 1987, pp. 559–566.

[13] T. G. Kim, "A knowledge-based environment for hierarchical modelling and simulation," Ph.D. dissertation, Dept. of Electrical and Computer Engr., University of Arizona, Tucson, AZ, 1988.

[14] P. Klahr, "Expressibility in ROSS, an object-oriented simulation system," in *Artificial Intelligence in Simulation*, G. C. Vansteenkiste, E. J. H. Kerckhoffs, and B. P. Zeigler, Eds. San Diego, CA: SCS Publications.

[15] K. J. Murray and S. V. Sheppard, "Automatic synthesis using automatic programming and expert systems techniques toward simulation modeling," in *Proc. Winter Sim. Conf.*, 1987, pp. 534–543.

[16] T. I. Oren, "Artificial intelligence and simulation," in *Artificial Intelligence in Simulation*, G. C. Vansteenkiste, E. J. H. Kerckhoffs, and B. P. Zeigler, Eds. San Diego, CA: SCS Publications, 1986, pp. 3–8.

[17] ———, "Bases for advanced simulation: Paradigms for the future," in *Modelling and Simulation Methodology: Knowledge Systems Paradigms*, M. S. Elzas, T. I. Oren, and B. P. Zeigler, Eds. Amsterdam: North Holland, 1989, pp. 29–44.

[18] Y. V. Reddy, M. S. Fox, N. Husain, and M. McRoberts, "The knowledge-based simulation system," *IEEE Software*, pp. 26–37, Mar. 1986.

[19] J. W. Rozenblit and B. P. Zeigler, "Concepts of knowledge based system design environments," in *Proc. 1985 Winter Simulation Conf.*, San Francisco, CA, 1985, pp. 223–231.

[20] J. W. Rozenblit and Y. Huang, "Constraint-driven generation of model structures" in *Proc. 1987 Winter Simulation Conf.*, Atlanta, GA, 1987, pp. 604–611.

[21] J. W. Rozenblit, "A conceptual basis for integrated, model-based system design," technical report, Dept. of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, Jan. 1986.

[22] ———, "A conceptual basis for model-based system design," Ph.D.

dissertation. Dept. of Computer Science, Wayne State University, Detroit, MI, 1985.

[23] S. Ruiz-Mier and J. Talavage, "A hybrid paradigm for modeling of complex systems," in *Artificial Intelligence, Simulation and Modelling*, L. A. Widman, K. A. Loparo, and N. Nielsen, Eds. New York: Wiley, 1989, pp. 381–395.

[24] S. Sevinc and B. P. Zeigler, "Entity structure based design methodology: A LAN protocol example," *IEEE Trans. Software Eng.*, vol. SE-14, no. 3, pp. 375–383, Mar. 1988.

[25] R. H. Sprague and E. D. Carlson, *Building Effective Decision Support Systems*. Englewood Cliffs, NJ: Prentice Hall, 1982.

[26] T. Thomasma and O. M. Ulgen, "Hierarchical, modular simulation modelling in icon-based simulation program generators for manufacturing," in *Proc. Winter Simulation Conf.*, San Diego, 1988, pp. 254—262.

[27] T. Winograd, "Frame representations and the declarative/procedural controversy," in *Representation and Understanding: Studies in Cognitive Science*, D. G. Bobrow and A. Collins, Eds. New York: Academic Press, 1976.

[28] A. W. Wymore, *A Mathematical Theory of Systems Engineering: The Elements*. New York: Wiley, 1967.

[29] L. A. Zadeh and C. A. Desoer, *Linear System Theory, The State Space Approach*. New York: McGraw Hill, 1963.

[30] B. P. Zeigler, F. E. Cellier, and J. W. Rozenblit, "Design of a simulation environment for laboratory management by robotic organizations," *J. Intell. Robotic Syst.*, (in press) 1989.

[31] B. P. Zeigler, "Hierarchical, modular discrete-event modelling in an object-oriented environment," *Simulation*, vol. 50, no. 5, pp. 219–230, 1987.

[32] ____, "Knowledge representation from Minsky to Newton and beyond," *Appl. Artif. Intell.*, vol. 1, pp. 87–107, 1987.

[33] ____, *Multifaceted Modelling and Discrete Event Simulation*. London, UK and Orlando, FL: Academic Press, 1984.

[34] ____, *Object-Oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems*. New York: Academic Press, 1990.

[35] G. Zhang and B. P. Zeigler, "The system entity structure: Knowledge representation for simulation modeling and design," in *Artificial Intelligence, Simulation and Modelling*, L. A. Widman, K. A. Loparo, and N. Nielsen, Eds. New York: Wiley, 1989, pp. 47–73.



Tag Gon Kim (S'84–M'88) received the B.S.E.E. and M.S.E.E. degrees from Pusan National University, Korea, and Kyungpook National University, Korea, in 1975 and 1980, respectively. He received the Ph.D. degree in electrical engineering from the University of Arizona, Tucson, AZ, in 1988.

From 1987 to 1989, he worked as a Research Staff Engineer in the Environmental Research Lab of the University of Arizona. Since August 1989, he has been an Assistant Professor in the Telecommunications and Information Sciences Lab, Department Electrical and Computer Engineering, The University of Kans Lawrence, KS. He also held a faculty position in the Department Electronics, National Fisheries University of Pusan, Pusan, Korea. research interests include artificial intelligence for advanced simulati methodology, computer systems modeling, and object-oriented softwar environment.

Dr. Kim is a member of the ACM, the AAAI, the SCS, and Kappa Nu.



Chilgee Lee received the M.S.E.E. degree fr Arizona State University and the B.S.E.E. de gree from Sungkyunkwan University, Seoul, K rea. He is a Ph.D candidate in the Artific Intelligence and Simulation Group, Departm of Electrical and Computer Engineering, at University of Arizona.

His research interests are in the areas artificial intelligence, modeling and simulati computer architectures, and parallel processi



Eric R. Christensen (M'83) received t M.S.E.E. degree from the Air Force Institute o Technology, and the B.S. degree from th United States Military Academy.

He is a Major in the U.S. Army Signal Co and a Ph.D. candidate in the AI and Simulation Group, Department of Electrical and Computer Engineering, University of Arizona. His search interests are in the areas of artific intelligence, modeling and simulation, co... mand, control, and communications system design.

Mr. Christensen is a member of the AFCEA.



Bernard P. Zeigler received the Ph.D. degree computer/communication science from t University of Michigan, Ann Arbor, in 19 and the preceding degrees from McGill Univer sity, Montreal, PQ, Canada, and from MIT Cambridge.

He is a Professor in the Department of El trical and Computer Engineering at the Univer sity of Arizona. He is the author of *Multi-facetted Modelling and Discrete Event Simulation*, (Academic Press, 1984) and *Theory Modelling and Simulation* (Wiley, 1976). His research interests inclu artificial intelligence, distributed simulation, and expert systems for simulation methodology.

Horrigan Analytics
1460 North Sandburg Terrace
Chicago, Illinois 60610

THE "CONFIGURATION PROBLEM" AND CHALLENGES FOR AGGREGATION

Timothy J. Horrigan

Prepared For

Conference on Variable-Resolution Combat Modeling
Washington, D.C.
5-6 May 1992

27 June 1992

document HAS 92-5

TABLE OF CONTENTS

	Page Number
NOTE.	iii
I. CONFIGURATION AND MATHEMATICAL MODELS OF COMBAT.	1
II. BASIC CONCEPTS AND DEFINITIONS OF CONFIGURAL THEORY.	3
1. The Attacker-Defender Configuration Defines the Structure of Configural Mathematical Models of Combat	3
2. Free Encounters and Aggregation Simplify the Mathematics of Target-Weapon Encounters but Weaken or Destroy Configuration	4
APPENDIX. A READING COPY OF THE VU-GRAPHS FOR "THE 'CONFIGURATION PROBLEM' AND CHALLENGES FOR AGGREGATION"	
Principal Points to Be Made	A-1
Illustrations Based on a Configuration Comprising Two Target-Shooter Pairs.	A-14
Illustrations Based on an Idealized, Simplified Air Defense Weapons System Defending an Extended Area	A-21
Conclusions	A-42

NOTE

This document contains the substance of the talk "The 'Configuration Problem' and Challenges for Aggregation" that was prepared for and presented at the Conference on Variable-Resolution Combat Modeling in Washington, D.C., on 5-6 May 1992. Because funds were severely limited, a paper covering the material presented could not be prepared. This document comprises introductory text that presents the basic concepts of configural theory and an appendix with reading copies of the vu-graphs used in the talk. It also includes reading copies of several vu-graphs that, because of a last-minute schedule change, were not presented. The reading copies differ from the projection copies in that additional explanatory text in vu-graph or caption format is included.

The introductory text notes that excluding configuration in mathematical models of combat can result in overstating casualty production by factors of two to five or more and weapons effectiveness by similar and greater factors. The combat situations discussed in the talk focus on stochastic variation that is understated or excluded as a result of excluding configuration, especially as a consequence of aggregation. Casualty production is not overstated at all in a number of situations discussed because they are intended to illustrate the great stochastic variation that is masked by aggregation, even in situations in which the average number of casualties is unaffected. In unclassified material that considers the consequences of excluding configuration more broadly, the overstatements of casualty production often exceed a factor of five. In combat situations considered in classified reports that address the effectiveness of proposed or developmental weapons, the overstatements of effectiveness sometimes exceed a factor of ten.

The concepts and mathematics of configural theory are results of research performed under a series of contracts supported by the Office of Naval Technology of the Office of the Chief of Naval Research and the Naval Sea Systems Command, Mine Warfare Systems Project Office (PMS 407). The mathematical relationships and the computational procedures that are used for the illustrations based on simplified, idealized air defense situations are products of research supported by the Office of Naval Technology.

I. CONFIGURATION AND MATHEMATICAL MODELS OF COMBAT

The customary conceptualization in defense analysis of the relationship between the behavior of weapons in use in combat and their individual characteristics, which is implicit in the derivative measures of effectiveness and in the associated mathematical models of combat, deterministic and stochastic alike, tacitly excludes a fundamental element of combat. That element, which is termed *configuration*, is the deployment or situation in space of the individual targets and the individual weapons of the attacker and the defender, each at a particular position and in a particular state. More precisely, configuration is the mathematical embodiment of the inseparability in the course of combat of real targets and real weapons from their locations in space.

Configuration not only very strongly affects the effectiveness of weapons in use in combat, but also very strongly affects analytical assessments of that effectiveness. In particular, because the customary quantification of the behavior of weapons in use in combat excludes configuration, the derivative mathematical models systematically make the casualty production of weapons in multiple-target, multiple-weapon encounters appear to be greater than it can be, often by factors of two to five or more. Correlatively, weapons effectiveness is overstated by similar and sometimes greater factors, the pace of combat is artificially increased, and weapons requirements are understated.

Although configuration in that technical sense is not in the military lexicon, the concept is of recognized military importance and is fundamental to strategy and tactics — for example, which weapons and combatants are placed where in a reverse slope defense, which ships are placed where in a carrier battle group, which aircraft are placed in what wave and with what formation in an air strike. Positioning weapons to exploit their characteristics, compensate for their weaknesses, and reinforce their combined effects can make a decisive difference. However, despite the recognized importance of the concept in military planning, its importance for the mathematical representation of the individual one-target, one-weapon encounters that underlie casualty production is mainly not realized. That is reflected in the widespread use of aggregation, which weakens or excludes configuration, in mathematical models of combat.

In developing mathematical models of combat, numerous fundamental decisions concerning what is to be included and how it is to be included are determined mainly by what the developers believe is necessary and what they believe is a suitable mathematical representation. What is believed in that respect is a theory or, more precisely, the product of a theory. That theory may be intuitive and implicit or formal and explicit. In science, "theory" denotes the abstract elements of a subject and the relationships among them that must be discerned and understood in order to explain and predict. It is usually mathematical and explicit. In defense analysis in general and in combat modeling in particular, there is little such theory. Even though mathematical terminology and representations are used extensively, existing theory is mainly intuitive and implicit. Moreover, the mathematical representation even of important factors in major

models is usually inadequate because the customary conceptualizations usually exclude configuration in important ways. In Lanchester theory and derivative mathematical models, for instance, configuration is excluded from target-weapon encounters.

The following section defines and briefly discusses the basic concepts of configural theory and relates aggregation to them. The substance of the talk given at the Conference on Variable-Resolution Combat Modeling is given in the appendix, which has reading copies of the vu-graphs that were prepared for it. Its principal point is that, to realize more fully the potential of modeling and simulation to improve the effectiveness of weapons, tactics, and training, models and simulations must present a true picture of combat to their users. To do that, as the combat situations examined in the appendix forcefully show, they must be stochastic and incorporate configuration. Most existing models cannot. In particular, Lanchester models or expected-value models cannot because such models, among other failings, have a mathematical structure that excludes configuration as well as randomness.

II. BASIC CONCEPTS AND DEFINITIONS OF CONFIGURAL THEORY

Encounters in combat are generally multiple-target, multiple-weapon encounters. Actual encounters are always configured: Each target is always of a particular kind and, at each instant of time, is at a particular position and in a particular state, and each weapon is always of a particular kind and, at each instant of time, is at a particular position and in a particular state. Furthermore, the states of the targets and of the weapons, which include the particular military role or mission of each, depend upon and constrain their locations. The targets and the weapons in the course of combat are indeed inseparable from their locations.

The targets and weapons of the attacker as deployed or situated in space, whether by chance or plan, thus define (and in configural theory are defined by) the *attacker configuration*. Similarly, those of the defender define (and in configural theory are defined by) the *defender configuration*. Each entity in such a configuration has a corresponding *encounter region*, the set of locations of an enemy entity at which an interaction can occur with a positive probability. A *configured encounter* between two entities occurs at the epoch at which one entity first enters the encounter region of another. A configured encounter between two configurations occurs at the epoch at which an entity in either configuration first encounters an entity of the other.

The attacker configuration and the defender configuration, or parts of them, may be in relative motion; and elements of each may be in motion relative to their configurations as well. Moreover, both the motion of the attacker configuration and the defender configuration and the relative motion of their mobile elements are usually coordinated or directed by extensive command, control, and communications systems so as to avoid configural disadvantages and to exploit configural advantages. Once an encounter occurs, the attacker configuration and the defender configuration thus interact to form a single entity, the *attacker-defender configuration*, as each simultaneously adapts and reacts to both the assessed strengths and weaknesses of its and the enemy's configuration and to the effects of exchanges of fires on those configurations in accordance with the strategy, tactics, and training that define its military behavior. An *engagement* defines (and in configural theory is defined by) the stochastic development of an attacker-defender configuration.

1. The Attacker-Defender Configuration Defines the Structure of Configural Mathematical Models of Combat

Configuration, principally in the form of the attacker-defender configuration, determines which targets can be encountered by which weapons, in what order, at what ranges, and with what velocities and orientations. The detection probabilities, acquisition probabilities, hit probabilities, and damage probabilities associated with a particular target-weapon encounter, of course, are each a function of the kind, state, and position of the target and the kind, state, and position of the weapon (as well as other variables). Also, the position of a particular target and a particular

weapon relative to each other is a function of the relative position of the attacker and the defender configurations. Consequently, the probability that any particular target is damaged by any particular weapon is a function of the relative position of the attacker and the defender configurations, among other variables, as well as those configurations themselves.

Accommodating the attacker-defender configuration thus has a major mathematical consequence: Numerous stochastic processes and the associated random variables that in combat are probabilistically dependent but in the customary quantifications of weapons effectiveness and in derivative mathematical models are treated as probabilistically independent embody in configural quantifications the necessary dependencies. The target-weapon ranges and relative orientations are a good example: As the attacker configuration is shifted by a random amount relative to the defender configuration (or vice versa), the relative positions of the targets and the weapons change in unison. Hence, randomness in the relative position of the attacker configuration and the defender configuration, regardless of its source (for example, the difference between the actual and expected location of an enemy force), introduces common random components into all the relative target and weapon positions. That also applies to the orientations of the attacker and the defender configurations at the epoch of encounter. Thus, in a multiple-target, multiple-weapon configured encounter, all target-weapon ranges are dependent random variables. Similarly, the relative velocity and the orientations of a target and a weapon that encounter each other are dependent random variables. Because the ranges, velocity, and orientations that define a target-weapon encounter are dependent random variables, random events such as target acquisitions and target kills, which are customarily treated as independent events within their respective classes, are dependent as well.

Multiple-target, multiple-weapon configured encounters are thus very complex mathematically. Indeed, mathematical formulas for weapons effectiveness or casualty production that correctly accommodate configuration (*configural* formulas) usually differ markedly from their customary counterparts, which exclude configuration and, accordingly, are termed *nonconfigural*.

Each target-weapon encounter in a multiple-target, multiple-weapon configured encounter, viewed by itself, is also a configured encounter: a one-target, one-weapon configured encounter. The target-weapon range, relative velocity, orientations, and other pertinent characteristics are all determined by the relative position of the attacker configuration and the defender configuration and the positions of the target and the weapon relative to their respective configurations — that is, by the attacker-defender configuration. Thus, even a one-target, one-weapon configured encounter can be very complex mathematically.

2. Free Encounters and Aggregation Simplify the Mathematics of Target-Weapon Encounters but Weaken or Destroy Configuration

Except in a few simple cases (and then usually unintentionally), analytical models do not address configured encounters, and Monte Carlo

simulations, although they can much more easily accommodate configuration than analytical models, nonetheless typically exclude much of it. Two commonplace simplifications, the free encounter and aggregation, which are fundamental in defense analysis, are the primary sources of the nonconfigural character of combat models and the consequent overstatement of weapons effectiveness and casualty production.

A *free encounter* is a special multiple-target, multiple-weapon encounter in which all the configural random variables that probabilistically define the constituent one-target, one-weapon encounters are independent. Accordingly, any particular one-target, one-weapon encounter within a multiple-target, multiple-weapon free encounter is also a free encounter. Free encounters are rare in combat. They mainly arise as a limiting case of configured encounters. A multiple-target, multiple-weapon configured encounter approaches a free encounter as a limit in combat situations in which the targets and the weapons that constitute the attacker-defender configuration are so widely separated that the probability that any particular target is encountered by more than one weapon and the probability that any particular weapon is encountered by more than one target are essentially zero.

Free encounters have great intuitive appeal and are simple mathematically. Acquisition and kill probabilities that are functions of the respective locations of the target and the weapon, for instance, become constant. The one-target, one-weapon duel that underlies Lanchester theory is an instance that is widely used.

Although a virtue of the intuitively appealing, mathematically simple, free-encounter idealization is that, in practice, it turns difficult mathematical problems (that usually have been neither conceptualized nor formulated and thus have not been identified as such) into simple problems that usually are comparatively easy to solve, its vice is that the resulting, simple problems, aside from their descriptive terminology, have little or nothing to do with combat. Furthermore, the mathematical flaw in the simplification (treating probabilistically dependent events and processes as independent), despite its often great quantitative impact, is subtle. Indeed, free encounters are usually seen as representative of combat rather than as what they are: mainly an atypical, limiting case of the strongly configured encounters that typify combat.

Aggregation in combat modeling is a procedure in which a set of entities that differ in some significant respects is replaced by a set of entities that are identical in those respects. It, too, is a powerful simplifier. In contrast to the free encounter, however, it has generally been recognized not only as an approximation but also as an approximation that in many applications would be better avoided (although that it acts to weaken or exclude configuration, and thereby to overstate weapons effectiveness, is not well known). Nevertheless, even though the consequent error is usually unknown, aggregation is widely used because it is perceived as essential to developing "useful" combat models — that is, models that developers can develop and run within the confines of the theory, algorithms, funding, and computational resources typically available.

Although much aggregation is explicit, the extent of the actual aggregation in those models — and, therefore, the degree to which configuration is further excluded — is usually much greater than is apparent.

In addition to aggregation that is explicitly imposed, aggregation is implicitly imposed whenever dissimilar or similar but not identical entities are treated as identical. For instance, groups of riflemen with different aiming errors engaging identical targets under identical conditions or groups of riflemen with identical aiming errors engaging identical targets that are differently exposed or at different ranges are implicitly aggregated by being postulated to have identical hit probabilities or to be engaging identically exposed targets at identical ranges. Conversely, postulating nonidentical targets to appear identical to their attackers aggregates the targets (even though the model may otherwise treat them as nonidentical — in printouts of casualties, for instance). Similarly, aircraft are aggregated whenever their radar cross sections are postulated to be constant — that is, independent of the aspect angle — or their pilot-dependent kill probabilities are postulated to be equal. More generally, postulating combatants to be identical that differ only in military behavior aggregates the associated combatants. Even the commonplace simplification of replacing random occurrence times or random numbers of rounds expended with average values aggregates the associated combatants. For example, if identical combatants are independently firing upon identical, nonfiring targets under identical conditions, aggregation equates the combatant to complete its mission with the smallest expenditure of ammunition to the combatant with the largest and, consequently, as combat proceeds, a partly full magazine to an empty one.

In Lanchester theory, which provides a major portion of, if not the entire, casualty-production representation used in many large-scale models, the aggregation is nearly total. The only independent variable in the Lanchester equations is time. There are no space variables. The large numbers of entities characteristic of the scenarios those models are intended to address are tacitly postulated to have no spatial distribution at all and, therefore, to form no configuration. Furthermore, the personnel and weapons respectively constituting the various elements of the opposing forces are postulated to be so homogeneous that they are differentiable functions of each other as well as time. By merely representing different entities with different characteristics at different positions as identical entities at those positions (or different positions), aggregation weakens configuration. Lanchester theory, because it further aggregates those entities into a single group or into a sequence of groups within which the individual entities do not even have position, entirely excludes configuration within the groups. As the hypothetical combat situations examined in the appendix illustrate, the overstatements of casualty production, the understatements of weapons required, and the understatements of the variation in casualty production that result from excluding configuration are substantial. In the last situation examined in the appendix, treating a configured encounter as a free encounter overstates casualty production by more than thirty percent, understates the number of weapon emplacements required by almost a factor of two, and understates the variation in casualty production by more than a factor of twelve.

APPENDIX

A READING COPY OF THE VU-GRAPHS FOR

"THE 'CONFIGURATION PROBLEM' AND CHALLENGES FOR AGGREGATION"

PRINCIPAL POINTS TO BE MADE

- ACCOMMODATING CONFIGURATION IS FUNDAMENTAL TO THE MATHEMATICAL REPRESENTATION OF COMBAT

- AGGREGATION PARTLY OR COMPLETELY EXCLUDES CONFIGURATION

- COMBAT MODELS BASED ON CUSTOMARY THEORY
  - — UNDERSTATE THE VARIATION IN THE OUTCOMES OF ENGAGEMENTS
  - — UNDERSTATE WEAPONS REQUIREMENTS
  - — OVERSTATE CASUALTY PRODUCTION

- CONFIGURAL THEORY PROVIDES A MEANS TO REALIZE MORE FULLY THE POTENTIAL OF MODELING AND SIMULATION TO IMPROVE WEAPONS, TACTICS, AND TRAINING

- THEORY — THE CONCEPTS AND THE RELATIONSHIPS AMONG THEM THAT MUST BE DISCERNED AND UNDERSTOOD IN ORDER TO EXPLOIT THE PHENOMENA OF INTEREST; IN PARTICULAR, THE MATHEMATICAL STRUCTURE AND THE ASSOCIATED MATHEMATICAL VARIABLES IT RELATES

- MODEL — A BODY OF MATHEMATICAL FORMULAS, PROCEDURES, AND RULES DERIVED FROM THEORY (I.E., AN ALGORITHM) THAT DEFINES THE COMPUTATIONS NECESSARY TO DETERMINE THE VALUES OF THE VARIABLES OF ONE SET FROM THOSE OF THE VARIABLES OF ANOTHER

"ONE RARELY VISITED [THE MATHEMATICAL WORKING GROUP] WITHOUT HEARING SOME WARNING ABOUT THE MISUSE OF EXPECTED VALUES, THE NEGLECT OF PROBABILISTIC DEPENDENCIES, OR THE IGNORING OF IMPORTANT SPATIAL CONFIGURATIONS IN THE MODELING OF COMBAT. IN FACT, SUCH PROBLEMS AND THEIR IMPLICATIONS FOR THE IMPLEMENTATION OF HIERARCHICAL STRUCTURES WERE OF SUCH CONCERN TO THIS GROUP THAT IT DEVOTED LESS ATTENTION THAN WE HAD ANTICIPATED TO SUCH OTHER PROBLEMS AS HOW BEST TO USE THE THEORY OF GAMES IN DEALING WITH THE TACTICAL AND STRATEGIC OPTIONS AVAILABLE TO OPPOSING SIDES IN COMBAT."

— MORIMOC WORKSHOP REPORT

More Operational Realism in the Modeling of Combat (MORIMOC), 25-27 February 1986, Military Operations Research Society Workshop Report (April 1991; AD-B154 505L), page 5-5.

OBSERVATIONS OF THE MATHEMATICAL WORKING GROUP

"1. COMBAT MODELING TODAY HAS SOME SERIOUS PROBLEMS, AND THEY IMPLY MORE THAN JUST MINOR READJUSTMENTS IN THE COMMUNITY'S WAY OF DOING BUSINESS.

2. THERE IS A WIDESPREAD LACK OF RIGOR IN HANDLING DATA, IN MATHEMATICAL TREATMENT OF OPERATIONS, IN MEASURES OF EFFECTIVENESS . . . .

. . . .

6. THERE IS EVIDENCE OF FREQUENT MISUSE OF 'EXPECTED VALUES,' AND THIS SEEMS TO BE DUE, AT LEAST IN PART, TO IGNORANCE OR NEGLECT OF SIGNIFICANT OPERATIONAL DEPENDENCIES AND SPATIAL CONFIGURATIONS OF OPERATIONAL ELEMENTS.

. . . .

8. NONE OF THE PROBLEMS SURFACED BY THE GROUP WAS NEW, AND NOT MUCH PROGRESS SEEMS TO HAVE BEEN MADE IN THEM OVER THE LAST 10 OR 15 YEARS."

— MORIMOC WORKSHOP REPORT

More Operational Realism in the Modeling of Combat (MORIMOC), 25-27 February 1986, Military Operations Research Society Workshop Report (April 1991; AD-B154 505L), page 3-12.

"COMPLEX COMBAT SIMULATIONS WHICH ESTIMATE OPERATIONAL PERFORMANCE AT THE FORCE-ON-FORCE LEVEL (SOME WOULD ALSO ARGUE AT THE ONE-[ON]-ONE ENGAGEMENT LEVEL) NATURALLY ENCOUNTER A GREAT DEAL MORE SKEPTICISM [THAN SYSTEM AND SUBSYSTEM MODELS AT THE ENGINEERING LEVEL], SINCE THESE HIGH LEVEL MODELS MUST NECESSARILY MAKE SIMPLIFYING ASSUMPTIONS AND SACRIFICE DETAIL. CONTRIBUTING TO THIS DISTRUST IS THE FACT THAT THE FUNDAMENTAL THEORETICAL BASES FOR THE SIMPLIFICATIONS ARE LESS WELL UNDERSTOOD (LANCHESTER'S EQUATIONS HARDLY INSPIRE THE CONFIDENCE OF MAXWELL'S)."

— DEFENSE SCIENCE BOARD

Report of the Defense Science Board Task Force on Improving Test and Evaluation Effectiveness (Office of the Under Secretary of Defense for Acquisition, December 1989), page 11.

CONFIGURAL THEORY TERMINOLOGY

- MULTIPLE-TARGET, MULTIPLE-WEAPON ENCOUNTERS
- THE ATTACKER-DEFENDER CONFIGURATION AND THE TARGET-WEAPON CONFIGURATION
- CONFIGURED ENCOUNTERS
- FREE ENCOUNTERS

CONFIGURED ENCOUNTER

- THE RANDOM VARIABLES THAT CHARACTERIZE THE TARGETS AND THE WEAPONS, SUCH AS THE TARGET AND THE WEAPON POSITIONS AND ORIENTATIONS, ARE PROBABILISTICALLY DEPENDENT IN A MANNER DETERMINED BY THE ATTACKER-DEFENDER CONFIGURATION

FREE ENCOUNTER

- THE TARGETS AND THE WEAPONS ENCOUNTER EACH OTHER PROBABILISTICALLY INDEPENDENTLY

- THE RANDOM VARIABLES THAT RESPECTIVELY CHARACTERIZE THOSE TARGETS AND THOSE WEAPONS ARE PROBABILISTICALLY INDEPENDENT

- THERE IS NO "INVISIBLE HAND" THAT MAKES SOME ENCOUNTERS POSSIBLE FOR ONE PARTICULAR TARGET OR WEAPON AND IMPOSSIBLE FOR ANOTHER. THE TARGETS AND THE WEAPONS ARE FREE FROM THE CONFIGURAL CONSTRAINTS IMPOSED BY THE ATTACKER-DEFENDER CONFIGURATION.

W89-008-1

CONFIGURATION

- PARTICULAR TARGETS AND PARTICULAR WEAPONS, EACH WITH ITS PARTICULAR STATUS, MUST BE AT PARTICULAR POSITIONS IN SPACE AT ANY PARTICULAR TIME

- THOSE TARGETS AND THOSE WEAPONS TOGETHER WITH THEIR POSITIONS IN SPACE DEFINE THE TARGET-WEAPON CONFIGURATION AT A GIVEN TIME

- AT ANY PARTICULAR TIME A PARTICULAR TARGET-WEAPON CONFIGURATION EXISTS, AND IT DETERMINES THE KIND, STATUS, POSITION, AND STATE OF MOTION OF EACH TARGET AND EACH WEAPON

- THE ATTACKER AND THE DEFENDER EACH CONSTITUTE A TARGET-WEAPON CONFIGURATION, AND THOSE CONFIGURATIONS COLLECTIVELY DETERMINE THE ATTACKER-DEFENDER CONFIGURATION

THE BEHAVIOR OF WEAPONS IN USE IN COMBAT IN CONFIGURAL THEORY

- THE BEHAVIOR OF WEAPONS IN USE IN COMBAT IS IDENTIFIED WITH THE FAMILY OF JOINT PROBABILITY DENSITIES THAT CONNECT THE ATTACKER-DEFENDER CONFIGURATION WITH
  - — THE RANDOM NUMBERS OF OWN WEAPONS EXPENDED,
  - — THE RANDOM NUMBERS OF OWN FORCE SURVIVORS, AND
  - — THE RANDOM NUMBERS OF ENEMY FORCE CASUALTIES

- CONVENTIONAL AGGREGATION, SUCH AS THAT IMPLICIT IN LANCHESTER THEORY — STOCHASTIC OR DETERMINISTIC — TREATS PROBABILISTICALLY DEPENDENT RANDOM EVENTS AS INDEPENDENT AND, IN PARTICULAR, TARGET-WEAPON ENCOUNTERS AS FREE ENCOUNTERS

AGGREGATION

- TREATING NONIDENTICAL ENTITIES OR PROCESSES AS IDENTICAL, SUCH AS
  - — ASSIGNING THE SAME HIT OR KILL PROBABILITIES TO NONIDENTICAL TARGETS WITH IDENTICAL COVER AT IDENTICAL RANGES
  - — ASSIGNING THE SAME SMALL ARMS PROFICIENCY TO ALL MEMBERS OF A HETEROGENEOUS GROUP
  - — ASSIGNING THE SAME HIT OR KILL PROBABILITIES TO IDENTICAL TARGETS WITH DIFFERENT COVER OR AT DIFFERENT RANGES
  - — ASSIGNING THE SAME ACQUISITION RATE TO IDENTICAL TARGETS WITH DIFFERENT DEGREES OF CONCEALMENT

HOW AGGREGATION DESTROYS CONFIGURATION AND UNDERSTATES THE NUMBER OF ROUNDS REQUIRED FOR A KILL

- IN A CONFIGURED ENCOUNTER, EVEN TWO INITIALLY IDENTICAL SHOOTERS THAT FIRE AT IDENTICAL, NONFIRING TARGETS ARE GENERALLY NOT IDENTICAL BECAUSE ONE USUALLY EXPENDS FEWER ROUNDS TO DESTROY ITS TARGET THAN THE OTHER

- AN AGGREGATED REPRESENTATION OF SUCH A MULTIPLE-TARGET, MULTIPLE-WEAPON CONFIGURED ENCOUNTER CREATES PHANTOM SHOOTERS, THAT IS, "SHOOTERS" THAT HAVE NO AMMUNITION BUT CONTINUE TO FIRE AND DESTROY TARGETS

ILLUSTRATIONS BASED ON A CONFIGURATION COMPRISING TWO TARGET-SHOOTER PAIRS

IDENTICAL TARGET-SHOOTER PAIRS

- TARGETS ARE IDENTICAL AND NONFIRING
- SHOOTERS ARE IDENTICAL, AND EACH FIRES AT A DISTINCT, SINGLE TARGET
- SHOOTERS FIRE INDEPENDENTLY OF EACH OTHER

THE AGGREGATION OF ALL SHOOTERS THAT IS IMPLICIT IN USING THE AVERAGE ROUNDS EXPENDED BY AN UNSPECIFIED SHOOTER MASKS THE DISPARITY IN ROUNDS REQUIRED FOR A KILL

CONFIGURATION COMPRISES TWO IDENTICAL TARGET-SHOOTER PAIRS

	SINGLE-ROUND KILL PROBABILITY		
SHOOTER	0.1	0.3	0.5
FIRST TO KILL	5.26	1.96	1.33
SECOND TO KILL	14.74	4.71	2.67
UNSPECIFIED	10.00	3.33	2.00

AVERAGE NUMBER OF ROUNDS EXPENDED

PROBABILISTICALLY IDENTICAL TARGET-SHOOTER PAIRS

- TARGET-SHOOTER PAIRS HAVE IDENTICAL SHOOTERS
- EACH TARGET IS RANDOMLY AND INDEPENDENTLY DETERMINED AND THE RESULTING SINGLE-ROUND KILL PROBABILITY IS EQUALLY LIKELY TO BE 0.1, 0.3, OR 0.5

IDENTICAL, AGGREGATED TARGET-SHOOTER PAIRS

- TARGET-SHOOTER PAIRS HAVE IDENTICAL SHOOTERS AND IDENTICAL, AGGREGATED TARGETS
- THE SINGLE-ROUND KILL PROBABILITY FOR THE AGGREGATED TARGET IN EACH PAIR IS 0.3, THE AVERAGE OF THE 0.1, 0.3, AND 0.5 SINGLE-ROUND KILL PROBABILITIES ASSOCIATED WITH THE EQUALLY LIKELY TARGETS

TARGET AGGREGATION UNDERSTATES THE NUMBER OF ROUNDS EXPENDED BY EACH SHOOTER AND THE TOTAL ROUNDS EXPENDED

ACTUAL CONFIGURATION COMPRISES TWO PROBABILISTICALLY IDENTICAL TARGET-SHOOTER PAIRS

	TARGETS		
SHOOTER	AGGREGATED	ACTUAL	RATIO
FIRST TO KILL	1.96	2.30	1.17
SECOND TO KILL	4.71	7.92	1.68
UNSPECIFIED	3.33	5.11	1.53

AVERAGE NUMBER OF ROUNDS EXPENDED

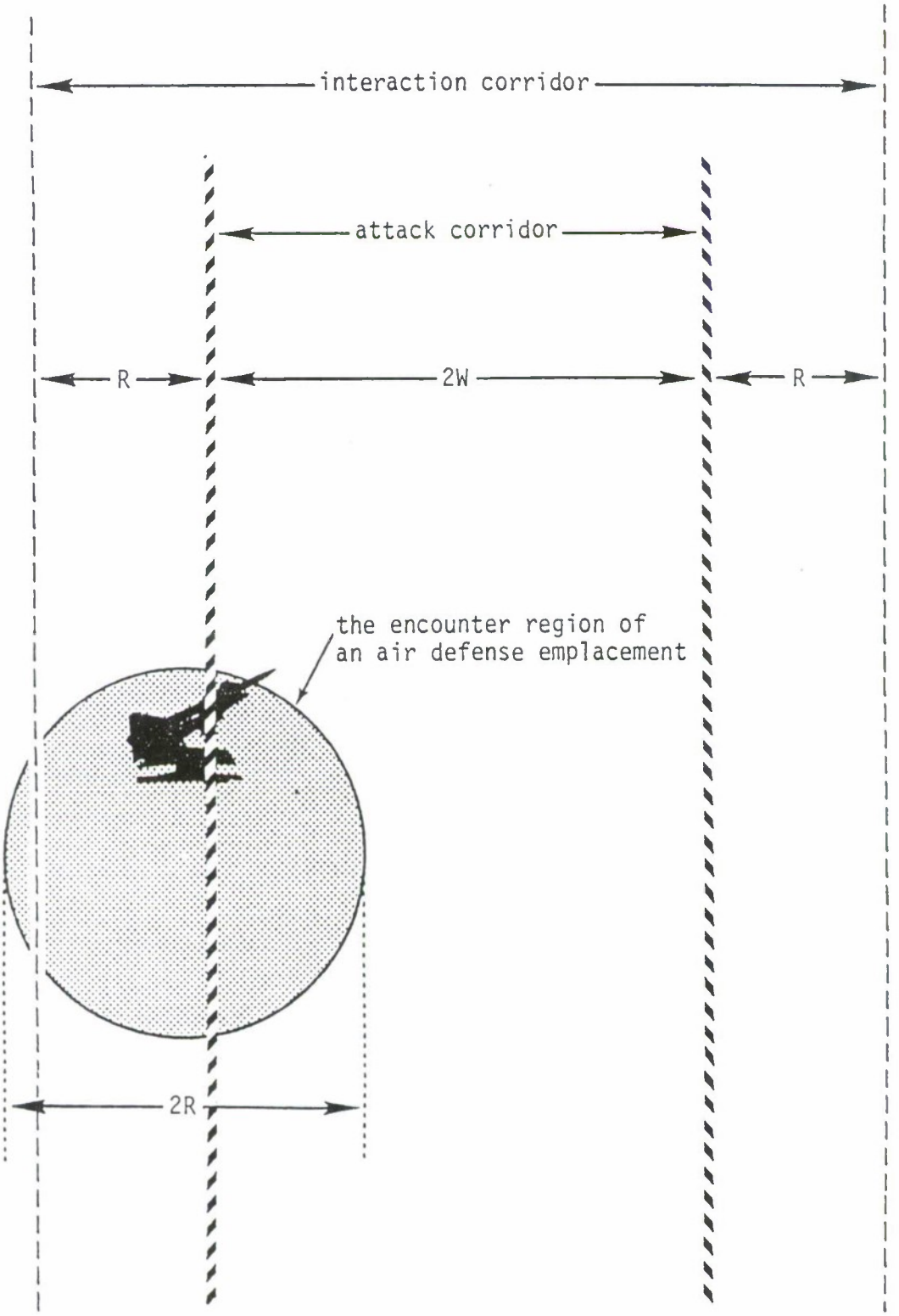
TARGET AGGREGATION UNDERSTATES THE SHOOTER-TO-SHOOTER VARIATION IN ROUNDS EXPENDED

TARGETS	SHOOTER FIRST TO KILL	SHOOTER SECOND TO KILL	RATIO
AGGREGATED	1.96	4.71	2.40
ACTUAL	2.30	7.92	3.44

AVERAGE NUMBER OF ROUNDS EXPENDED

ILLUSTRATIONS BASED ON AN IDEALIZED, SIMPLIFIED AIR DEFENSE WEAPONS SYSTEM DEFENDING AN EXTENDED AREA

THE INTERACTION CORRIDOR OF AN ATTACK CORRIDOR IS THE LARGEST REGION THROUGHOUT WHICH AN EMPLACEMENT OF A PARTICULAR AIR DEFENSE SYSTEM HAS A POSITIVE PROBABILITY OF BEING ENCOUNTERED BY AN ATTACKER IN THE ATTACK CORRIDOR



ATTACKER-DEFENDER CONFIGURATION

- THE DEFENDER'S WEAPON EMPLACEMENTS ARE UNIFORMLY RANDOMLY AND INDEPENDENTLY DISTRIBUTED WITHIN THE DEFENDED AREA
- THE ATTACK CORRIDOR IS LOCATED INDEPENDENTLY OF WEAPON EMPLACEMENTS
- THE ATTACKER'S TRAJECTORIES ARE UNIFORMLY RANDOMLY AND INDEPENDENTLY DISTRIBUTED ACROSS THE ATTACK CORRIDOR

PERFECT FIRE CONTROL

- NO ATTACKER IS ENGAGED BY MORE THAN ONE EMPLACEMENT SIMULTANEOUSLY

- AN ACQUIRED ATTACKER IS IMMEDIATELY FIRED UPON, AND THE RESULT OF THE FIRING — THE ATTACKER IS UNDAMAGED OR DESTROYED — IS KNOWN IMMEDIATELY AT ALL EMPLACEMENTS IN THE INTERACTION CORRIDOR

OPERATIONAL CHARACTERISTICS OF THE HYPOTHETICAL ANTIAIRCRAFT WEAPON

AVAILABILITY*	
MISSILE	0.8
LAUNCHER	0.875
ACQUISITION PROBABILITY	0.8
DAMAGE PROBABILITY	0.5
RANGE (YARDS)	200,000
NUMBER OF ROUNDS AT AN EMPLACEMENT	AS SPECIFIED
NUMBER OF FIRINGS PER DETECTED ATTACKER	4

* apportionment of a 0.7 single-round availability between an individual missile and the multiple-round launcher

COMBAT SITUATION

- 100 ATTACKERS USE AN ATTACK CORRIDOR 5 NAUTICAL MILES WIDE THROUGH THE DEFENDED AREA
- ANTIAIRCRAFT WEAPONS CANNOT BE DESTROYED BY THE ATTACKERS

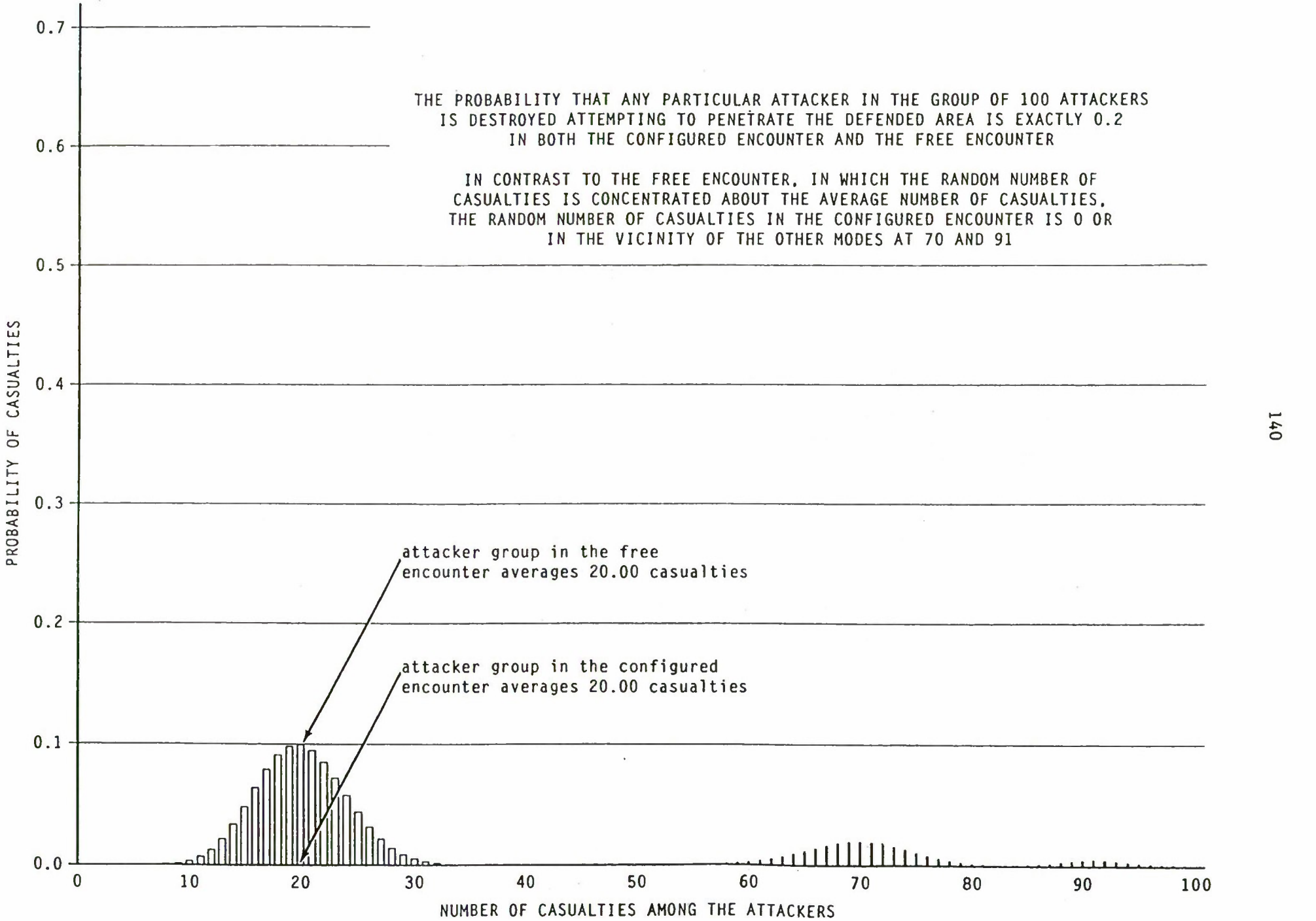
A COMPARISON OF CASUALTY PRODUCTION IN CONFIGURED AND FREE ENCOUNTERS WITH THE SAME AVERAGE NUMBER OF CASUALTIES

- POSTULATING CASUALTY PRODUCTION TO BE APPROXIMATED BY THE AVERAGE NUMBER OF CASUALTIES IN EFFECT ENTAILS POSTULATING THE ENCOUNTERS TO BE FREE ENCOUNTERS

- THE PROBABILITY DENSITY OF CASUALTIES IN A CONFIGURED ENCOUNTER CAN DIFFER DRAMATICALLY FROM THAT OF THE CORRESPONDING FREE ENCOUNTER EVEN THOUGH EACH AVERAGES EXACTLY THE SAME NUMBER OF CASUALTIES

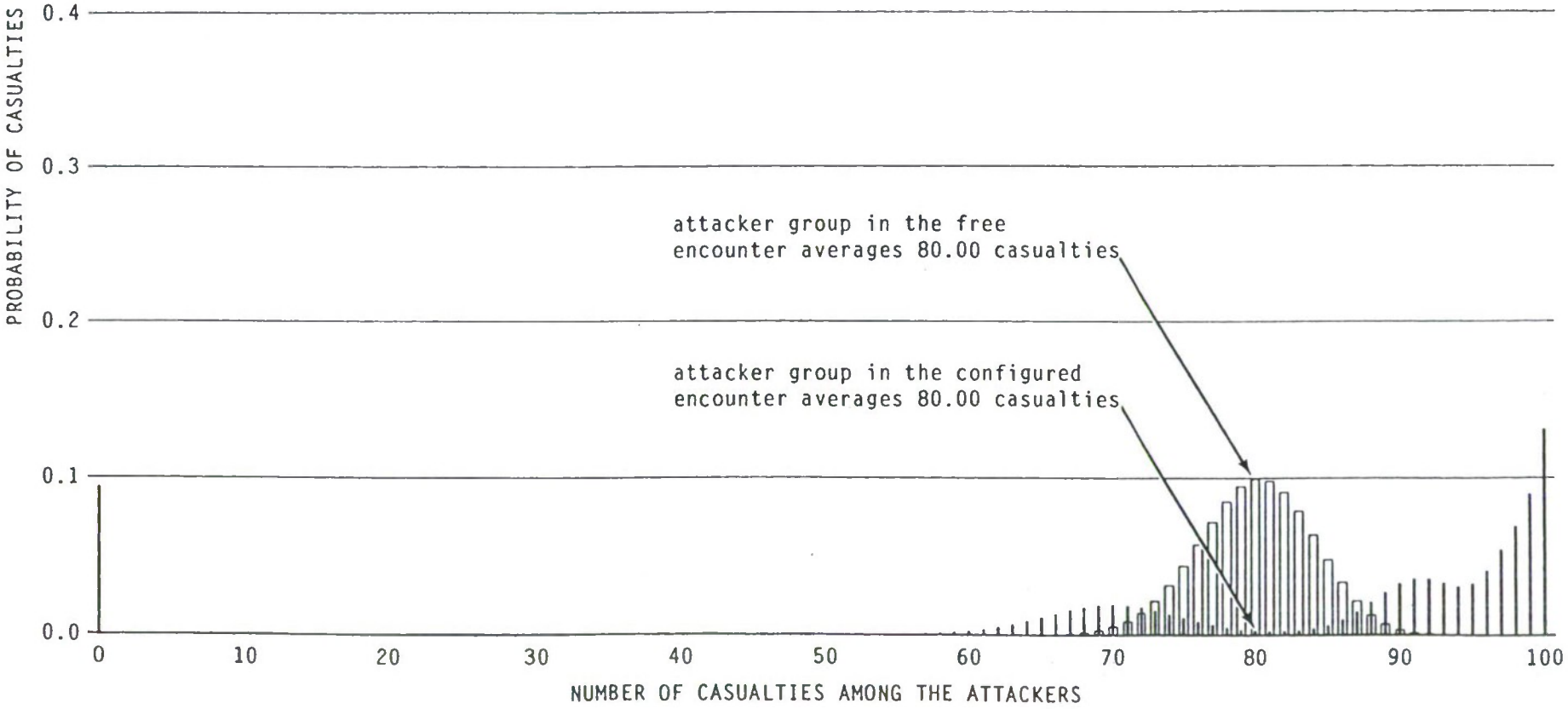
CONSTANT KILL PROBABILITIES IN A CONFIGURED ENCOUNTER

- IN A CONFIGURED ENCOUNTER IN WHICH EACH EMPLACEMENT IN THE DEFENDER CONFIGURATION HAS AT LEAST AS MANY ROUNDS AS THE MAXIMUM POSSIBLE NUMBER THAT A SINGLE EMPLACEMENT COULD FIRE AT THE GIVEN GROUP OF ATTACKERS,
  - — EVERY ATTACKER IN THE GROUP HAS EXACTLY THE SAME PROBABILITY OF BEING DESTROYED AND
  - — THAT CONSTANT KILL PROBABILITY EXACTLY EQUALS THE CONFIGURAL ATTRITION RATE


THE PROBABILITY THAT ANY PARTICULAR ATTACKER IN THE GROUP OF 100 ATTACKERS IS DESTROYED ATTEMPTING TO PENETRATE THE DEFENDED AREA IS EXACTLY 0.2 IN BOTH THE CONFIGURED ENCOUNTER AND THE FREE ENCOUNTER
IN CONTRAST TO THE FREE ENCOUNTER, IN WHICH THE RANDOM NUMBER OF CASUALTIES IS CONCENTRATED ABOUT THE AVERAGE NUMBER OF CASUALTIES, THE RANDOM NUMBER OF CASUALTIES IN THE CONFIGURED ENCOUNTER IS 0 OR IN THE VICINITY OF THE OTHER MODES AT 70 AND 91
attacker group in the free encounter averages 20.00 casualties
attacker group in the configured encounter averages 20.00 casualties
PROBABILITY OF CASUALTIES
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
10
20
30
40
50
60
70
80
90
100
NUMBER OF CASUALTIES AMONG THE ATTACKERS

THE PROBABILITY THAT ANY PARTICULAR ATTACKER IN THE GROUP OF 100 ATTACKERS IS DESTROYED ATTEMPTING TO PENETRATE THE DEFENDED AREA IS EXACTLY 0.8 IN BOTH THE CONFIGURED ENCOUNTER AND THE FREE ENCOUNTER

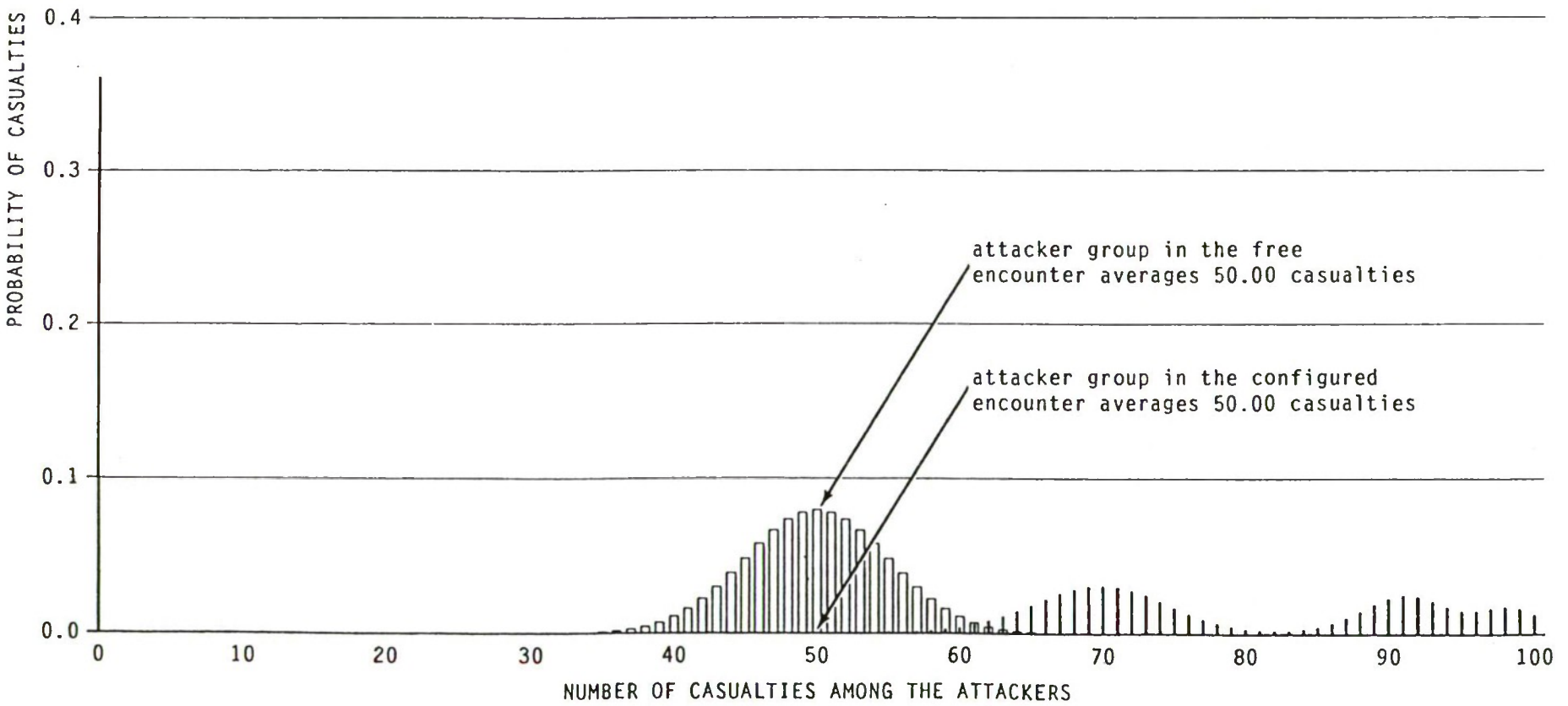
IN CONTRAST TO THE FREE ENCOUNTER, IN WHICH THE RANDOM NUMBER OF CASUALTIES IS CONCENTRATED ABOUT THE AVERAGE NUMBER OF CASUALTIES, THE RANDOM NUMBER OF CASUALTIES IN THE CONFIGURED ENCOUNTER IS 0 OR IN THE VICINITY OF THE OTHER MODES AT 70, 91, AND 100



HORRIGAN ANALYTICS

THE PROBABILITY THAT ANY PARTICULAR ATTACKER IN THE GROUP OF 100 ATTACKERS IS DESTROYED ATTEMPTING TO PENETRATE THE DEFENDED AREA IS EXACTLY 0.5 IN BOTH THE CONFIGURED ENCOUNTER AND THE FREE ENCOUNTER

IN CONTRAST TO THE FREE ENCOUNTER, IN WHICH THE RANDOM NUMBER OF CASUALTIES IS CONCENTRATED ABOUT THE AVERAGE NUMBER OF CASUALTIES, THE RANDOM NUMBER OF CASUALTIES IN THE CONFIGURED ENCOUNTER IS 0 OR IN THE VICINITY OF THE OTHER MODES AT 70, 91, AND 98

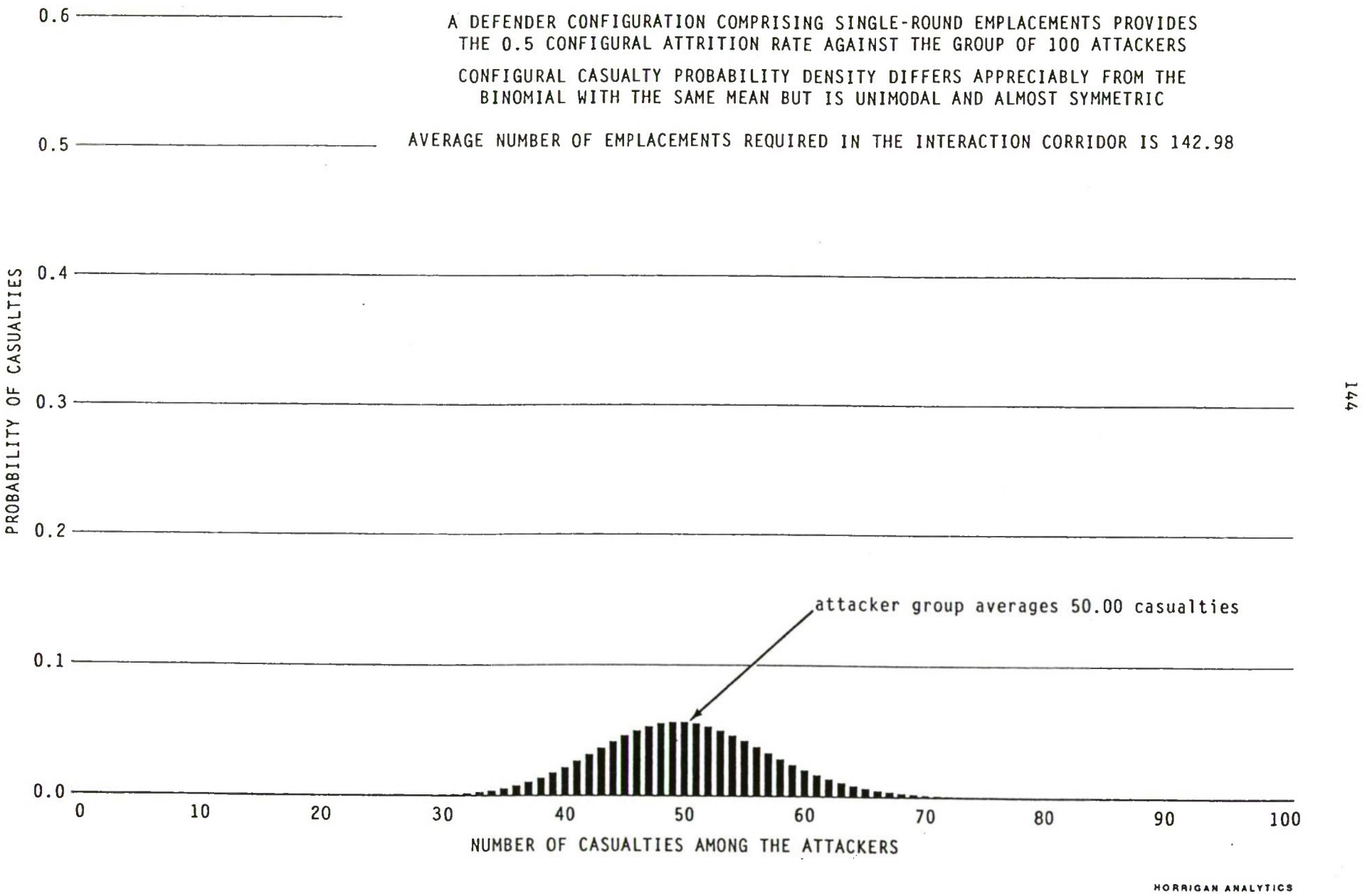


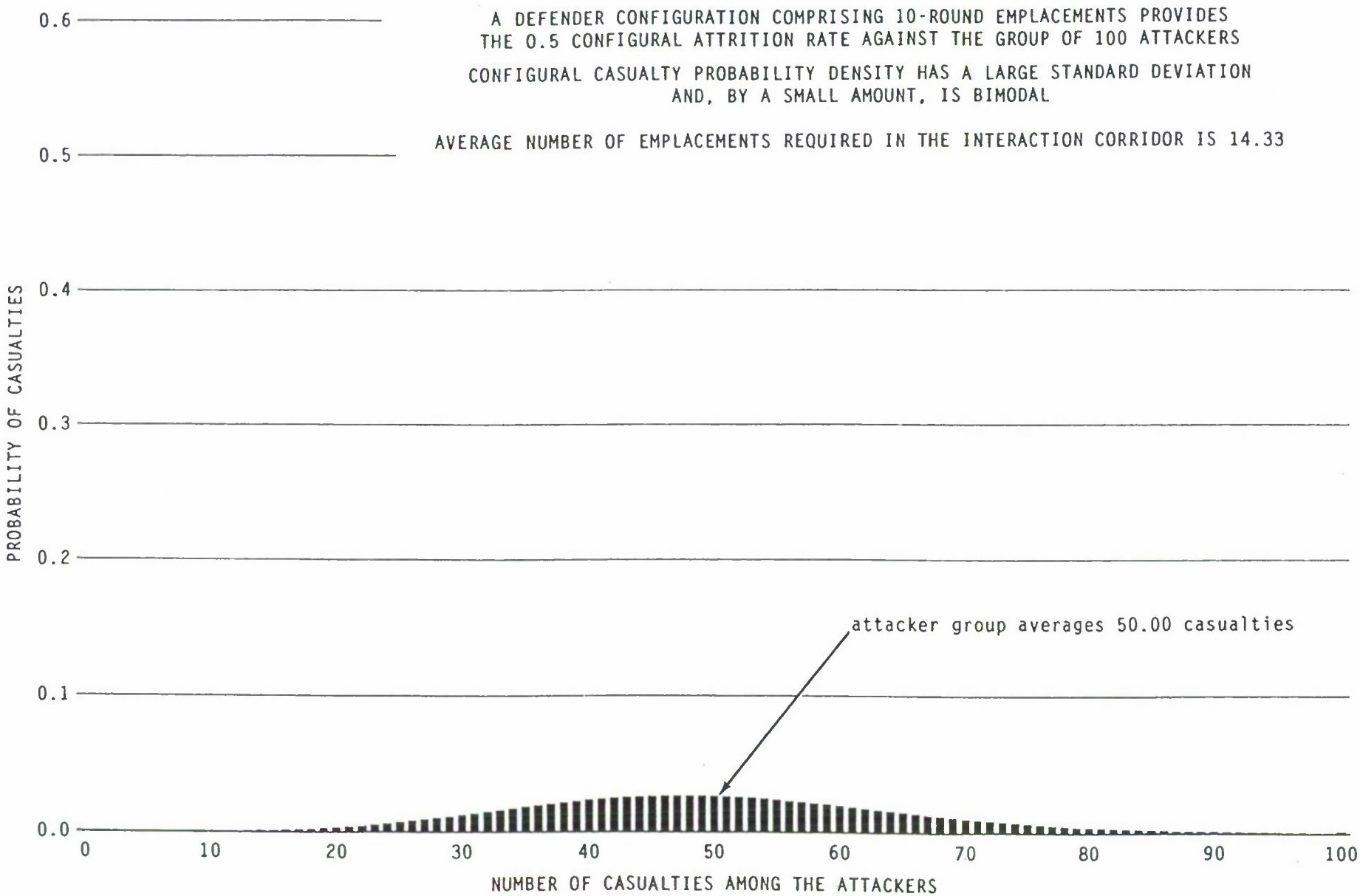
HORRIGAN ANALYTICS

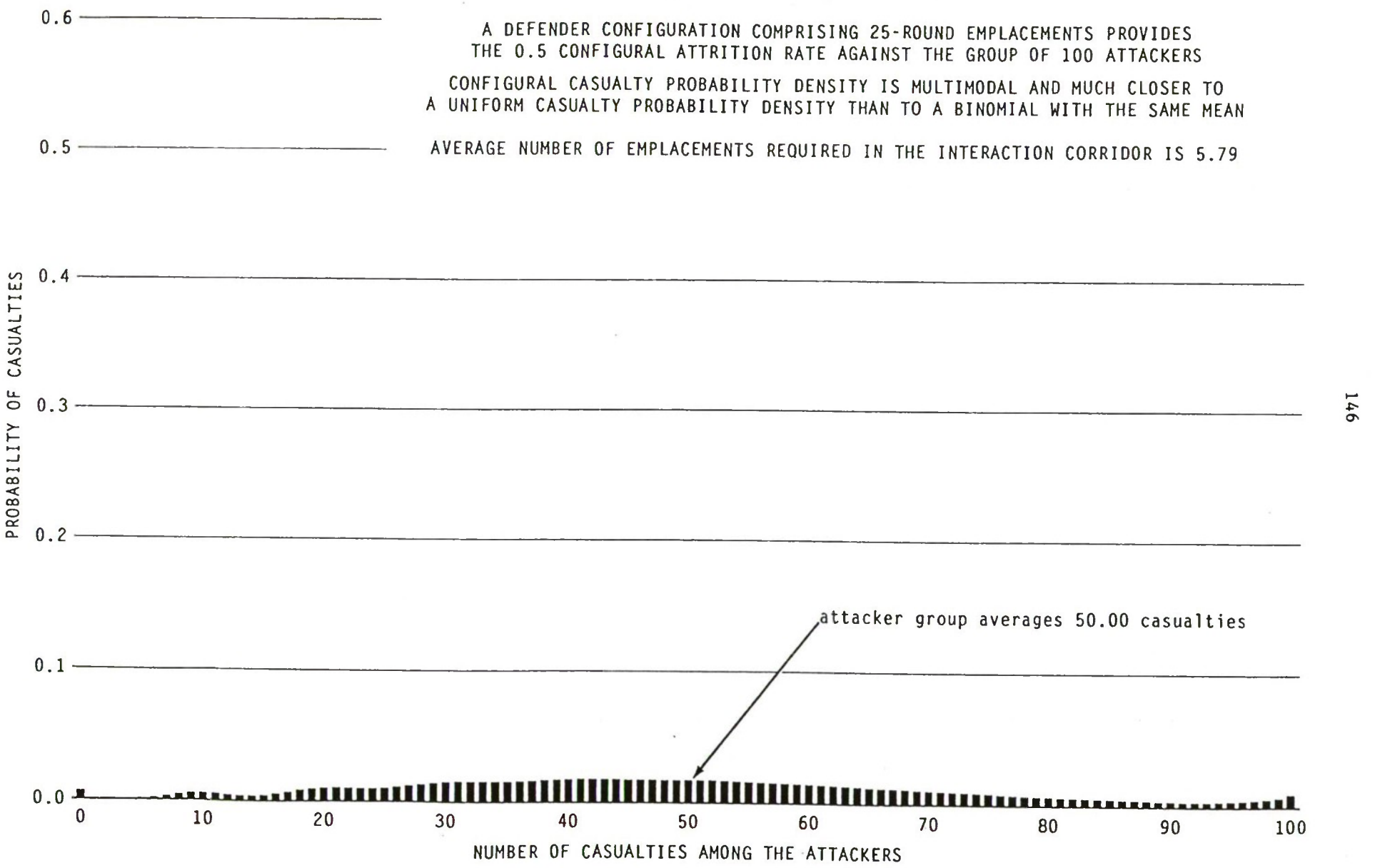
AN ILLUSTRATION OF THE CONFIGURAL EFFECTS ON CASUALTY PRODUCTION OF DIFFERENT CONFIGURATIONS THAT EXACTLY AVERAGE THE SAME NUMBER OF CASUALTIES

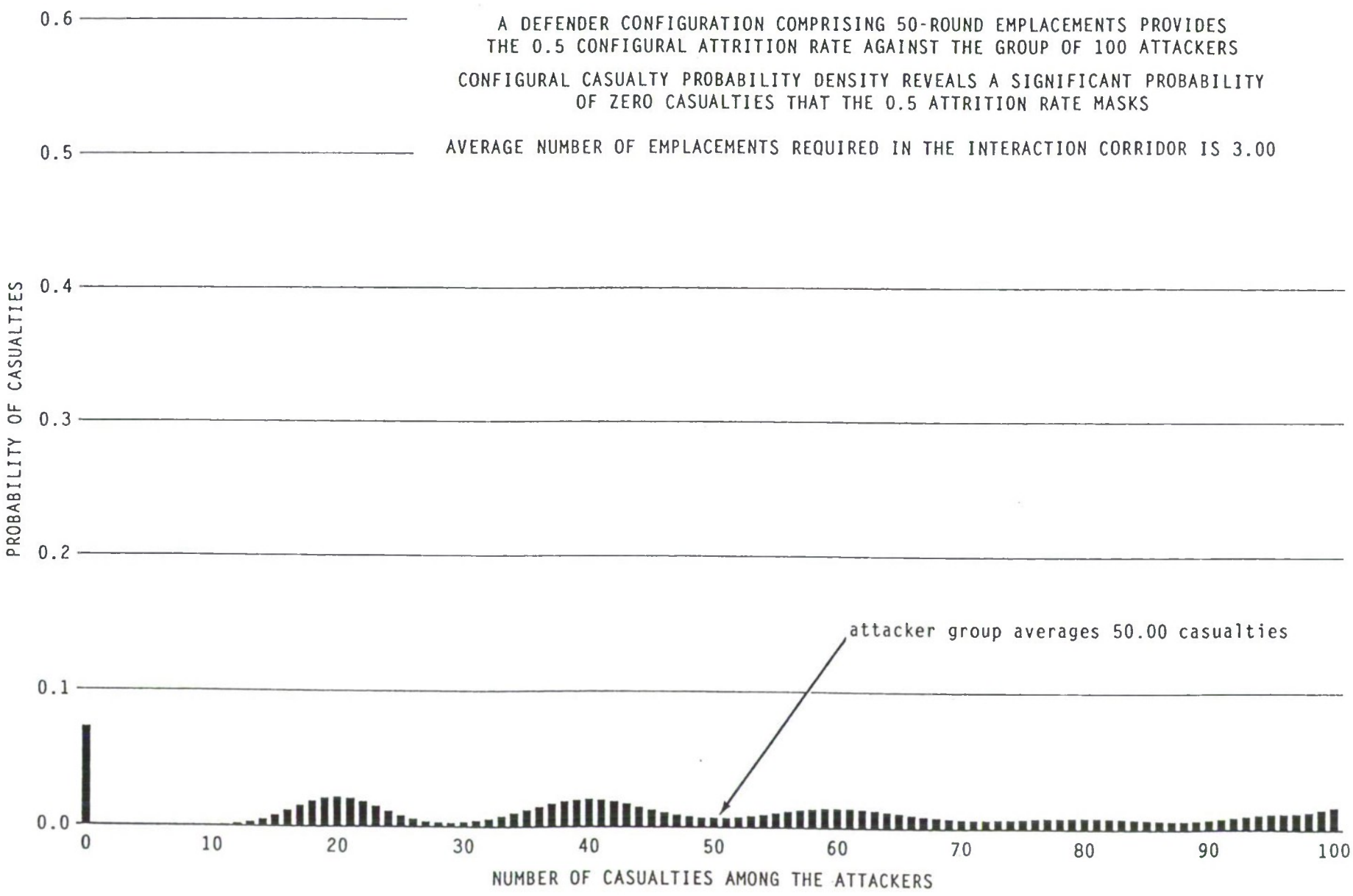
- CHANGING THE NUMBER OF ROUNDS THAT ARE INITIALLY AT EACH WEAPONS EMPLACEMENT IN THE DEFENDER CONFIGURATION OR THE NUMBER OF EMPLACEMENTS CAN GREATLY CHANGE THE PROBABILITY DENSITY OF CASUALTIES EVEN THOUGH THE AVERAGE NUMBER OF CASUALTIES REMAINS EXACTLY THE SAME

- ANY SINGLE NUMBER IS AT BEST MARGINALLY REPRESENTATIVE OF CASUALTY PRODUCTION FOR SINGLE-ROUND EMPLACEMENTS AND IS GROSSLY NONREPRESENTATIVE FOR EMPLACEMENTS WITH 25 ROUNDS OR MORE


A DEFENDER CONFIGURATION COMPRISING SINGLE-ROUND EMPLACEMENTS PROVIDES THE 0.5 CONFIGURAL ATTRITION RATE AGAINST THE GROUP OF 100 ATTACKERS
CONFIGURAL CASUALTY PROBABILITY DENSITY DIFFERS APPRECIABLY FROM THE BINOMIAL WITH THE SAME MEAN BUT IS UNIMODAL AND ALMOST SYMMETRIC
AVERAGE NUMBER OF EMPLACEMENTS REQUIRED IN THE INTERACTION CORRIDOR IS 142.98
PROBABILITY OF CASUALTIES
0.6
0.5
0.4
0.3
0.2
0.1
0.0
attacker group averages 50.00 casualties
0
10
20
30
40
50
60
70
80
90
100
NUMBER OF CASUALTIES AMONG THE ATTACKERS
HORRIGAN ANALYTICS


A DEFENDER CONFIGURATION COMPRISING 10-ROUND EMPLACEMENTS PROVIDES
THE 0.5 CONFIGURAL ATTRITION RATE AGAINST THE GROUP OF 100 ATTACKERS
CONFIGURAL CASUALTY PROBABILITY DENSITY HAS A LARGE STANDARD DEVIATION
AND, BY A SMALL AMOUNT, IS BIMODAL
AVERAGE NUMBER OF EMPLACEMENTS REQUIRED IN THE INTERACTION CORRIDOR IS 14.33
PROBABILITY OF CASUALTIES
0.6
0.5
0.4
0.3
0.2
0.1
0.0
attacker group averages 50.00 casualties
0
10
20
30
40
50
60
70
80
90
100
NUMBER OF CASUALTIES AMONG THE ATTACKERS
HORRIGAN ANALYTICS


A DEFENDER CONFIGURATION COMPRISING 25-ROUND EMPLACEMENTS PROVIDES
THE 0.5 CONFIGURAL ATTRITION RATE AGAINST THE GROUP OF 100 ATTACKERS
CONFIGURAL CASUALTY PROBABILITY DENSITY IS MULTIMODAL AND MUCH CLOSER TO
A UNIFORM CASUALTY PROBABILITY DENSITY THAN TO A BINOMIAL WITH THE SAME MEAN
AVERAGE NUMBER OF EMPLACEMENTS REQUIRED IN THE INTERACTION CORRIDOR IS 5.79
PROBABILITY OF CASUALTIES
0.6
0.5
0.4
0.3
0.2
0.1
0.0
attacker group averages 50.00 casualties
0
10
20
30
40
50
60
70
80
90
100
NUMBER OF CASUALTIES AMONG THE ATTACKERS
HORRIGAN ANALYTICS

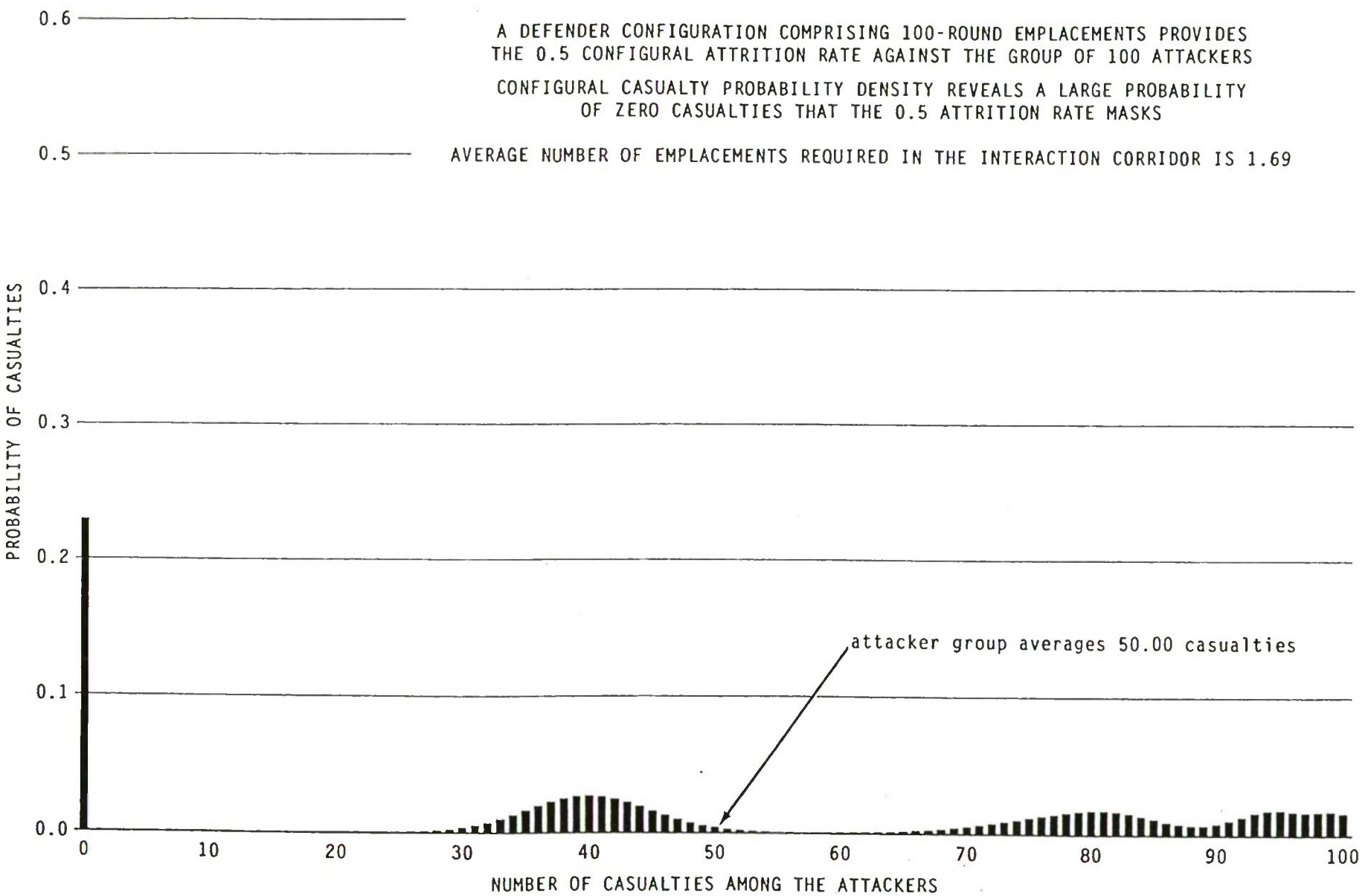


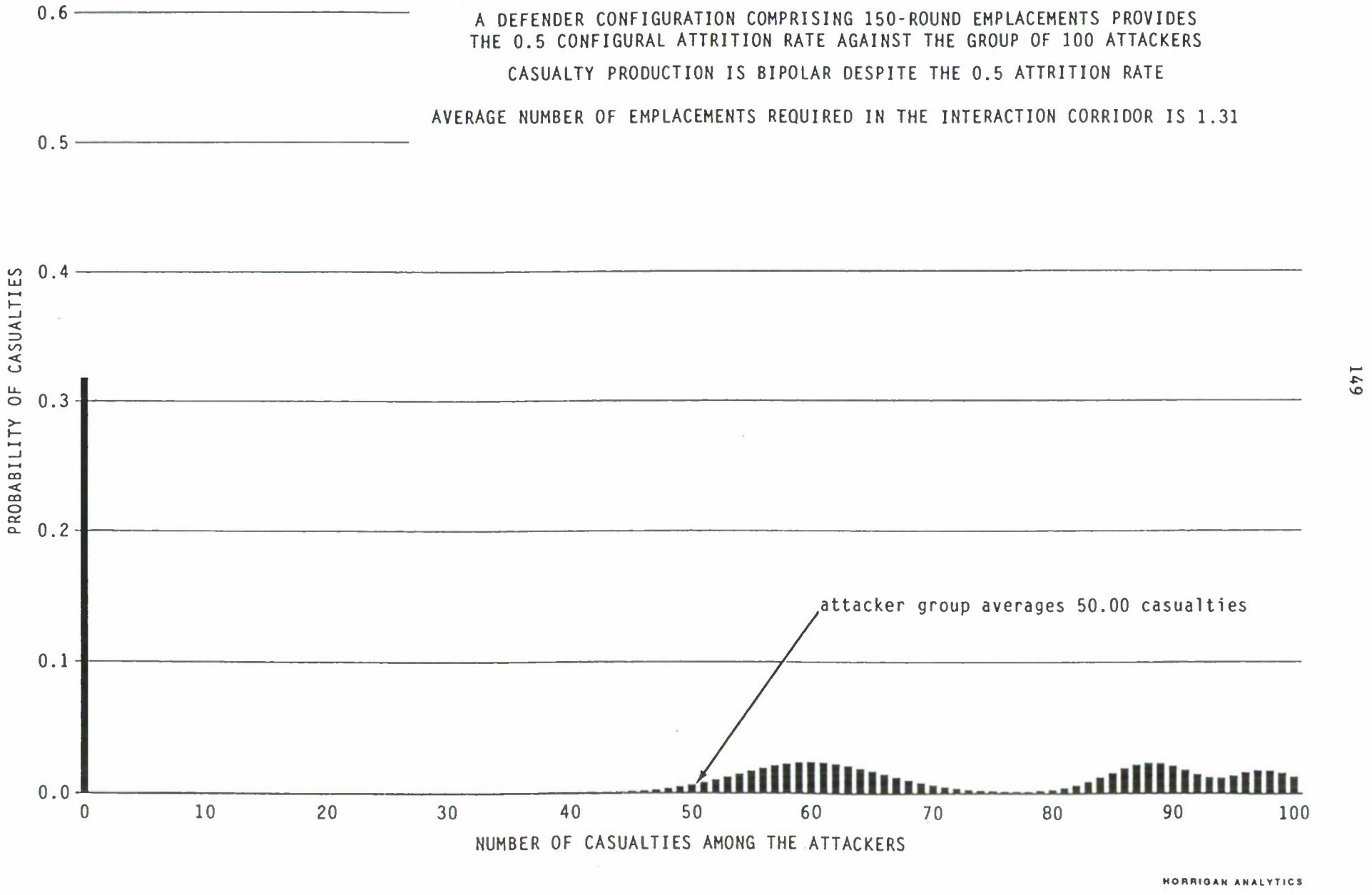
HORRIGAN ANALYTICS

A DEFENDER CONFIGURATION COMPRISING 100-ROUND EMPLACEMENTS PROVIDES THE 0.5 CONFIGURAL ATTRITION RATE AGAINST THE GROUP OF 100 ATTACKERS

CONFIGURAL CASUALTY PROBABILITY DENSITY REVEALS A LARGE PROBABILITY OF ZERO CASUALTIES THAT THE 0.5 ATTRITION RATE MASKS

AVERAGE NUMBER OF EMPLACEMENTS REQUIRED IN THE INTERACTION CORRIDOR IS 1.69

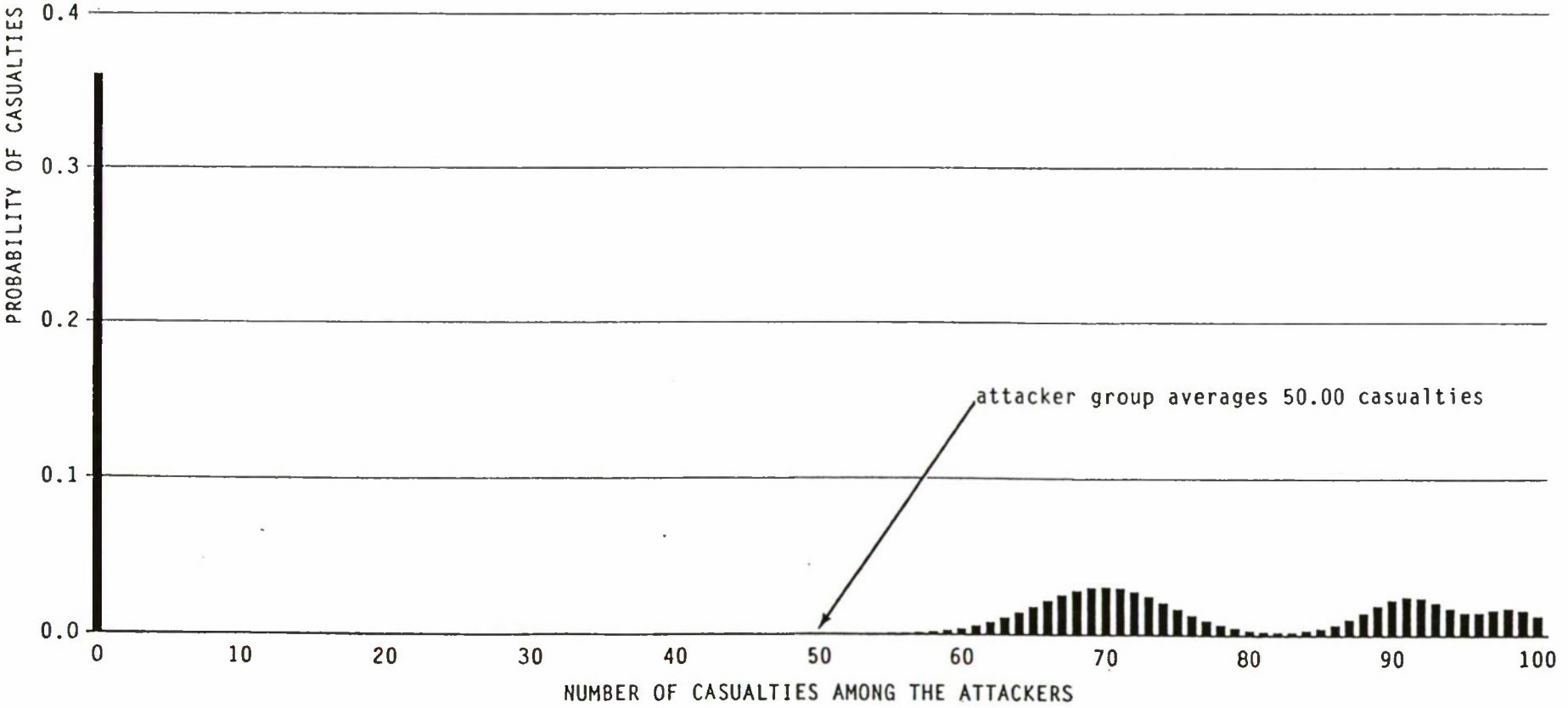



A DEFENDER CONFIGURATION COMPRISING 150-ROUND EMPLACEMENTS PROVIDES
THE 0.5 CONFIGURAL ATTRITION RATE AGAINST THE GROUP OF 100 ATTACKERS
CASUALTY PRODUCTION IS BIPOLAR DESPITE THE 0.5 ATTRITION RATE
AVERAGE NUMBER OF EMPLACEMENTS REQUIRED IN THE INTERACTION CORRIDOR IS 1.31
PROBABILITY OF CASUALTIES
0.6
0.5
0.4
0.3
0.2
0.1
0.0
attacker group averages 50.00 casualties
0
10
20
30
40
50
60
70
80
90
100
NUMBER OF CASUALTIES AMONG THE ATTACKERS
HORRIGAN ANALYTICS

A DEFENDER CONFIGURATION WITH AT LEAST 400-ROUND EMPLACEMENTS PROVIDES THE 0.5 CONFIGURAL ATTRITION RATE AGAINST THE GROUP OF 100 ATTACKERS

CASUALTY PROBABILITY DENSITY IS SUFFICIENTLY BIPOLAR THAT THE PROBABILITY OF ANY NUMBER OF CASUALTIES BETWEEN 40 AND 60 IS NEARLY ZERO

AVERAGE NUMBER OF EMPLACEMENTS REQUIRED IN THE INTERACTION CORRIDOR IS 1.17



A DEFENDER CONFIGURATION WITH A HIGH FREE-ENCOUNTER ATTRITION RATE

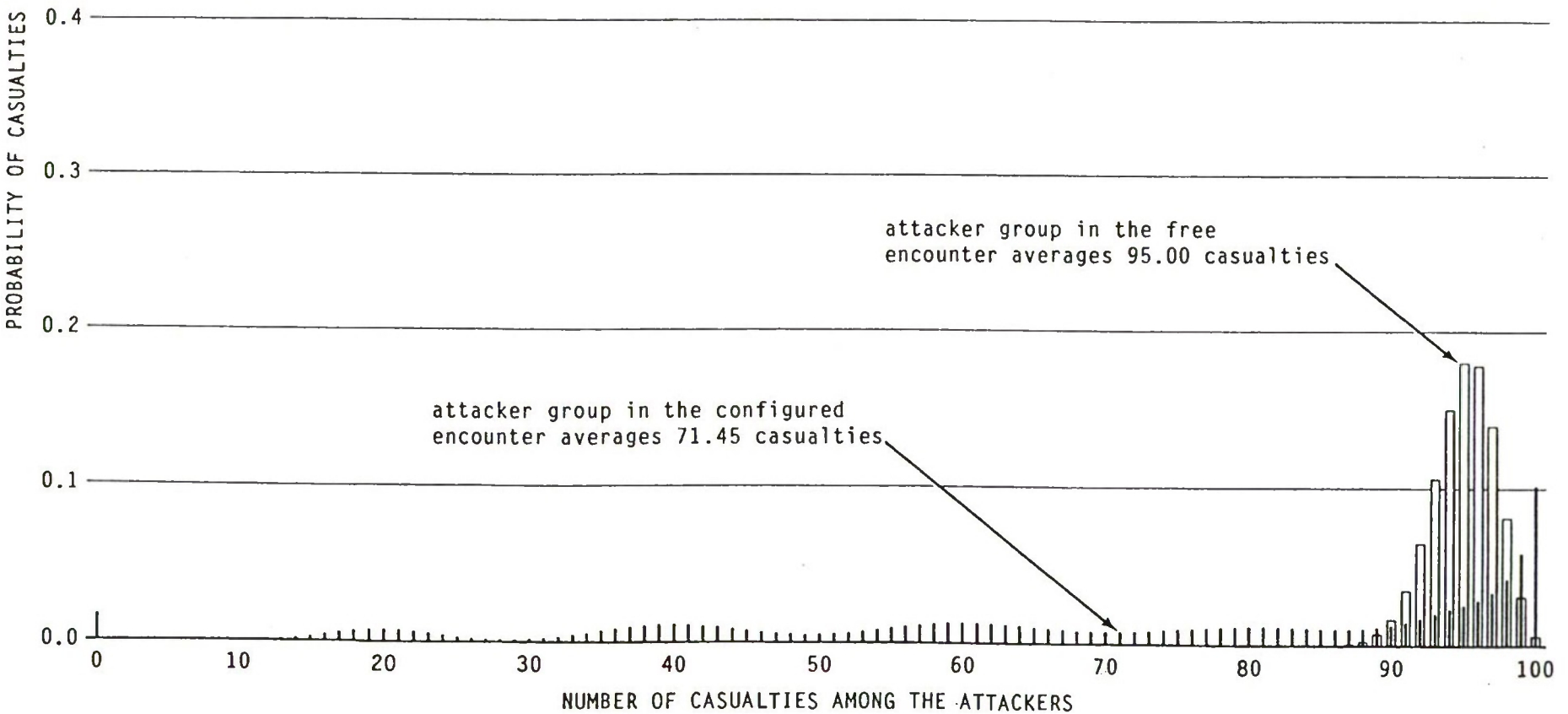
- THE EMPLACEMENT DENSITY OF THE DEFENDER CONFIGURATION PROVIDES A 0.95 FREE-ENCOUNTER ATTRITION RATE AND AVERAGES 5.04 EMPLACEMENTS IN THE INTERACTION CORRIDOR

- EACH WEAPON EMPLACEMENT IN THE DEFENDER CONFIGURATION INITIALLY HAS 50 ROUNDS

A VIVID ILLUSTRATION OF THE GENERAL INADEQUACY OF CUSTOMARY AGGREGATION
TO ACCOMMODATE THE STOCHASTIC VARIATION INHERENT IN COMBAT

NO SINGLE NUMBER IS REPRESENTATIVE OF THE CASUALTY PRODUCTION OF THE CONFIGURED ENCOUNTER

THE CONFIGURED ENCOUNTER NOT ONLY RESULTS IN SUBSTANTIALLY FEWER CASUALTIES THAN THE CORRESPONDING FREE ENCOUNTER BUT ALSO IN A STANDARD DEVIATION (26.65) THAT IS MORE THAN TWELVE TIMES GREATER THAN THAT OF THE FREE ENCOUNTER (2.18)



HORRIGAN ANALYTICS

CONCLUSIONS

- CONVENTIONAL AGGREGATION IS A MAJOR IMPEDIMENT TO REALIZING THE POTENTIAL OF MODELING AND SIMULATION FOR IMPROVING THE EFFECTIVENESS OF WEAPONS, TACTICS, AND TRAINING

- CONFIGURATION SOLVES THE "AGGREGATION PROBLEM"

Crossing Levels of Resolution in Defense Analysis: A Requirement of the New Strategic Environment

Reiner K. Huber
Institut für Angewandte Systemforschung
und Operations Research
Fakultät für Informatik
Universität der Bundeswehr München
Werner-Heisenberg-Weg 39
W-8014 Neubiberg, FRG

1. Introduction

In the context of this paper, the term *defense analysis* is used in the sense of highly structured analysis by means of formal models of the subject matter as practiced, e.g., in operational research and systems analysis. Thus, nearly 80 percent of the life time of defense analysis sofar is characterized by the bipolar East-West confrontation of the two superpowers (USA and USSR) that had emerged from World War 2 and their respective alliances.[1] The military focus of this bipolar confrontation was the inner-German border in Europe's Central Region where both sides maintained unprecedented levels of ready military hardware and manpower. Over the years, the short warning and high density military conflict between NATO and the Warsaw Pact (WP) in the Central Region had become the principal design scenario for NATO and the defense planners of most Western nations, including Switzerland.

In order to maximize the credibility of collective response to a WP aggression, and thus of deterrence, NATO organized its conventional *forward defense* in the central region along the well remembered layer-cake pattern of eight national corps (plus one German division in LANDJUT). This was done in a rather static manner aimed at maximizing attrition of the attacking forces in tactical-level engagements that left little room for operational-level maneuver. As a consequence, defense analysis was concerned

---

[1] The bipolar East-West Confrontation (Cold War) encompasses essentially the time period between the beginning of the Soviet blockade of West-Berlin in June 1948 and the fall of the Berlin wall in November of 1989, or the dissolution of the Warsaw Pact in April 1991, respectively. The birth of operational research dates back to an air defense exercise in the UK in 1938 (see Waddington (1973) and Miser (1980)).

mainly with tactical-level requirements and cost-effectiveness studies aimed primarily at improving the capability of the defense forces for inflicting attrition on a massive armored threat through superior technical and tactical performance of weapon systems and support systems, including command, control, communications, and intelligence.

Also, most of the problems of defense analysis were of a fairly long-term nature requiring answers to questions arising in course of a well-organized and continuous process of defense planning and force modernization in response to the perceived evolution of the principal threat.

These characteristics of the Cold War period have not only formed the style of defense analysis (as being more reactive than anticipative), but left their distinct mark on the analysis tools as well. For instance, most *combat models* developed during that period exhibit a rather high degree of detail, but rather little operational flexibility. And *decision or planning models* frequently have been designed to find optimal solutions under certainty (e.g., for the "authorized" design scenario) rather than robust solutions under risk and uncertainty.

The so-called *piston-type* models are typical examples for modeling NATO's previous layer-cake-type defense in the central region: Weapon systems and military units move back and forth within rigid boundaries representing, for example, corps or division sectors, with movement rates and direction being controlled by attrition as expressed in terms of parameters such as intantaneous force ratios, loss rates, and loss thresholds (see, e.g., Dare and Thomas (1975), Dare (1979), Faddy (1975), Goad (1979)).

An example for a planning model is discussed by Schmitz (1986). Designed as a linear program for the determination of the "optimal" land force weapon mix for countering a given threat, the coefficients for its objective functions are to be determined from

a hierarchy of combat simulations models the resolution of which reaches down to the level of individual weapon systems.[2]

These examples illustrate why the defense analysis community started to question the viability of its models, but also of its style, when the magnitude of the changes following the fall of the Berlin wall became apparent (see, e.g., Hillestad et. al (1992), and Huber (1990)). However, it is not the intent of this paper to present another critique of the presumed shortcomings of the old models in the new world. Rather, it takes a look at why *variable-resolution* would be a highly desirable feature of models for today's and tomorrow's defense analysis.

2. The New Situation and the New Issues

As of the day this paper is written, the strategic situation may be characterized by the following facts:

1) The principal threat that represented the yardstick for almost all Western defense efforts since World War 2 has disappeared. The likelihood of a large scale military invasion of NATO territory from the East is practically zero. Thus, there is strong public pressure in most countries of the alliance to cash the peace dividend and reduce defense spending.

2) The Soviet Union has been replaced by 15 independent states. 11 of them have agreed to coordinate their economic and security policies within the Commonwealth of Independent States (CIS). Almost all of the new states want to maintain their own armed forces. Four of them (Russia, Belorus, Ukraine and Kazakhstan) have nuclear weapons stationed on their territories the control of which is still largely unresolved.

---

[2] The "balanced" force mix solutions offered by this planning model do not imply robust solutions with a view to scenario uncertainty. Rather, they represent compromise solutions satisfying, for one given scenario, different objective functions "to the greatest possible extent", i.e., the values for the objective functions must not fall below specified bounds (Schmitz et al. (1986), p. 163).

3) As a consequence of the collapse of communism in Eastern Europe and the former Soviet Union, there is a resurgence of national, ethnic and religious tensions. Exacerbated by the poor economic conditions in the region, they imply a severe risk of escalation affecting the security of regional neighbours, and even of Western European countries directly and indirectly.[3]

4) Chaotic developments in the former Soviet Union might generate sufficient public support for another coup by conservative and/or nationalistic forces aimed at re-establishing the Soviet Union or another centrally controlled superstate.

5) Irrespective of the "lessons" of the Gulf War, military threats to Western security interests outside of Europe will remain a major cause of concern, especially when considering the proliferation of advanced military technology around the world.

6) This is also true for threats to Europe and North America from without, including terrorism as an instrument of international blackmail. Given the availability of weapons of mass destruction, international terrorism may assume an entirely new dimension.

These facts underline the high degree of uncertainty of today's global politico-military environment, the encouraging signs of a new type of security cooperation among the nations of Eurasia and North America notwithstanding. Both, the uncertainty of developments as well as the cooperative approach to security still to be evolved, confront defense analysis with a host of new issues which reach beyond purely military concerns, and compared to which those of the bipolar world appear, retrospectively, as having been rather marginal in nature. For example, the fundamental problem of defense planning in the Cold War period boiled down to the question of how more or less well defined military requirements could best be met. Today, defense analysis must first deal with

---

[3] In this context, the control of nuclear warheads in the former Soviet Union is a special problem of immediate concern to all countries of the entire northern hemisphere. In addition to long-lasting damage from fallout, their employment in civil war-like conflicts between antagonistic parties within the former Soviet Union could cause a panic of unprecedented proportions among the peoples of Eastern Europe, and tidal wave of refugees flooding Central and Western Europe.

the question of how to define the requirements, in a highly transparent manner that accounts for the uncertainties of the new multipolar environment, before it may go about looking at ways to meet them.[4]

Of course, the uncertainties of the political environment notwithstanding, requirements can only be defined if assumptions are made about the capabilities of all military actors one may eventually be confronted with in a multipolar international system. In other words, the requirements are contingent to these assumptions. And since this is true for all military actors within the system, defense requirements analysis becomes more or less tantamount to an analysis of the interdependencies of the military capabilities of the entire international system. For example, for military actors within the CSCE-area, requirements analysis would involve at least four fundamental questions to be answered consecutively (see also Huber and Schindler (1992)):

1) Given the conventional out-of-area contingencies, what is the minimal *offensive* potential that the parties to the system, or certain subgroups of parties (such as, e.g., NATO or WEU), must be able to provide for the protection of their security interests through the threat of intervention?

2) Given the offensive military potentials of all parties and their build-up rates, a) can the military requirements for out-of-region and out-of-area contingencies be met? b) what is the minimal *defensive* potential each party needs to have, and/or must be able to build up to, in order to assure *internal military stability* by being capable of credibly deterring, at all times, all potential aggressors from within the system, considering existing alliances and *ad hoc* coalitions that may eventually be formed given the uncertainties of the political developments within the CSCE-area?

3) What are the quantitative and qualitative (i.e., numerical and structural) constraints to be imposed on the offensive

---

[4] In reality, both questions need to be looked at simultaneously or iteratively. This is because requirements do depend on the alternatives for meeting them, in particular when constraints limit the available choices.

capabilities of all parties, and how must their defensive capabilities be improved, in order to assure for each party that its actual defensive potential is at least equal to the minimal defensive potential required for a credible deterrence of potential aggressors?

4) Given the minimal offensive potential required for (collective) out-of-region and out-of-area operations, and given the minimal defensive potentials and offensive constraints for assuring internal stability, what would be the fair share that each member of the CSCE should contribute to the cause of common security?

These questions illustrate that the character of requirements analysis has changed from being primarily *tactical*-level analysis during the days of the bipolar East-West confrontation to include *operational* and *strategic*-level analysis in the new multipolar world, extending beyond strictly military considerations to account for political, economic, ethnic and other non-military factors (e.g., in terms of potential aggressors and *ad hoc* coalitions).

3. Demands on Analysis and Variable Resolution Models

Summarizing the above, it is concluded that, in the future, defense analysis will be characterized by 1) reduced defense budgets; 2) a considerably extended dimension owing to the high degree of scenario uncertainty; 3) a greatly extended scope (big questions!); 4) shorter deadlines; 5) a much wider purpose, including that of serving as a *confidence and security building measure (CSBM)* in its own right and of providing the analytical framework for cooperative security and crisis and conflict management. This is essentially equivalent to saying that problems of much higher complexity must be analyzed faster, with fewer resources, and in a transparent manner conducive to interdisciplinary and international dialogue.

As a consequence, it will not be possible anymore to develop tailor-made models for extensive experimentation in most cases. Rather, analysts must increasingly rely on knowledge bases

which aggregate or disaggregate military orders so that they address entities at whatever level the simulation takes place. Either way would permit commanders and staffs to communicate, with the simulated superordinate commanders, neighbours, and subordinate units, in a rather realistic manner. Other than for widening the circle of experts available to the military knowledge engineers, this is even more important with a view to the trend that life exercises and maneuvers have to be increasingly replaced by computer-based exercises, due to limited budgets and public pressure, especially in the densely populated areas of Western and Central Europe.

3.2 Minimal Complexity of Models

In order to assure efficient processing of large numbers of parametric variations of scenario parameters and tactical/operational circumstances, the aggregation level of the simulation models ought to be as high, or equivalently, the resolution level as low as the objectives of the respective study permit. In other words, the simulation systems need to be designed in a manner permitting to configure the "simplest" model for the analysis purpose at hand.[9]

For example, in a study aimed at determining the effectiveness of deep air interdiction, a highly aggregated differential combat model may be adequate to represent ground combat between the front-line forces of both sides. However, because of the item level dynamics of the interactions of aircraft and air defense systems, an explicit representation at the item level may be required for air-to-air and ground-to-air combat functions (see Parry (1984), p. 555). However, one may want to temporarily replace, in a simulation, the aggregated ground combat model by a more detailed, higher resolution simulation in order to capture the synergistic effects of artillery strikes or close air support attacks.

---

[9] In addition to efficient processing, simple models also offer better tractability of results, thus greatly facilitating their usefulness as tools of debate among decision makers and experts of different disciplines and cultural backgrounds (see also Miller (1982), p. 368).

generated, in an anticipative manner, from the evaluation of simulation experiments performed on a continuous basis, similar to the way in which basic research operates.[5] For *ad hoc* up-dating of the knowledge bases, readily available operational models will have to be used that require no or only small adaptions to fit the problem at hand. In addition, the models must be designed in a manner that permits to process, within relatively short spans of time, large numbers of scenarios under a wide range of circumstances in any region relevant to the analysis issue, either directly or indirectly.[6]

It goes without saying that these requirements may only be met by analytical or *closed* simulation models, i.e., models without human decision makers in the command and control loop, the complexity of which can be kept at the lowest possible level compatible with the objectives of the analysis at hand. Therefore, *variable resolution* appears to be a highly desirable, if not necessary, feature of these models for the reasons explained below.

3.1 Closed Simulation Models

In closed combat simulation models, military decisions are usually generated by means of empirically-based rules which prescribe the actions to be taken when certain events occur or conditions are

---

[5] To support this research, training games for military staffs on all levels should be designed so that their (routine) evaluation would contribute to up-dating the knowledge on the forces in being.

[6] For example, Huber (1992) has proposed a model for sizing and structuring the conventional forces in a multipolar environment which shows, among other things, how the size and military potential of one actor (such as, e.g., the Ukraine) affects the potential that all the others (e.g., NATO or Germany) have to provide in order to warrant military stability in the Eurasian international system. An essential input to this model is the so-called *stable regional force ratio* (SRFR) between all potential military antagonists (including, e.g., between Armenia and Azerbaijan) that may be estimated by means of the analytical *Generalized Force Ratio Model* (GEFRAM; Huber (1990,1992)) which, in turn, requires input data generated from more or less detailed simulation experiments on combat between the respective antagonists, given their force structures, operational philosophies, tactics, equipment, and the respective terrain.

met.[7] These rules tend to reflect *current* military expertise and tactical/operational philosophies. Thus, they are not necessarily efficient when applied under circumstances other than those underlying their original formulation. Furthermore, even if they adhere to the same school of operational philosophy and principles, military experts often disagree on rules to be employed in specific situations. For both of these reasons, rule sets need to be developed iteratively, with current military expertise providing the basis for initial rule hypotheses which are reviewed and adapted in the light of their impact on simulation results.

Thus, the iterative development of *robust* decision rules requires not only a wide range of tactical/operational conditions to be covered, but also as large a number of military experts as possible to be involved. To this end, it appears indispensible that (routine) training exercises of military staffs will be used for that purpose (see also footnote 5). However, for computer-based exercise systems to provide a realistic testbed in the military knowledge engineering process, the architecture of their simulation models must be flexible enough to accommodate different expertise and individual styles of military experts. In particular, the simulation models must be capable of coping with any level of aggregation at which the experts choose to articulate their expertise, and which may be different in different functional areas.[8]

This does not necessarily require that the simulation takes place at the different levels of aggregation at which the experts or commanders address the system in each of the functional areas. It would also be accomplished by appropriate mechanisms or links

---

[7] Only in highly simplified analytical models of combat may mathematical optimization techniques be employed for modeling tactical or operational decision processes (see, e.g., Berkovitz and Dresher (1959), Weiss (1959), Dresher (1971), Huber (1978)).

[8] This feature would also be highly desirable given the fact that future military conflict scenarios can be expected to involve coalition forces formed ad hoc in a crisis, which need to reconsider and adapt their national command and control philosophies to fit the requirements of coalition warfare as efficiently as possible. On a less demanding time scale, the same holds true for the international *rapid response forces* being proposed by NATO for the purpose of providing credible capabilities for collective out-of-region and out-of-area intervention.

Similarly, in a low resolution airland combat simulation at theater or corps level, area air defense systems may be represented in a highly aggregated manner such as, e.g., in terms of the *aircraft sortie attrition rates* in the respective areas which had been determined a priori from high resolution item level simulations, given certain states of the respective air defense systems (centralized or decentralized control; degrees of suppression; ECM-states; etc.). Whenever events occur, during the simulation at theater or corps level, that change the state of the air defense systems to an extent not reflected by any of the previously considered states, the low resolution simulation is triggered for *ad hoc* generation of the attrition rates that characterize the new state.[10]

The capability for invoking high resolution models within combat simulations at a higher level of aggregation will be necessary the more, the less resources can be devoted to anticipative research aimed at generating, from high resolution simulation, the inputs required for coping with the high degree of scenario uncertainty in analysis. In addition, considering the ever more limited availability of analysis funds, it would seem logical to exploit any ongoing analysis or computer-based exercise for reviewing and extending the data bases for models and the knowledge base for analysis.

3.3 Integrated Model Hierarchies

As Davis and Huber (1992) point out, the general notion of *variable resolution combat modeling* has existed since at least the early 1980s. Its origins are related to a modeling concept called the *hierarchy of models* pursued since the mid-1970s by several analysis institutions in Europe, notably the British DOAE and the German IABG (see Dare (1979) and Niemeyer (1979)), in order to keep the complexity of combat models within manageable bounds as the scope of analysis extended from weapon-on-weapon duels,

---

[10] It should be noted, however, that by invoking a higher resolution model within combat simulations at a lower resolution level one does not merely continue the simulation at the higher resolution level. Rather, there will have to be a series of simulation runs in order to capture the uncertainies associated with the higher resolution scenario as well as the stochastic effects of the involved combat processes.

through tactical and operational level combat, to strategic level conflict. The idea was that simulation models developed for stand-alone use at the separate levels would be *externally linked* through data. In this manner, a model at any given level would provide inputs for the simulation experiments at the next higher level, and scenario parameters for the simulations at the next lower level. Thus, the *hierarchy of models* concept was not meant to, and did not, provide variable resolution in the sense postulated above. To this end, the models need to be *integrated* into the hierarchy, i.e., they must be *internally linked* and, most importantly, they must be based on a common *single architecture* for all models.[11]

Based on mid-1980s research on the *Rand Strategy Assessment System* (RSAS), Davis concluded that if models were designed in a quasi rigorously hierarchical manner, different resolutions would be achieved simply by truncating or expanding the model tree at the desired level. This is illustrated by Fig. 3.1 which shows the basic architecture of the analytical model GEFRAM (see footnote 6). At the highest level (lowest resolution) it provides analysts, or participants in a wargame using the model as a decision support system, with information about the force ratio in main-thrust sectors, the implications of which military commanders and many analysts feel they understand intuitively. This, however, is determined by a mass of underlying data based on expert judgement and/or simulation experiments. In the extreme, the model may be operated at a level that makes use of payload information and

---

[11] The distinction between internal and external linkage of models in hierarchies was first drawn in a 1982 NATO conference surveying the state of the art in combat modeling (Huber (1984)). The model hierarchies presented at the conference, by the UK's DOAE and Germany's IABG, were externally linked. Internal linkage was discussed for the model hierarchy concept proposed by the US Army's Model Improvement Program (AMIP). However, its internal linkage not withstanding, AMIP did not feature variable resolution, primarily because the models in the hierarchy had been independently developed and employed different methods of aggregation and time representation. Therefore, it turned out to be extremely difficult to develop consistent supporting data bases for the various models in the AMIP hierarchy. In order to overcome these difficulties, Parry (1984) proposed the concept of a *self-contained hierarchical architecture* that would 1) be capable of representing airland combat from the theater to item level; 2) allow for substitutable modules; 3) permit representation of functional areas at *variable resolution* (pp. 554-555).

SSPKs against typical ground force elements (E), or even at a still lower level using range-payload relationships, information about the average distance from bases to targets, about conditional PK-values and delivery tactics. However, one may instead want to specify, at a higher level, the per sortie effectiveness in killing ground force elements or, given the ground force composition, the average ground force potential killed per sortie.

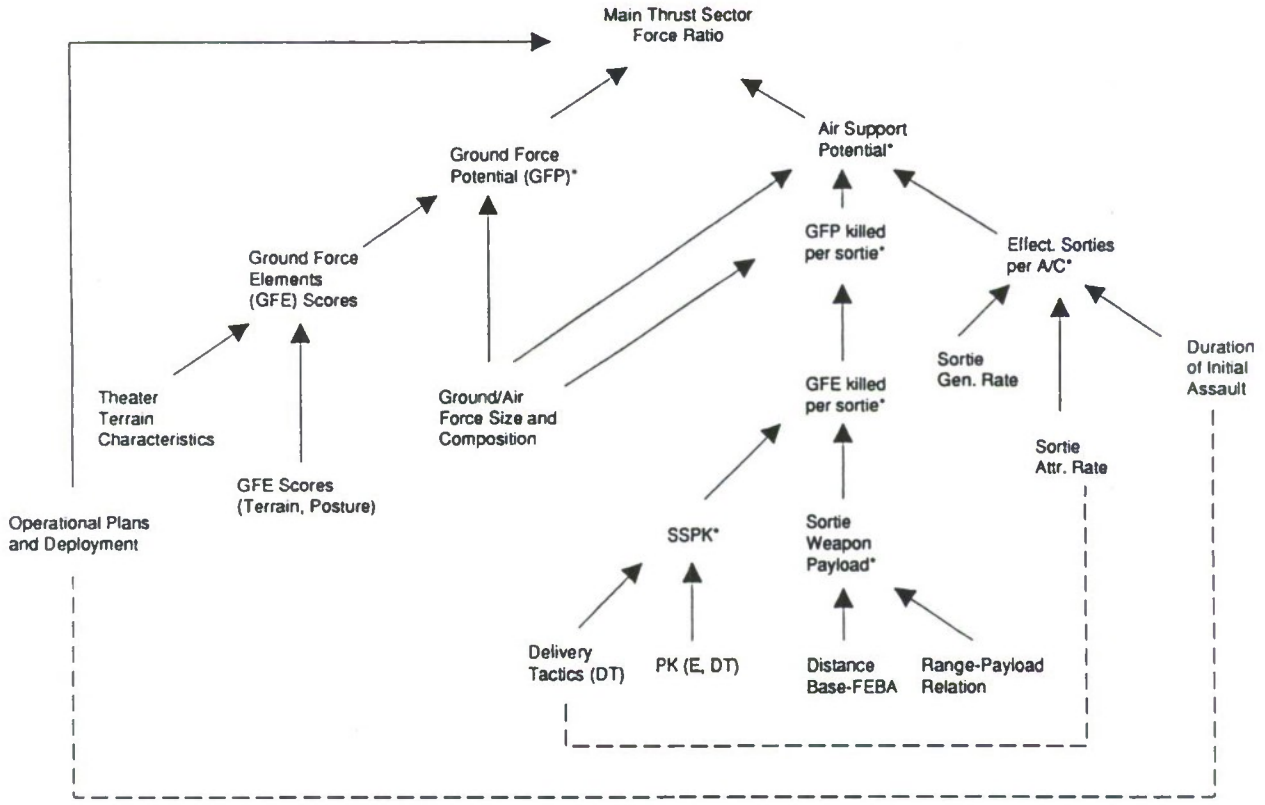


Fig. 3.1: The Generalized Force Ratio Model (GEFRAM) with variable resolution

In standard situations familiar to the analyst it may well be possible to operate the model on a fairly high level if there is sufficient empirical information for specifying the required data. However, it is unlikely that this would be the case if the theater in question differed considerably from the one underlying the standard situations. For example, out-of-area operations might

reduce the sortie generation rate to some degree, depending upon the time available to build up the logistics support organization. Thus, the data on effective sorties per aircraft would have to be reviewed in terms of how they are affected by the sortie generation rates, given the duration of the ground force operations to be supported, and given the sortie attrition rate. In addition, the assumptions about these data may need substantiation based on still lower resolution simulations of the ground support processes, of the ground force battles in main-thrust sectors, and of the air defense interactions.

As defense planners will increasingly be confronted with new and unfamiliar scenarios, the capability to change, more or less *ad hoc*, to fairly high resolution levels in the analysis becomes more important. This is because of the need for testing the compatability of *a priori*-data on a given level of resolution. For example, the parameter *GFE killed per sortie* depends, among other things, on the delivery tactics as well as on the distance to be covered by the sorties over unfriendly territory. Since both of the latter parameters affect the *sortie attrition rate* as well, there may be an incompatability between the available data on *sortie attrition rate* and *GFE killed per sortie*. The same may be true for estimates on the duration of initial assaults and the presumed operational plans.

3.4 The Down-Link Problem

Of course, the incompatability of data items at a given level is an indication that reality may not be captured very well by models of a strictly hierarchical design. There are many cross-relationships between the branches of the hierarchical tree. Unless they can be broken by appropriate approximations (see Davis and Huber (1992)), they must, at the very least, be pointed out to the model user by specifiying all data affected by a specific change of resolution. Better yet, the model might be designed in a manner that the appropriate configuration for *ad hoc* compatability testing is automatically generated. This is primarily a problem of developing *down-links* capable of generating, without manual intervention, the initial situations for the higher resolution simulations from the data resulting from the higher level (lower resolution) simulations (see Parry (1984), p. 556).

The difficulties to be expected in the design of down-links obviously depend on the nature of the models to be linked and on the distance of aggregation levels to be bridged. The experience gained sofar with the model hierarchy developed at this author's institute (see Hofmann and Huber (1990)) seems to indicate that hierarchical models of air and air defense operations are much more benign in this respect than those of ground combat operations. For example, invoking the high resolution (Btl/Rgt level; single shot resolution) simulation model BASIS, within a simulation with the low resolution (theater or corps level; Btl/Cp resolution) model KOSMOS would require, among other things, the automatic generation of 1) the initial positions for all individual weapon systems in a digitized map of the respective terrain (with a horizontal resolution of 50 by 50 m and a vertical resolution of 10 cm) (see also footnote 10); 2) the *tactics programs* which specify the tactical plans for both, attacker and defender, as well as a set of conditional orders telling each weapon system or unit what to do when certain evens occur, such as changing positions when losses reach a given threshold.[12]

Even though we have not yet been able to devote much effort to the question of automating the down-link from KOSMOS to BASIS, the fact that it took experienced military commanders several iterations to specify mutually satisfactory tactical plans for BASIS experiments indicates that the development of plausible and sufficient sets of down-link rules is not a trivial matter, especially when terrain/tactics interactions dominate the selection of positions and tactical plans. Our experience also suggests that the underlying military decision processes may have to be modeled as multidimensional stochastic processes characterized by more or less strongly correlated variables (see Hofmann and Huber (1990)).

---

[12] For a description of BASIS, the reader is referred to Hofmann et al. (1986) and Schaub (1991). KOSMOS is discussed by Hofmann and Rochel (1990) and Rochel (1990).

4. Summary

In the foregoing paper, the attempt was made to explain why the global strategic environment that has emerged since the end of the Cold War is likely to reinforce the need for variable resolution combat simulation beyond the reasons discussed by Davis and Huber (1992) and Parry (1984). In addition to the high degree of scenario uncertainty and the significantly extended scope of issues, in conjunction with reduced resources available for defense analysis, it is the *multi-purpose* application by rather *diverse users* that makes variable resolution a highly desirable, if not necessary, feature of combat simulation models. In fact, the implementation of a rational approach to collective security and crisis management may well depend on the availability of variable resolution models that permit debating issues in a structured manner, at whatever level the political and military decision makers may choose in a multi-national environment.

References

Berkovitz, L.D., Dresher, M. (1959): A Game Theory Analysis of Tactical Air War. *Opns. Res.* 7, pp. 599-620

Dare, D.P. (1979): On a Hierarchy of Models. In: *Operational Research Games for Defense* (R.K. Huber, K. Niemeyer, H.W. Hofmann, Eds.), München: Oldenbourg, pp. 285-307

Dare, D.P., Thomas, G.S. (1975): The NATO Deployment Model. In: *Military Strategy and Tactics* (R.K. Huber, L.F. Jones, E. Reine, Eds.), New York: Plenum, pp. 243-259

Davis, P.K., Huber, R.K. (1992): Variable-Resolution Combat Modeling: Motivations, Issues, Principles. RAND-Report N-3400-DARPA, Santa Monica

Dresher, M. (1971): The N-Stage Game and Lagrangian Multipliers; The N-Stage Game and DYGAM. SRI/NWRC Tech. Note 61

Faddy, D.L. (1975): A Review of the Land Battle Models used at DOAE and some Application. In: *Military Strategy and Tactics* (R.K. Huber, L.F. Jones, E. Reine, Eds.), New York: Plenum, pp. 171-180

Goad, R. (1979): The Modelling of Movement in Tactical Games. In: *Operational Research Games for Defense* (R.K. Huber, K. Niemeyer, H.W. Hofmann, Eds.), München: Oldenbourg, pp. 190-214

Hillestad, R., Huber, R.K., Weiner, M. (Eds., 1992): New Issues and Tools for Future Military Analysis: A Workshop Summary. RAND-Report N-3403-DARPA, Santa Monica: RAND Corp.

Hofmann, H.W., Huber, R.K., Steiger, K. (1986): On Reactive Defense Options - A Comparative System Analysis of Alternatives for the Initial Defense Against the First Strategic Echelon of the Warsaw Pact in Central Europe. In: *Modeling and Analysis of Conventional Defense in Europe* (R.K. Huber, Ed.), New York-London: Plenum Press, pp. 97-140

Hofmann, H.W., Huber, R.K. (1990): Zur Architektur eines operationsanalytischen Modellinstrumentariums für die sicherheits- und verteidigungspolitische Analyse und bereichsübergreifende Planung des BMVg. Bericht Nr. S-9003, Institut für Angewandte Systemforschung und Operations Research, Fakultät für Informatik, Universität der Bundeswehr München, August

Hofmann, H.W., Rochel, T.H. (1990): Modelling of C^3I in a Closed Land Combat Simulation System at Corps/Army Level (KOSMOS). In: Proceedings from the Symposium on *Command, Control and Communications in Combat Models and Games*, Report AC/243 (Panel 7) TP/2, Vol. 1, Brussels: NATO

Huber, R.K. (Ed., 1984): *Systems Analysis and Modeling in Defense - Development, Trends, Issues*, New York: Plenum

Huber, R.K. (1990): An Analytical Approach to Cooperative Security: First Order Assessment of Force Posture Requirements in a Changing Security Environment. In: *Military Stability - Prerequisities and Analysis Requirements for Conventional Stability in Europe* (R.K. Huber, Ed.), Baden-Baden: NOMOS Verlagsgesellschaft, pp. 165-184

Huber, R.K., Schindler, O. (1992): Multipolar Stable Defense - An Analytical Framework for Its Design. Bericht Nr. S-9203, Institut für Angewandte Systemforschung und Operations Research, Fakultät für Informatik, Universität der Bundeswehr München, March

Miller, R.S. (1982): The Assessment of Command and Control. In: *Führungs- und Informationssysteme* (H.W. Hofmann, R.K. Huber, P. Molzberger, Eds.), München: Oldenbourg, pp. 325-369

Miser, H.J. (1969): Operations Reserach and Systems Analysis, IIASA-RR-81-9, May

Niemeyer, K. (1979): Zur Struktur einer Hierarchie von Planspielmodellen. In: *Operationsanalytische Spiele für die Verteidigung* (R.K. Huber, K. Niemeyer, H.W. Hofmann, Eds.), München: Oldenbourg, pp. 308-324

Parry, S.H. (1984): A Self-contained Hierarchical Model Construct. In: *Systems Analysis and Modeling in Defense* (R.K. Huber, Ed.), New York: Plenum, pp. 547-558

Rochel, T.H. (1990): Zur Architektur geschlossener Gefechtssimulationsmodelle höherer Abbildungsebene unter besonderer Berücksichtigung der Modellierung und Implementierung von Führungsprozessen. Dissertation, Universität der Bundeswehr München

Schaub, T. (1991): Zur Aggregation heterogener Abnutzungsprozesse in Gefechtssimulationsmodellen. Dissertation, Universität der Bundeswehr München

Schmitz, W., Reidlhuber, O., Niemeyer, K. (1986): Some Long-term Trends in Force Structuring. In: *Modeling and Analysis of Conventional Defense in Europe* (R.K. Huber, Ed.), New York: Plenum, pp. 141-166

Waddington, C.H. (1973): *OR in World War 2 - Operational Research against the U-Boat*, London: Elek Science

Weiss, H. (1959): Some Differential Games of Tactical Interest and the Value of a Supporting Weapon System. *Opns. Res.* 7 (1959), pp. 180-195



APPLICATIONS OF HIGH AND LOW RESOLUTION MODELS IN CROSS RESOLUTION ANALYSIS

5 MAY 1992

W. PETER CHERRY

VECTOR RESEARCH, INCORPORATED
P.O. BOX 1506
ANN ARBOR, MICHIGAN 48106

SOME VRI STUDIES IN THE NEW POLITICAL-MILITARY ENVIRONMENT

CONVENTIONAL FORCES IN EUROPE REDUCTIONS (CFE-1)	(MAY - SEP 1989)
US AND NATO MODERNIZATION ISSUES	(JAN 1990)
US ARMY END STRENGTH VERSUS RDA: RISKS AND PAYOFFS	(APR 1990)
POST CFE OPERATIONAL CONCEPTS	(MAR - JUNE 1990)
ASSESSMENT OF TOTAL FORCE ALTERNATIVES	(JULY 1990)
US WARFIGHTING CAPABILITY IN SOUTHWEST ASIA	(AUG - NOV 1990)
NATO MULTINATIONAL RAPID REACTION FORCE	(DEC 1990 - MAR 1991)
US WARFIGHTING CAPABILITY IN LATIN AMERICA	(APR - JULY 1991)
ROK ARMY MODERNIZATION NEEDS	(SEP - OCT 1991)

–CONTINUED –

SOME VRI STUDIES IN THE NEW POLITICAL-MILITARY ENVIRONMENT

(CONCLUDED)

SYSTEM SPECIFIC STUDIES **(AUG 1989 - PRESENT)**

- AH-64 - AH64/LB
- RAH-66/LB
- ASM
- M1A1, M1A2, BLOCK III
- AFAS
- MLRS, ATACMS-I, II
- UAV, JSTARS
- AAWS-M
- NLOS
- LOSAT
- ATM
- ARMY INTRA-THEATER AIRLIFTERS

LRAMRP EVALUATION AND ANALYSIS **(APR - JULY 1991)**

ARMY MOBILIZATION CAPABILITIES **(MAY 1991 - PRESENT)**

ARMY MODERNIZATION INVESTMENT STRATEGY **(JULY - OCT 1991)**

ARMY FORCE STRUCTURE REDUCTIONS **(SEP - DEC 1991)**

VRI HIGH LEVEL ANALYSES

- A LOW RESOLUTION MODEL IS GENERALLY A MAJOR TOOL:
  - DECISION ISSUES
  - TIME AVAILABLE
  - PRECISION

- LOW RESOLUTION MODEL IS TIED TO A HIGH RESOLUTION SIMULATION:
  - STATISTICAL FIT
  - UPDATE AND VERIFICATION -- "RE-FIT"

- CRITICAL ISSUES ON OCCASION ADDRESSED IN HIGHER RESOLUTION, NARROWER SCOPE:
  - DIGITAL TERRAIN
  - ITEM LEVEL REPRESENTATION



BACKGROUND

-- MACRO AS AN ANALYSIS TOOL --

INTEGRATING TOOL

QUICK TURNAROUND

BROAD DIRECTIONS CONCERNING MODERNIZATION, FORCE STRUCTURE, ETC.

TOP-LEVEL LOOK -- MORE DETAIL WITH VIC, VECTOR-3, . . .

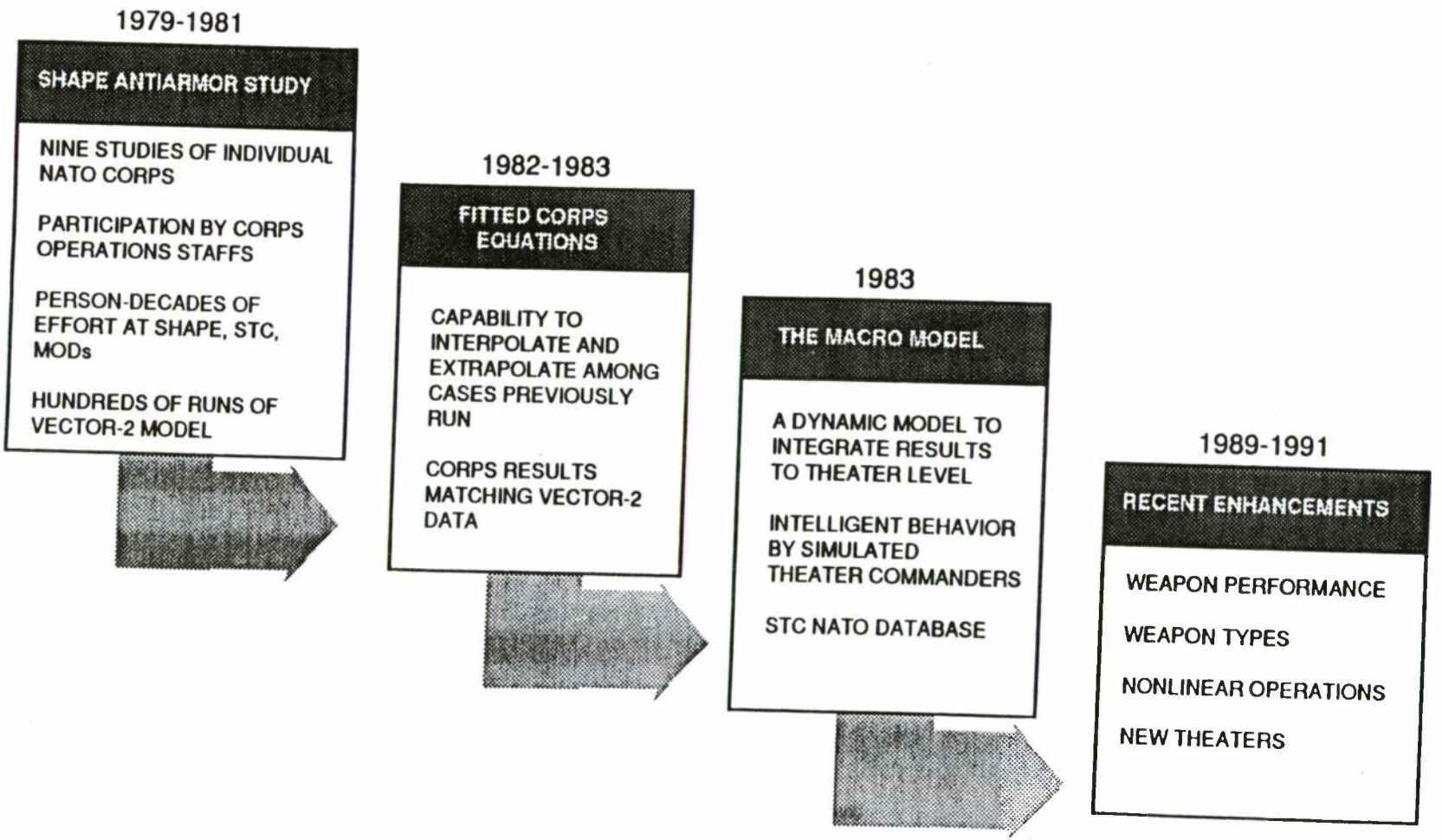
SUPPORTS ANALYSTS' USE OF MILITARY EXPERTISE AND LOGIC

NOT A STAND ALONE MODEL

AGGREGATE AND RELATIVELY ABSTRACT

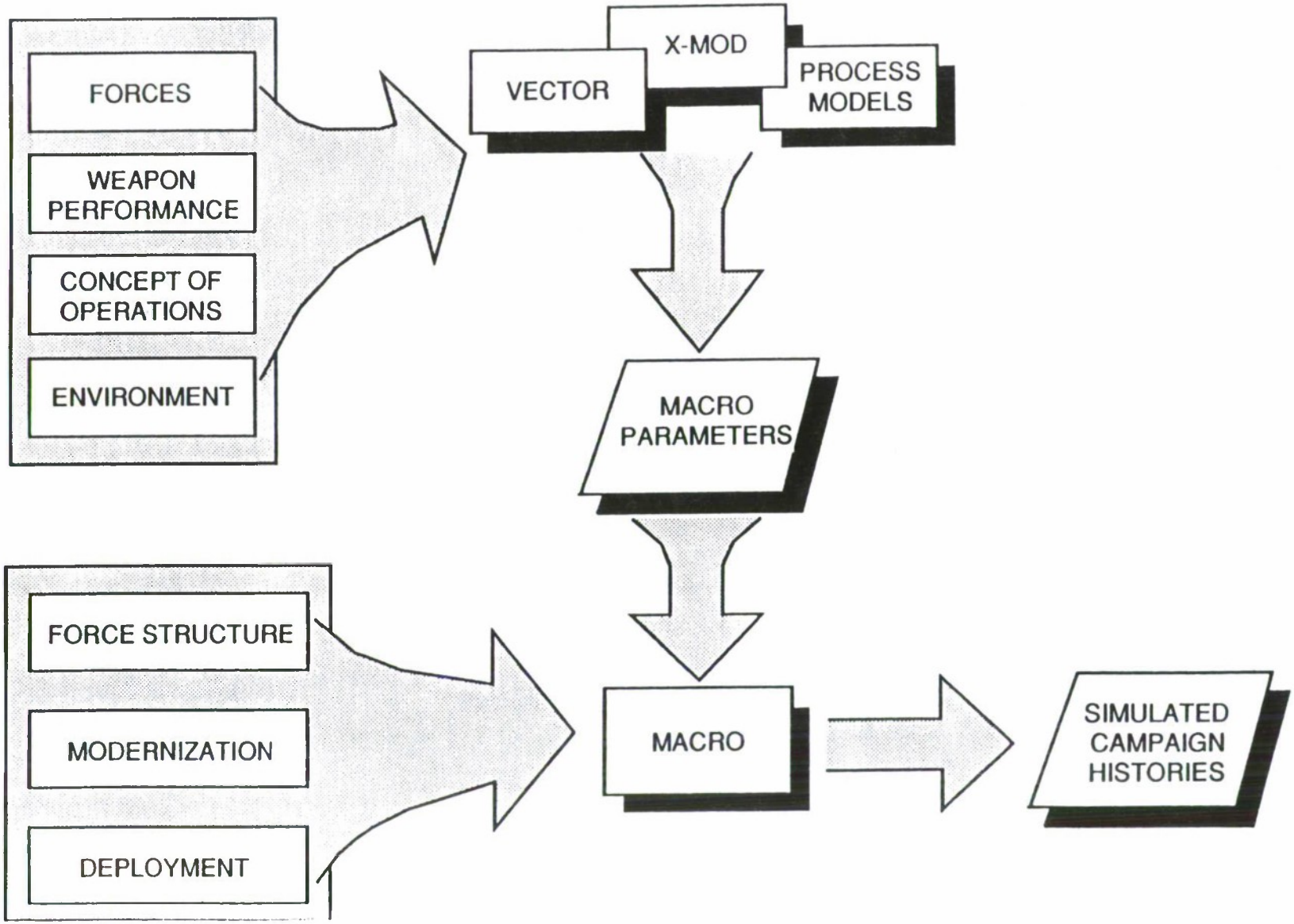
BACKGROUND

-- HISTORY OF MACRO DEVELOPMENT --



THE REFITTING PROCESS

-- MACRO STUDY PROCESS --



THE VECTOR CAMPAIGN MODEL

REPRESENTS THE SPECTRUM OF ACTIVITIES AND PROCESSES INVOLVED IN A THEATER-LEVEL, TWO-SIDED, AIRLAND CAMPAIGN. THIRTY-SECOND TIME RESOLUTION FOR SOME ACTIVITIES

SCOPE: THEATER, ARMY GROUP, CORPS, OR DIVISION LEVEL

INCORPORATES MODULAR, DETAILED MODELS OF PHYSICAL AND BEHAVIORAL PROCESSES. EXPLICIT REPRESENTATION AND RESOLUTION TO INDIVIDUAL WEAPON/ITEM SYSTEMS

AUTOMATED ARTIFICIAL INTELLIGENCE ("EXPERT SYSTEM") MODULES FOR RESOURCE ALLOCATION AND TACTICAL DECISION MAKING

GENERATES A DETAILED HISTORY OF THE CAMPAIGN

VECTOR-2 SUCCESSFULLY TESTED AGAINST RESULTS OF THE GOLAN HEIGHTS CAMPAIGN IN THE 1973 ARAB-ISRAELI WAR

VECTOR-3 IS ONE OF THE LATEST IN THE VECTOR MODEL SERIES

FEATURES OF VECTOR-3

CONSISTENT WITH ARMY'S CORPS-LEVEL VIC, BUT WITH SOME FEATURES NOT AVAILABLE IN VIC

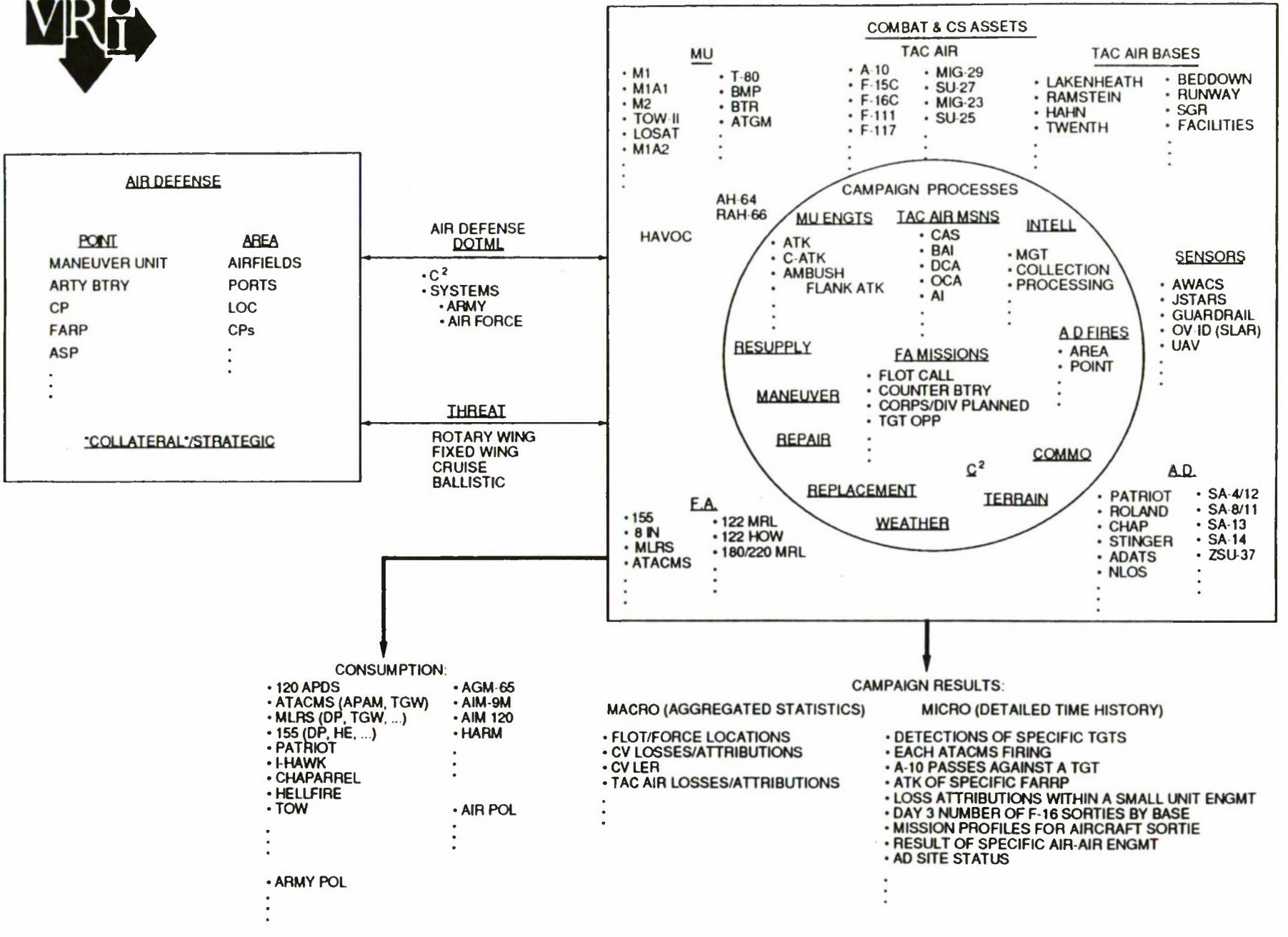
BALANCED REPRESENTATION OF ARMY AND AIR FORCE ASSETS AND OPERATIONS, WITH FULL REPRESENTATION OF THEATER AIR WAR:

- EXPLICIT LOCATIONS AND CHARACTERISTICS OF REAL AIRFIELDS
- MISSION APPORTIONMENT AND ALLOCATION
- SORTIE GENERATION
- AIRCRAFT FLIGHT
- MISSION EFFECTS FOR CAS, BAI, AI, SEAD, OCA, DCA, BARCAP, ESCORT, SURVEILLANCE, AND TRANSPORT MISSIONS
- AD SENSORS, C , SYSTEMS

EXPLICIT AND DETAILED REPRESENTATION OF REAR-AREA OPERATIONS:

- DETAILED REPRESENTATION OF LOCATION AND MOVEMENT OF RESERVE AND SECOND ECHELON MANEUVER UNITS, AND LOCATION AND EMPLOYMENT OF COMBAT SUPPORT ASSETS
- EXPLICIT REPRESENTATION OF TERRAIN AND OF GROUND TRANSPORTATION NETWORK
- REPRESENTATION OF COMBAT ENGINEERS PERFORMING MOBILITY, COUNTERMOBILITY, AND SURVIVABILITY MISSIONS
- REPRESENTATION OF CSS FACILITIES AND PROCESSES (TRANSPORT, SUPPLY, MAINTENANCE)

ANALYSIS OF AIR DEFENSE USING VECTOR-3



OTHER EXPERIENCE

- SYSTEM EFFECTIVENESS:
  - LOW RESOLUTION HIGH LEVEL WARGAME TO TEST CONCEPTS OF OPERATION IN THEATER CONTEXT
- ALGORITHM PERFORMANCE:
  - ENTITY-ACTIVITY-OBSERVABLE-SIGNATURE PATTERNS DRIVEN BY CORPS LEVEL CAMPAIGN MODEL TO TEST INTELLIGENCE/FUSION
- TEST AND EVALUATION:
  - "ACCELERATED TIME" COMBAT MODELS TO TEST FIELD ARTILLERY TACTICS AND COMMAND CONTROL, AIR DEFENSE PERFORMANCE
- STRATEGY AND TACTICS:
  - HIGH RESOLUION AIR CAMPAIGN TO EXAMINE "OPTIMAL" STRATEGIES AND TACTICS



REALIZATIONS FROM CENTRAL VALUES

-- SOME PROBLEMS TO BE SOLVED --

- DECISION THRESHOLDS:
  - CHANGE CONCEPT
  - EXECUTE CONTINGENCY
  - EXERCISE DISCRETION
- TEMPORAL AND SPATIAL CORRELATIONS:
  - AGGREGATE-DISAGGREGATE-AGGREGATE-DISAGGREGATE
  - SIMULTANEOUS/NEAR SIMULTANEOUS OBSERVATION BY DIFFERENT OBSERVERS
  - INTERRUPTED OBSERVATION BY SAME OR DIFFERENT OBSERVERS
- TEMPORAL AND SPATIAL SIGNATURES:
  - WAKES
  - OFFSETS



THE USES OF CROSS RESOLUTION MODELS AND ANALYSIS

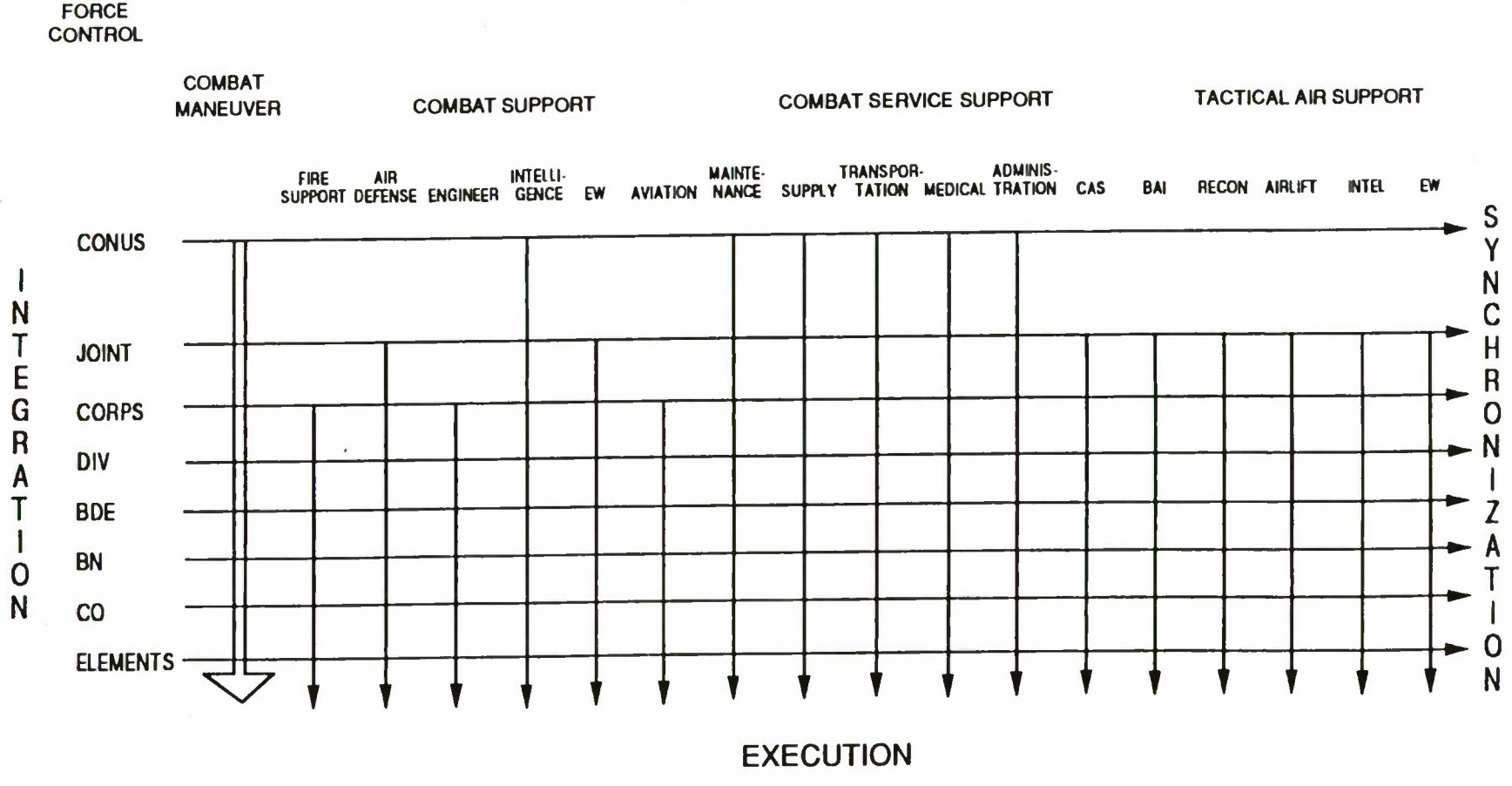
- CONTEXT AND CONSISTENCY:
  - INITIAL CONDITIONS
  - BATTLEFIELD FUNCTIONAL AREAS
  - ADJACENT UNITS
  - SUPERIOR UNITS

- CONFIDENCE:
  - WEAPON SYSTEMS
  - TACTICS AND DOCTRINE
  - TERRAIN AND WEATHER
  - COMMAND AND CONTROL

- CRITICAL TIME AND PLACE:
  - EVENT
  - ACTIVITY
  - CAUSE AND EFFECT

TACTICAL SYNCHRONIZATION

-- THE C^2 MATRIX[1] --



[1] DePUY, WILLIAM, "CONCEPTS OF OPERATION: THE HEART OF COMMAND, THE TOOL OF DOCTRINE", **ARMY**, AUGUST 1988, pp. 26-40.



C^2 PROCESSES

-- UNDERSTANDING THE IDEAS WHICH LIE AT THE HEART OF THE C^2 PROCESS --

- DEVELOPING NESTED CONCEPTS OF OPERATION: THE CRITICAL PATH
- SPECIFYING TASK ORGANIZATION
- SPECIFYING/CLARIFYING COMMAND RELATIONSHIPS
- ESTABLISHING CROSS-SERVICE PROCEDURES AND ORGANIZATION

-- CONTINUED --



C^2 PROCESSES

(CONCLUDED)

- DIRECTING TACTICAL SYNCHRONIZATION OF COMBAT SUPPORT, COMBAT SERVICE SUPPORT, AND CROSS-SERVICES FUNCTIONS WITH MANEUVER
- CONDUCTING CORPS COLLATERAL OPERATIONS
- HARMONIZING MAJOR CAMPAIGNS
- CONDUCTING JOINT COLLATERAL OPERATIONS

THE ULTIMATE PRODUCT OF C^2

THE ULTIMATE PRODUCT OF C^2 IS A HIGH DEGREE OF EXPLOITATION OF THE POTENTIAL INHERENT IN THE JOINT FORCE. IT MOBILIZES OR ACTIVATES EVERY LEVEL OF COMBAT POWER PRESENT ON THE BATTLEFIELD

- INCREMENTS OF COMBAT POWER:
  - LEVEL I: COMPANY AND BATTALION FIRE AND MANEUVER
  - LEVEL II: COMBAT SUPPORT AND COMBAT SERVICE SUPPORT
  - LEVEL III: CROSS-SERVICE SUPPORT CAS, BAI, EW, AIRLIFT
  - LEVEL IV: BRIGADE AND DIVISION SYNCHRONIZATION WITH MANEUVER OF COMBAT SUPPORT, COMBAT SERVICE SUPPORT, AND CROSS-SERVICE SUPPORT
  - LEVEL V: CORPS CONDUCTED COLLATERAL OPERATIONS IN SUPPORT OF MANEUVER
  - LEVEL VI: JOINT COMMAND LEVEL CONDUCTS MAJOR CAMPAIGNS SUPPORTED BY JOINT COLLATERAL OPERATIONS
  - LEVEL VII: COMBINED FORCE (HEADQUARTERS) HARMONIZES NATIONAL CAMPAIGNS WITH CROSS-REINFORCEMENT AS PERMITTED BY INTEROPERABILITY CONSTRAINTS

-- CONTINUED --

THE ULTIMATE PRODUCT OF C^2

(CONCLUDED)

- DEGREES OF EXPLOITATION, EFFECTIVENESS, AND INTENSITY:
  - TO THE EXTENT THAT THE BLUE FORCE CAN SEIZE AND HOLD THE INITIATIVE, THE MORE INCREMENTS OF POWER IT CAN GENERATE AND APPLY TO THE ENEMY
  - TO THE EXTENT THAT RED SEIZES THE INITIATIVE, THE BLUE FORCE WILL BE FORCED DOWN TOWARD LEVEL I AND A SHARPLY LOWER DEGREE OF EXPLOITATION AND EFFECTIVENESS



MACRO ISSUES

- DIFFERENT SCENARIOS
- IMPORTANCE OF WARNING
- FORCE GENERATION:
  - MOBILIZATION
  - DEPLOYMENT
  - SUPPORT AND SUSTAINMENT

WE MUST PAY MORE ATTENTION TO THE FORCES INVOLVED, HOW WE PUT THEM IN THEATER AND HOW WE PROVIDE THEM SUPPORT -- THE GENERAL'S JOB

-- CONTINUED --

MACRO ISSUES

(CONCLUDED)

- INFORMATION DOMINANCE OR SUPERIORITY IS NOW A CRITICAL ISSUE:
  - TECHNICALLY IT MAY BE MORE DYNAMIC
  - TACTICALLY IT CAN BE KEY
  - OPERATIONALLY IT CAN BE DECISIVE

- TREND EVIDENT FOR TEN YEARS:
  - PRECISION MUNITIONS
  - TARGET ACQUISITION
  - DEEP FIRES
  - INTELLIGENCE AND SURVEILLANCE

WE MUST UNDERSTAND THE DYNAMICS AND OUTCOMES OF THE INFORMATION WAR

MICRO ISSUES

- WHAT IS IMPORTANT IN SMALL UNIT COMBAT?
  - S.L.A. MARSHALL
  - RONNIE SHEPHARD
  - DAVID ROWLAND
  - NATIONAL TRAINING CENTER

(• DESERT STORM LESSONS LEARNED)

TASK PERFORMANCE AND BEHAVIOR -- INCREMENTS OF COMBAT POWER

- ROLE OF INITIAL CONDITIONS
- ROLE OF LEADERSHIP AND SUPERVISION
- LEVELS OF PARTICIPATION
- LEVELS OF SOLDIER/SYSTEM CONTRIBUTION

PARADIGM

- THE KEY FACTOR IS NOT HOW WELL A WEAPON IS OPERATED, IT IS THE CONDITIONS UNDER WHICH THE WEAPON IS BROUGHT TO BEAR

- FOCUS SHOULD CHANGE:
  - CONCEPT OF OPERATIONS
  - PLANNING AND PREPARATION
  - EXECUTION

- ATTRIBUTES DESIRED:
  - SOUND
  - SYNCHRONIZED
  - SMOOTH

- PROCESSES TO STUDY:
  - PACE OF BATTLE
  - FOG OF WAR



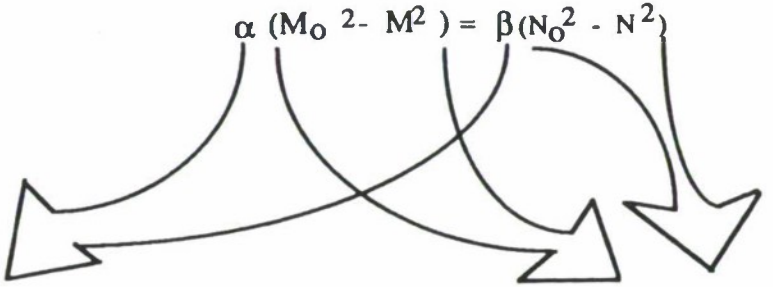
ALL PHASES MUST BE EXAMINED

- PLANNING
- PREPARATION
- EXECUTION

MUST CONSIDER THE FOG OF WAR

PRIORITIZING RESEARCH
(WITH APOLOGIES TO FREDERICK LANCHESTER)

$$\alpha (M_o{}^2 - M^2) = \beta (N_o{}^2 - N^2)$$



- EFFECTIVENESS IN DELIVERING LETHALITY
- EFFECTIVENESS IN REDUCING VULNERABILITY
  - ARMOR
  - INFANTRY
  - CAVALRY
  - ARTILLERY
  - AIR DEFENSE
  - ENGINEERS
- PRIME TASKS
- SOME SECONDARY TASKS
- BEHAVIOR

- EFFECTIVENESS IN POSITIONING FORCES
- EFFECTIVENESS IN CONSTITUTING FORCES
  - REARM
  - REFUEL
  - RESUPPLY
  - RECOVER
  - REPAIR
  - MAINTAIN
  - TRANSPORT
  - PERSONNEL
- PRIME TASKS
- SECONDARY TASKS
- BEHAVIOR



IMPLICATIONS FOR VARIABLE RESOLUTION MODELING

- BASIS SHOULD BE COMMAND AND CONTROL
- DRIVERS SHOULD BE MISSIONS
- ELEMENTS REPRESENTED:
  - CONCEPT OF OPERATIONS
  - TASK ORGANIZAITON
  - COMMAND RELATIONSHIPS
  - TASK ASSIGNMENTS
  - SYNCHRONIZATION AND HARMONIZATION
- PROCESSES REPRESENTED:
  - PLANNING
  - PREPARATION
  - EXECUTION

FOOD FOR THOUGHT

-- FALKLANDS, 1982 --

- BELGRANO
- SHEFFIELD (COVENTRY, ARDENT, ANTELOPE)
- SAN CARLOS
- GOOSE GREEN
- ATLANTIC CONVEYOR
- BLUFF COVE
- PORT STANLEY

HIERARCHY OF MODELS
FOR
CONVENTIONAL AIR POWER ANALYSIS

MR R. H. REISS
REGIONAL FORCES DIVISION
AIR FORCE STUDIES AND ANALYSES AGENCY

TITLE2.DRW

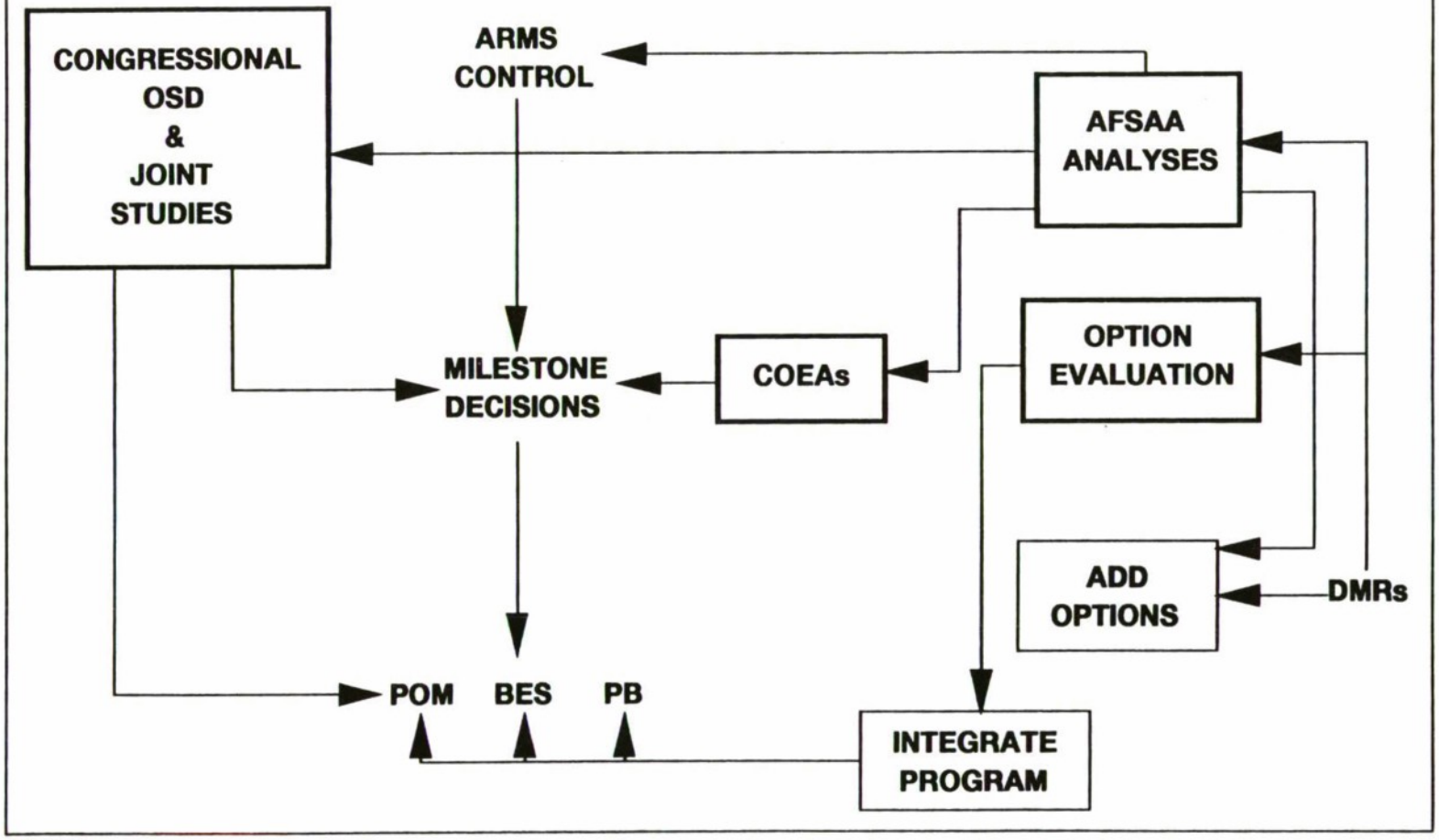
OUTLINE

- ANALYTIC REQUIREMENTS
- MODEL HIERARCHY
- ANALYSIS OF AIR FORCE BUDGET OPTIONS
- LESSONS LEARNED

AFSAA/SAG
4 MAY 1992
OPTS.DRW

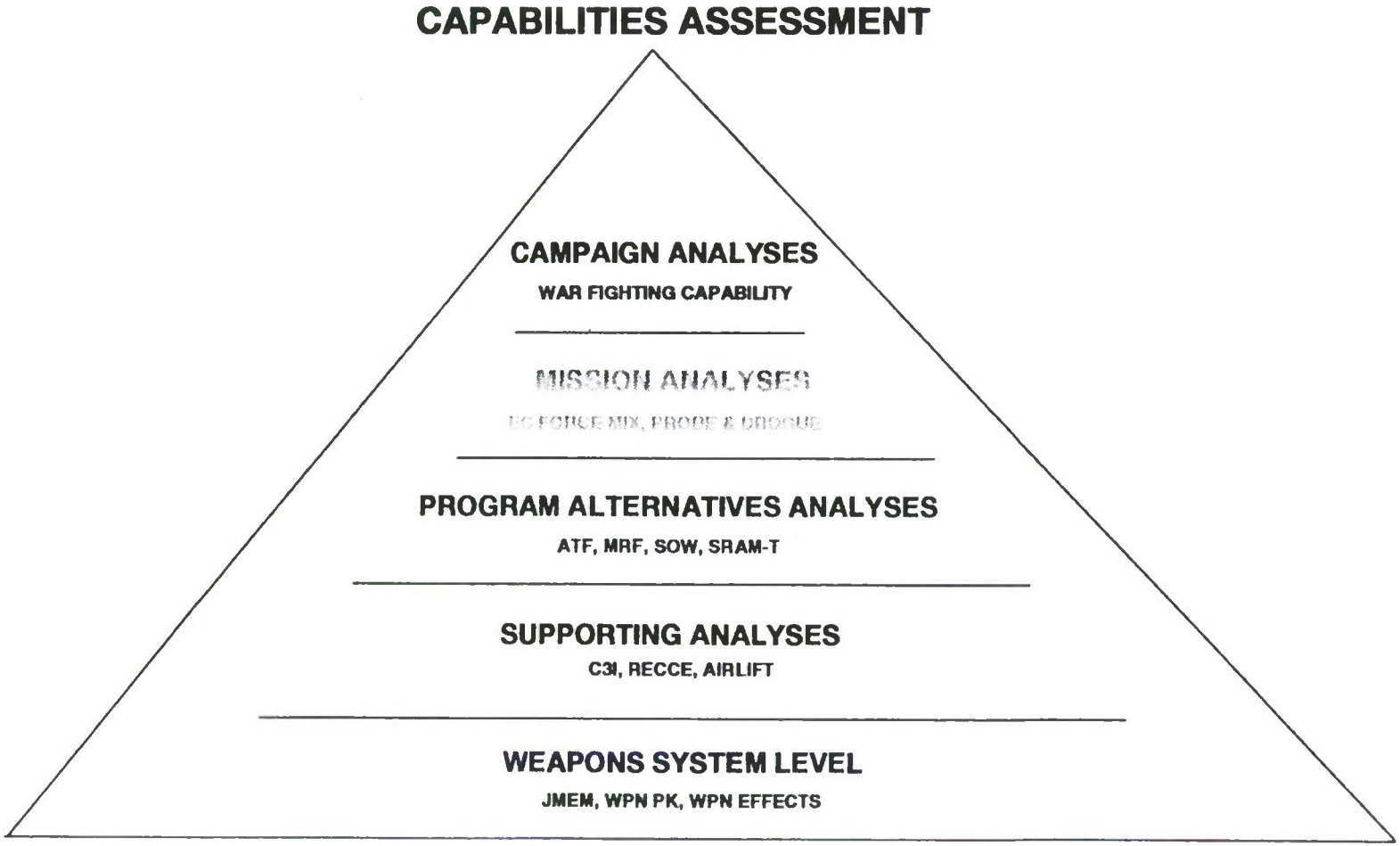
AFSAA ROLE IN RESOURCE ALLOCATION

RESOURCE ALLOCATION...ANY RESOURCES DECISION THAT EFFECTS THE TRAINING, ORGANIZING, AND EQUIPPING OF THE USAF.

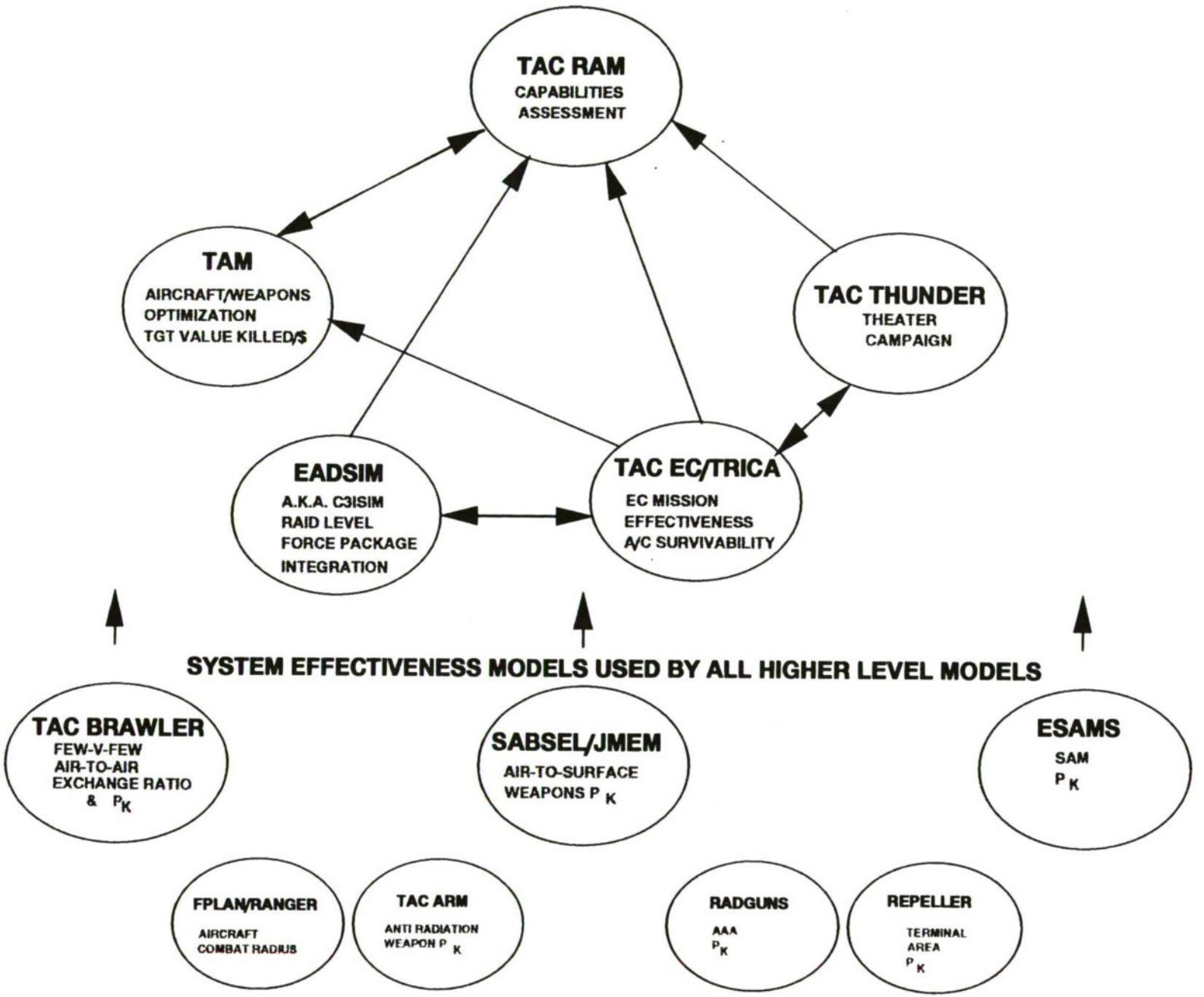


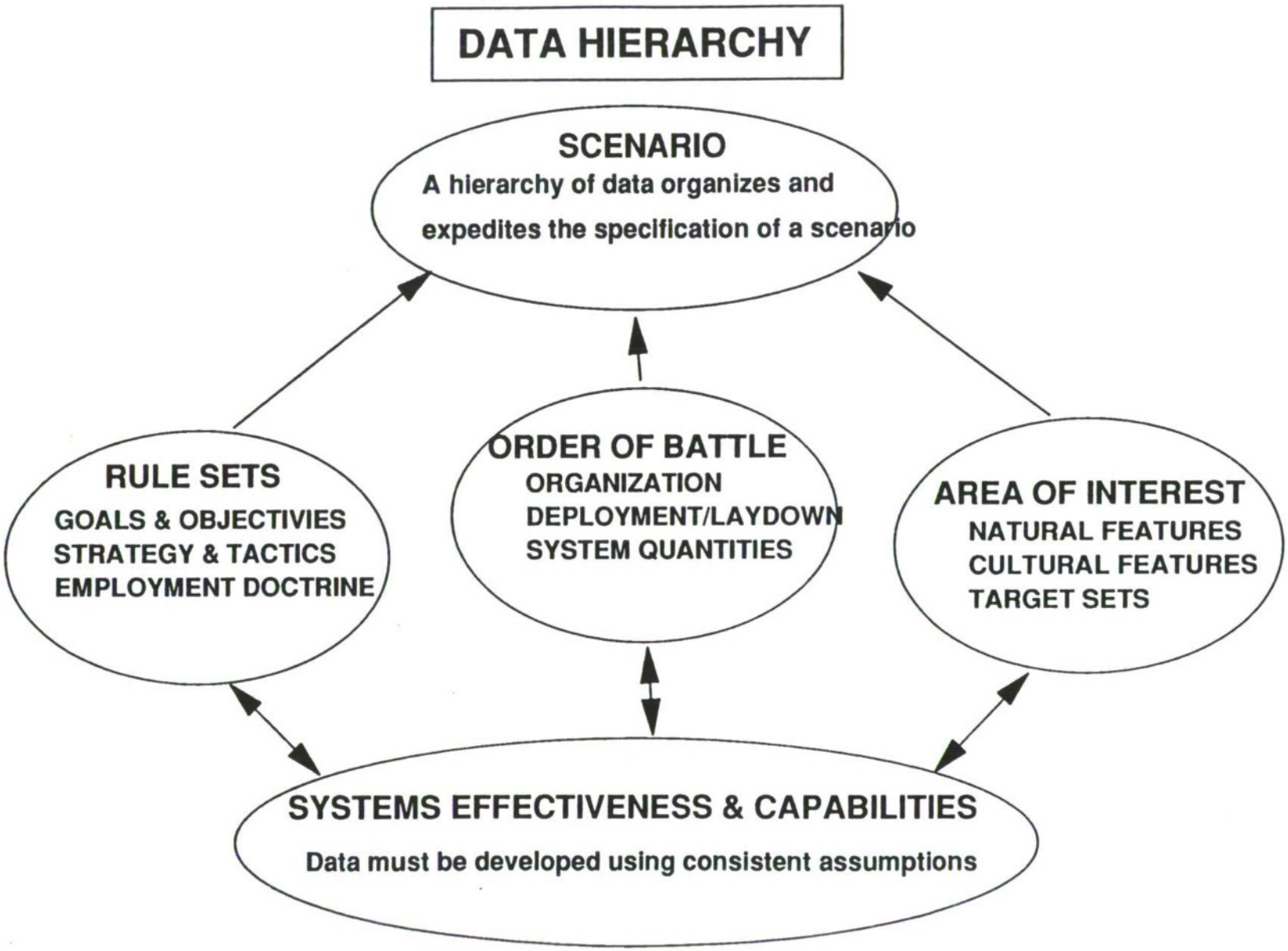
AFSAA/SAG
17 MARCH 1992
ARRA.DRW

ANALYTIC REQUIREMENTS

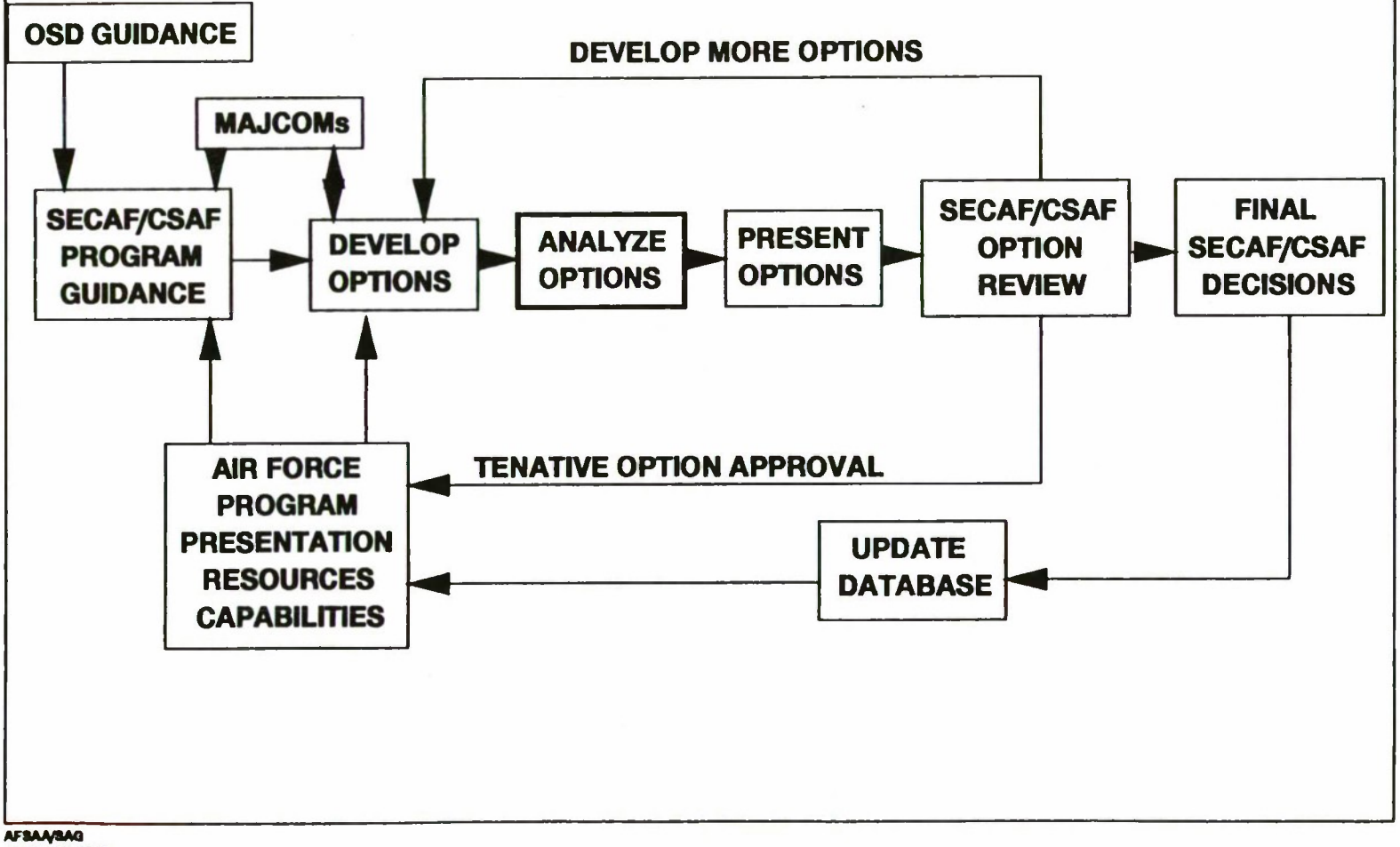


SAG MODEL HIERARCHY



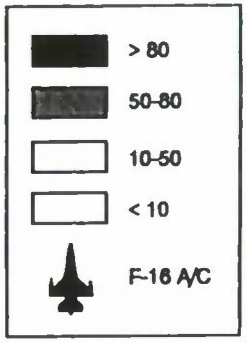

DATA HIERARCHY
SCENARIO
A hierarchy of data organizes and
expedites the specification of a scenario
RULE SETS
GOALS & OBJECTIVIES
STRATEGY & TACTICS
EMPLOYMENT DOCTRINE
ORDER OF BATTLE
ORGANIZATION
DEPLOYMENT/LAYDOWN
SYSTEM QUANTITIES
AREA OF INTEREST
NATURAL FEATURES
CULTURAL FEATURES
TARGET SETS
SYSTEMS EFFECTIVENESS & CAPABILITIES
Data must be developed using consistent assumptions

AFSAA ROLE IN RESOURCE ALLOCATION PROCESS (POM DEVELOPMENT)

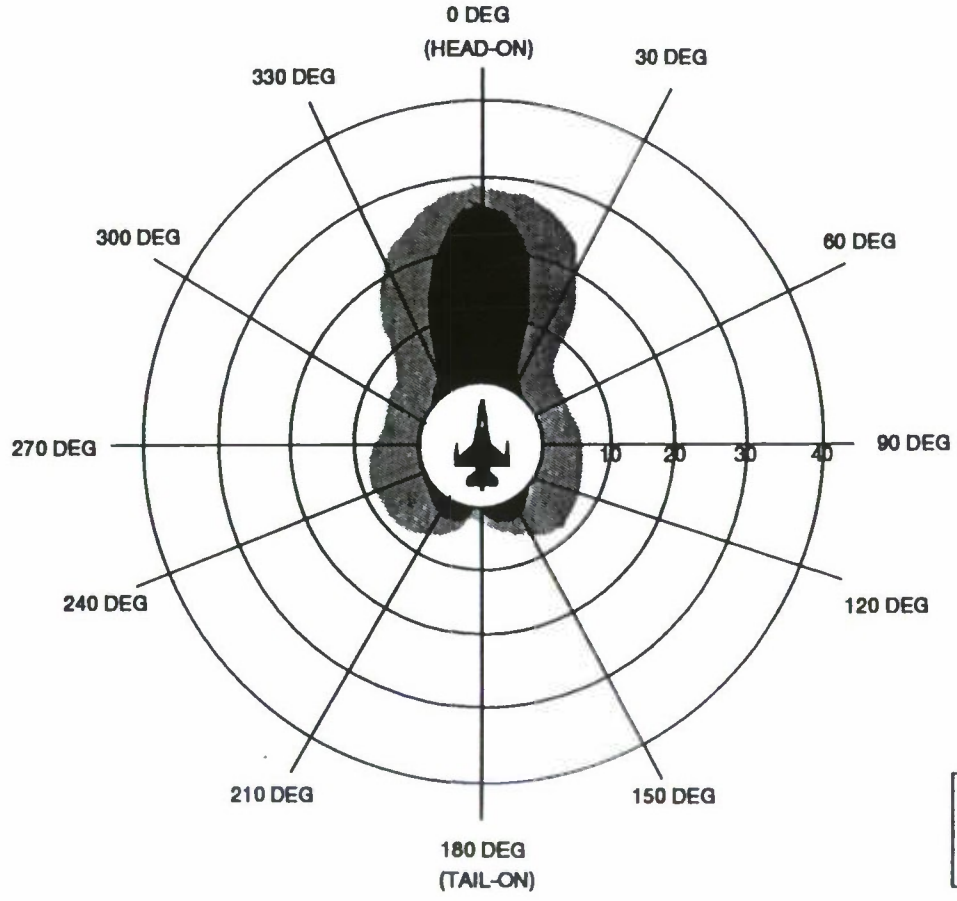


AFSAA/SAQ
17 MARCH 1992
ARRAP.DRW

P_K ENVELOPE (NOTIONAL)



SA-10 VS F-16 STRAIGHT AND LEVEL



$P_{K\ AVE} = 0.63$

AFSAA/SAG
17 JANUARY 1992
PK1.DRW

INVESTMENT SENSITIVITY OT BUDGET LEVEL

- SOUTH WEST ASIA - 1st
- NORTH EAST ASIA - 1st
- NORTH EAST ASIA - 2nd

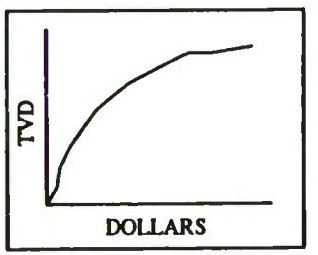
TIME LINE

C DAY
W DAY
D DAY

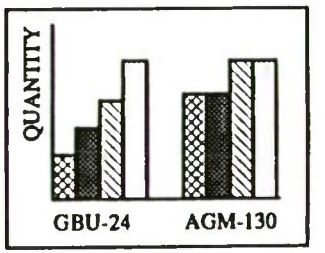
TPFDD

CLOSURE DATES
READY TO FIGHT

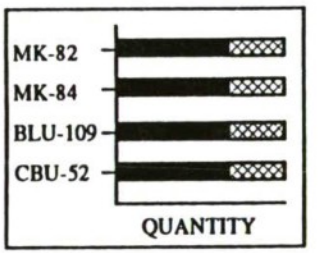
TVD vs BUDGET



INVESTMENT



STOCKPILE USAGE



Source: SAGF
OVIEW2

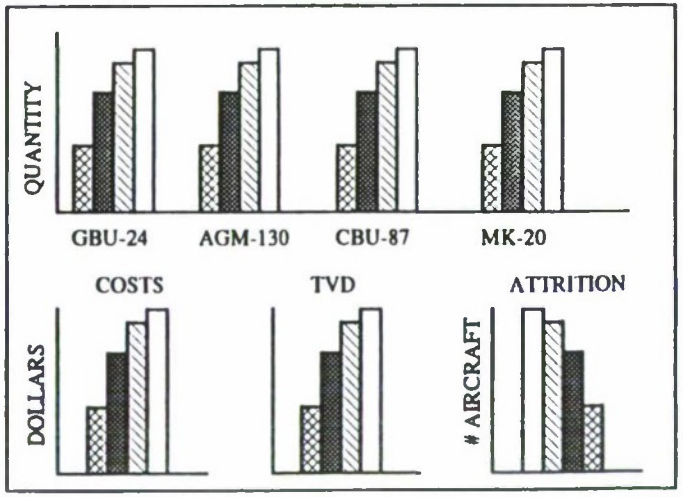
INVESTMENT SENSITIVITY TO SCENARIO

- SWA 1st/NEA 1st COMBINED
  - -- 0 DOLLARS
  - -- 3.0 B DOLLARS
- SWA 1st/NEA 2nd COMBINED
  - -- 0 DOLLARS
  - -- 4.0 B DOLLARS

STOCKPILE USAGE



PROCUREMENT OPTIONS



Source: SAGF
OVIEW3

OBSERVATIONS & LESSONS LEARNED

HIERARCHICAL APPROACH CAN SUPPORT SHORT SUSPENSE/ QUICK TURN ANALYSES

- MOST OF SCENARIO AND SYSTEMS EFECTIVENESS DATA MUST BE DEVELOPED IN ADVANCE
- NEED CAPABILITY TO GENERATE ALL DATA REQUIRED BY THE HIGHER LEVEL MODELS
- ALLOWS ANALYTICAL RESOURCES TO FOUCS ON THE ISSUE INSTEAD OF DATA COLLECTION

SYSTEMS EFFECTIVENESS DATA BASE

- DATA ON SPECIFIC SYSTEMS HAS A LONG SHELF LIFE
- ALL DATA MUST BE CREATED USING CONSISTENT ASSUMPTIONS
- POPULATING THE DATA BASE IS A CONTINUOUS PROCESS

COMBINING DETERMINISTIC AND STOCHASTIC ELEMENTS IN VARIABLE RESOLUTION MODELS

Dr. Patrick D. Allen
RAND
1700 Main Street
Santa Monica, CA 90407-2138
Electronic Mail: Patrick_Allen@rand.org

Abstract

This paper describes two rule-of-thumb techniques for combining stochastic and deterministic elements in a single variable-resolution model. The first rule is to explicitly define the situation being assessed, and the transition criteria from one situation to the next. The second is to define variables that are linearly divisible over time so that the stochastic and deterministic representations can be readily compared.

Background

By their nature, variable-resolution models tend to span many command echelons. For example, an aggregate representation of forces may be used in a theater-level combat model, while a representation of specific aircraft flying missions in that theater involves much more detail. In theory, a variable-resolution model should be able to span the representation of both the aggregate force interactions and the more detailed aircraft-on-aircraft interactions.

However, the preferred representation at each echelon may be significantly different. For example, the aggregate theater-level representation may prefer a deterministic combat assessment process, while the more detailed aircraft-level representation may prefer a stochastic combat assessment process. The deterministic and stochastic representations are significantly different in their approach and their processes--so much so that it is often difficult to describe a useful comparison between the two. For example, the sequential attrition processes assessed against penetrating aircraft from air-to-air and ground-to-air assets may make the loss rates in the stochastic process significantly different than the loss rates in the deterministic process.

This paper will briefly describe a few of the problems encountered when trying to combine stochastic and deterministic elements in a single variable resolution model, and present two rule-of-thumb design techniques for building both deterministic and stochastic representations of the same event in a single variable-resolution model. The example we will use is from the Theater-level Combat or Nonlinear Combat (TLC/NLC) model under development at RAND.

Explicitly Define the Situations

The first rule-of-thumb is to make sure that all of the situations to be assessed are explicitly defined. (This rule may sound simplistic, but it is amazing how frequently it is ignored in model design.) Explicitly defining the situations for assessment makes the comparison between different versions of the model relatively easy. For example, let us assume that the defined engagement types are not a function of the number of aircraft of each side involved in the engagement. The algorithms used to assess the outcomes of these engagements may not be sufficiently robust to account for the wide range of results that could be expected in engagements with a broad range of numbers of engaged aircraft.

For example, a stochastic representation of a many-on-many (aircraft) engagement may be readily compared to a deterministic representation of the same event, and obtain comparable results. Since the central limit theorem may apply in the many-on-many case, the average result may be closely related to the mean or mode of the stochastic distribution. If, however, one attempts to compare the results of a few-on-many or few-on-few engagement using the average or mean value, the results may not be comparable. Since the central limit theorem is not likely to apply when few assets are involved on one side, then one must more carefully define the situation to handle the factors that cause the central limit theorem to not apply. A different algorithm must be defined for many-on-many and few-on-few engagement assessment processes.

Similarly, if the situation changes during the assessment process, the criteria used to determine the transition from one type of situation to another type of situation must be explicitly defined as well. For example, the number of ground-to-air engagements against penetrating aircraft tends to increase as the number of penetrating aircraft increases. However, if the number of penetrating aircraft is so high that the defending ground-to-air assets have already maximized the number of engagements they can prosecute, then the model's determination of the number of ground-to-air engagements must be capped by this maximum number, and not increase as a function of the number of penetrating aircraft. The transition from a non-saturated defense to a saturated defense changes the situation such that the number of engagements no longer increases with the number of penetrating aircraft. In effect, the algorithm applied to determining the number of engagements has changes from a proportional to a fixed rate.

In a detailed stochastic representation, the maximum firing rate for each ground-to-air asset may be explicitly defined. In the more aggregate representation, a maximum firing rate for the group of ground-to-air assets may be defined. If either the deterministic or the stochastic representation preclude the representation of situation where the defenses could be saturated, then their results would not be comparable.

Therefore, the first rule-of-thumb in designing a variable-resolution model that may be either deterministic or stochastic is to clearly define the situations being assessed, ensuring that distinctly different cases are assessed differently, but in a manner comparable in both the deterministic and stochastic representations. This latter requirement for comparability is the focus of the second rule-of-thumb.

Define Variables that Are Linearly Divisible Over Time

The second rule-of-thumb is to define variables that are linearly divisible over time. If the variables defined in the model are not linearly divisible over time, then the comparison between deterministic and stochastic processes is very difficult to accomplish. The reason is that stochastic variables carry with them an implicit representation of time. A probability is the chance of a specific event occurring *within a specific period of time*. If the time frame is changed, the probability that the event occurs does not change linearly.

To illustrate why this is an important feature, let me briefly describe an example from the TLC air combat model. In TLC, there are ground-to-air and air-to-air engagement zones that penetrating aircraft fly through on their way to the target (see Figure 1).

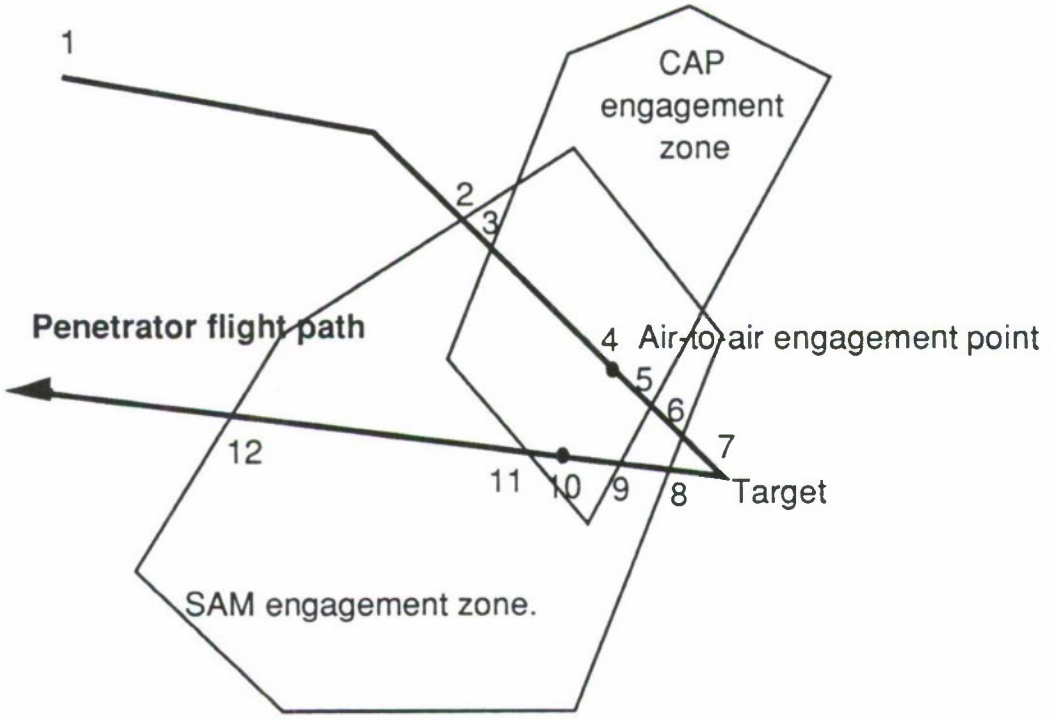


Figure 1: Sample Sequence of Events in TLC Air Combat Model

In this example, the penetrating aircraft fly from their bases (point 1) and enter the SAM engagement zone (point 2). Shortly thereafter, they enter a CAP or air-to-air engagement zone (point 3). No assessment occurs until the aircraft reach the air-to-air engagement point (point 4). At this point, any ground-to-air engagements that would have been assessed between point 3 and point 4 are now determined, followed by air-to-air engagements against against any surviving penetrators. Assuming the penetrators are not destroyed and do not abort their mission, they exit the air-to-air engagement zone (point 5) and then exit the ground-to-air engagement zone (point 6). At point 6, and ground-to-air engagements not yet assessed since point 4 are now assessed. Note that had there been no CAP engagement zone, all of the ground-to-air assessments would have been calculated to occur at point 6, rather than some at point 4 and some at point 6. The surviving penetrators then reach the target (which may have its own terminal defenses) and then egress, repeating a similar set of engagements depending upon the amount of time in each engagement zone.

Now let us compare a deterministic and stochastic representation of the preceding sequence of events. For example, let us define a variable that is the loss rate for penetrating aircraft crossing this ground-to-air engagement zone. Let us assume that for 100 penetrating aircraft, the loss rate is 15 percent. In both the deterministic and stochastic representation, 100 penetrating aircraft will lose 15 aircraft if there is only a SAM engagement zone and no CAP engagement zone. So far, so good.

If we add a CAP engagement zone, however, the comparison of the two methods begins to break down. For example, let us assume that the air-to-air engagement is assessed two-thirds of the way across the ground-to-air engagement zone. In the deterministic representation, 10 penetrating aircraft are lost before the air-to-air engagement is assessed, and 5 more after the air-to-air engagement. (For the sake of comparison, assume that no penetrating aircraft were lost in the air-to-air engagement. The air-to-air engagement in this case just serves as the dividing line in the ground-to-air assessment process.)

In the stochastic version, the determination of how many aircraft are lost before and after the air-to-air engagement is not so easy. For example, let us assume that of the 15 percent loss rate, 10 percent is applied to the penetrating aircraft before the air-to-air engagement, and 5 percent is applied after. The resulting calculation is as follows: before the air-to-air engagement, there are 100 penetrating aircraft. A 10 percent loss rate means that 10 aircraft are shot down, and 90 penetrating aircraft remain. No penetrating aircraft are lost in the air-to-air engagement assessed at this point. The 5 percent loss rate is then applied to the surviving (90) penetrating aircraft, resulting in 4.5 aircraft shot down. As a result, the stochastic process assesses 14.5 aircraft lost, while the deterministic process assesses 15 penetrating aircraft lost. The stochastic assessment process cannot be linearly divided over time and still produce the same results as the deterministic process. The reason is that stochastic variables

include an implicit rate that does not vary linearly over time. The key to designing a variable-resolution model with both stochastic and deterministic representations of the same events is to define model variables that are linearly divisible over time.

For example, one could design a model variable called "the number of engagements of penetrating aircraft by ground-to-air defenses". In this case, let us assume that in the preceding example, there were 30 missile firings against the 100 aircraft resulting in 15 penetrating aircraft shot down. The probability that a given missile engagement destroyed an aircraft is 50 percent. To assess the 30 missile engagements, assume 20 (two-thirds) occurred before the air-to-air engagement, and 10 occurred after. The first 20 missile engagements result in 10 aircraft destroyed (0.5 kills per missile firing). There are 90 aircraft remaining. After the air-to-air engagement has been assessed (with no additional losses to the penetrators), the remaining 10 missile engagements are assessed. Another 5 aircraft are shot down due to the 50 percent kill rate per engagement. As a result, the same number of penetrating aircraft are destroyed whether or not an air-to-air engagement was assessed.

The variable in the preceding example was defined to be linearly divisible over time. As a result, the stochastic and deterministic representation of the same event will lead to the same (average) result. For example, if one were to randomly determine whether each of the 30 ground-to-air engagements hit on a 50 percent probability, the average number of aircraft lost over many repetitions would be the same expected value in the stochastic version as in the deterministic version.

In the two preceding examples, if the air-to-air attrition rate was so severe as to cause 50 of the remaining 90 aircraft to be destroyed, then the second ground-to-air engagement may be distinctly different. If the penetrators do not abort the mission, then the number of ground-to-air engagements may no longer be 10 more missile engagements. That is why it is important to explicitly define what constitutes a specific type of assessment process, which is the first rule-of-thumb described above. Similarly, if the ground-based air defenses were already swamped, then the number of engagements may not change because the maximum number of engagements allowed has already been reached.

Conclusions

The two suggested rule-of-thumb techniques are to explicitly define the situations being assessed (and the transition process from one situation to another), and to define model variables that are linearly divisible over time. These two rule-of-thumb techniques allow both the deterministic and stochastic representation of the same events to be included in the same model and still produce comparable results. Designing the model so that the outputs of different versions are comparable allows the *causes* behind the outputs to also be readily comparable. The use of the same key variables

as the basis for both the low resolution and high resolution versions of the model will facilitate this comparison of results. Without the use of selected key variables that mean the same thing in each model, there would be little opportunity to calibrate results that could be readily traced as to the cause in each model. For example, if both models have a variable called the "probability of engagement of penetrators by interceptors," then the cause of attrition to the penetrators by the interceptors can be readily traced and compared in both models. Exactly how that probability is defined may be a function of a large number of other variables in the high resolution model, and may be defined by fiat in the low resolution model, but the basic design structure lends itself to easy comparison for purposes of cause tracing.

Author: Günther Scheckeler, IABG/WSP
Date: April 30, 1992
Purpose: Variable Resolution Model Symposium, Washington USA, May 5-7

A Hierarchy of Models at Different Resolution Levels

Abstract

Since the beginning of the seventies IABG, mainly by its war gaming centre, is developing and using simulation models and wargames for research purposes as well as for training and exercise.
The resolution problem was solved by a hierarchical approach.
Presently model families are available, which cover the area from battalion level to theatre level. The model families at each level may be used as research or training tools for its own, or as a means to generate a database to be evaluated and aggregated for the next higher model in the hierarchy.
These aggregation processes are automated, using traditional statistic methods, as well as new ones like the evolution theory.
During the last effort for instance, some 500 battalion simulations generated the database for brigade to corps level, and some 50 brigade size simulations were used to build the database at theatre level.

In addition to this aggregation process, a message based concept was developed to connect the different submodels within a model family. This allows 'plug compatible' submodels at different resolution levels within the family.
One example at corps level is logistics, which by using the same interface is represented by two models with different resolutions, which may execute simultaneously, each supporting one of the opposing forces.

The realization of these approaches is facilitated by a simulation software package, developed at IABG, which was used to program and implement all models.

1. Experiences

1.1 A History of Efforts at IABG's Wargaming Department

Setting up a model architecture, it is appropriate to look back at experiences, successes and failures that were acquired in the past. This review draws attention on the fact, that many basic principles for such a model architecture had been developed early. Some of them could be realised, many of them are re-invented regularly. Often it is necessary to convince changing military personnel and contractors of the benefits during exhausting discussions.

The necessity for integrated models arose very early. The objective of the project 'Tactical System Study II' (1963-64) was to develop military requirements for the equipment of the German air force with manned systems. A first model was developed to represent the integrated air war as good as possible at that time. In the following years, this model was used in extended and adapted versions for a series of studies by the German and US air force.

The project 'Optimal Composition of the Air Force' required to represent the whole integrated air war with all mutual effects of a potential enemy. The above mentioned analytical model was not adequate for this task. It was decided to use simulation methods as a flexible basis for future developments.

The first larger interoperating model family was developed and used to support the preparation of the "Bundeswehrplanübung 1970" (Federal Armed Forces Map Exercise).
The experiences from that effort are still valid, but it was difficult at that time to realise the new methods and principles.

The most important experiences were:
A hierarchy of methods or the so called iterative application of models is a prerequisite to accomplish complex studies. Different methods complement each other, each contributing its strength were the others are weaker.

The following years saw the development of interactive simulation models, mainly representing the land war, supporting greater study projects like MBFR or "Combat Troops 80". Following the mentioned experiences, closed simulation models were developed in addition to the interactive ones, where possible.

The problem of getting adequate data, especially for models at higher

echelons, came up very soon in the course of the development of the land war models. The WEI/WUV method, as well as own experiments with combat force numbers derived from educated guesses, were only partly sufficient and acceptable. For that reason the principle of a model hierarchy was developed. In a bottom-up approach, according to the hierarchy level of the forces, data were generated at lower levels and aggregated for use at higher levels.

In simultaneously this principle provided the opportunity to investigate sensitive scenarios in a top-down approach to determine areas of interest to be looked at in more detail by lower level models.

While it was a necessity to use the bottom-up approach to generate data for models at higher levels, it was only since the mid-eighties that the mentioned top-down assessment of problems was used as a basis for the study planning in the German MOD.

More work in the field of the model hierarchy was done since the mid seventies in a study called "war gaming conception". In the following of this study the simulation frame software BASIN was developed, which provides a set of software tools and software support to implement models in a standardised and harmonised way.
Simultaneously the techniques like decision tables, modularity, portability and data storing and retrieval were introduced and could be better supported.
BASIN was adopted by a couple of departments at IABG and still is the tool with which many of the simulation models are implemented. BASIN in its FORTRAN version is completed, while ADA versions are available since 1990/1991 and still are extended.

1.2 Methodological Approach

Based on the early experiences conceptions were developed for the systematic development of submodels which could be put together to integrated models supporting war games. This conception was refined during the years and supported by improving software techniques. The approach was validated continuously by using the models in wargames for research purposes, for the education of military staff and for classical operations research studies.

Even under changing political conditions, we hold this approach for suitable to meet the new requirements. The statements made in the paper "Considerations on a structure of analyses and models for problems of the armed forces" (Niemeyer, 1987) are still valid:

"A concise description is a precondition to analyse and investigate a system like the defence system. This description must encompass the following categories:
- Task and objectives
- environment
- The inner structure or the composition of the system."

Could this description be accomplished with adequate methods, then it is possible to make objective oriented changes, improvements and even optimisation. The objective of the armed forces in Central Europe may be stated shortly:
- Maintain peace
- Crisis management and maintaining political freedom of action
- Integrated defence within NATO

The environment of a system defines its confinements and limits within which it must be seen and formed, i.e. the available defence area, a given infrastructure, limited personnel and budget usually are fixed restrictions.

More possibilities are seen however, considering the inner structure:
- The fast and in some areas even revolutionary development of technologies and science open up new ways
- The continuous renewal of the weapon systems and the elements of the defence system allow adaptation and some times optimisation under changing conditions.

The task to define this defence system in a analytical way is dependent on some uncertainties:
- To maintain freedom of planning, it is necessary to assess time periods relatively far in the future.
- The performance of a defence system is extremely dependent on the behaviour or the armed forces of the potential enemy.
- Many phenomena of the battle and the conflict are unknown or at least poorly understood.
- The system is highly complex

We regard the last topic in more detail. It leads to a first methodical approach. The complexity and diversity is shown by the historically developed hierarchical structure of the armed forces. We must distinguish several hierarchical and process levels, which are to be assessed in a different way. Units of measurement at lowest level i.e differ by magnitudes from those at highest level. Objects at higher levels are to be described in other categories than those on lower levels.

At higher levels major command units must be considered, while at lower levels we have homogeneous weapon system units. In principle it can be assumed, that the totality of objects at one level are the building elements of the next higher level.

On the other hand one must derive the tasks and objectives at one level from plans and objectives of the superordinate level.

These considerations lead to an approach, where the structure of the analytical work resembles the hierarchical structure of the armed forces. In an iterative process conditions and scenarios are developed by analyses at higher levels which define those at lower levels. On the other hand, results, models and methods at lower levels are the basis for studies at higher levels.

From the systems analysis point of view, this top-town-approach where the total system is divided in its subsystem definitely should be preferred, because the interrelations and objectives remain visible. In the succession the bottom-up-approach is used to verify assumptions and to refine the system to be planned in more detail.

A second important point considering a defence system, are the various interrelations of different functions with their mutual influences and synergetic effects. Many mutual effects are completely defined by the corresponding functions of the enemy.

The human brain, addressing problems in a linear way, usually can not cope with that variety of factors and effects without the help of tools.

Mathematical simulation models together with the classic method of wargames are a means to guarantee the assessment of the many mutually interrelated single processes during an investigation.

Comparing wargames and closed simulations, it is clearly seen, that the advantages of one method supplement the deficiencies of the other one. It can be concluded, that both methods, supplementing each other, lead to more knowledge. No method can fully substitute the other. Additionally wargames are a tool to systematically investigate in decision rules, which are needed for the decision logic of closed simulations. The closed simulation models, providing the capability of many variations in short time, allow the sensitivity analyses of many assumptions and parameters, which afterwards may be tested and verified in wargames.

Due to the high costs for development and maintenance of models and their

data bases the tendency is to refrain from separate closed simulation models. Instead command and control modules should be developed, which cover the command and control process for time frames as long as possible. In that way the advantages of wargames are ideally combined with those of closed simulations, if a uniform conception for development and use of those models can be maintained.

The alternating use of different models, often called iteration of models can also be extended to other model types like analytical ones or optimisation models.

The following experiences are important when developing the instruments for armed forces analyses:

- The close linkage of models with different basic structures (i.e. analytical models, optimisation models and simulation models) usually is not successful, because its not possible to create a common structural basis. At IABG this approach was successful within the area of simulation models (KORA, AGATHA). The basic structure was provided by the simulation frame software BASIN, which enabled the data flow between different models by a common time axis, commonly defined objects and standardised data interfaces.

- Linking models of various very different levels is not much promising. Here it is very difficult to define a common basis.

- The development of databases which are not model oriented, is not much promising, compared with the effort to maintain actuality and accuracy. This is valid for technical/tactical data as well for aggregated data at higher levels.

- The differentiation between closed simulation models and interactive models is more based on the type of application and not on the model type. In each type of use, the basis for the evaluation is the simulation with the same object and the same data base. They differ by the extend of modules for command and control used, which cover a longer or shorter time span. A model of that type can be situated in any place between closed and interactive models. How it is called depends more on the application or the own point of view.

- To develop acceptable and suitable command and control modules, a empirical data bases for command and eontrol rules, created by interactive simulations is a necessity. Interactive simulation are used to test and validate the C^2 modules by comparing them with decisions made by human players. In that way a growing database is developed, by which the time

horizon of command and control rules in simulation models can be extended. Simultaneously a knowledge data base for expert systems is created, as the knowledge database and the command and control module are essentially identical. Expert systems additionally use a facts database, a inference machine and a dialog system.

2 The Scheme of the Model Hierarchy

2.1 Categories of Models and System Levels

To develop an architecture for models, methods and studies, it is necessary to agree on criteria for structuring the various kinds of model building. The two categories chosen in this scheme are the type and technique of the models on the one hand and the represented system, i.e. the defence systems with its different system levels on the other hand. Other important categories are thinkable like the purpose or the kind of use of a model, the usability of a model and the degree of resolution. Some of these are implicitly covered by the model technique category.
The second category of the system levels has been proved as suitable in most cases.
Both categories viewed as lines and columns build a table, each element of which denotes a reasonable combination i.e. a model technique for a distinct system level. (Picture 2-1)

Picture 2-1: Scheme of Model Hierarchy

Methods / System Levels			Simulation						
			interactiv	closed					
		Free Games (1)	Model Games (2)	Detailed Simulations (3)	Aggregated Simulations (4)	Analytical Methods (5)	Expenditure Methods (6)	ADP Techniques (7)	Personnel Planning (8)
Long-Term Security System (years)	A								
Short-Term Security System (months)	B								
Multilateral Military System (weeks)	C								
National Military System Force Structures (days)	D								
Combined Weapons (hours)	E								
Weapon Systems, Functions (minutes)	F								

Industrieanlagen-Betriebsgesellschaft mbH, Einsteinstr. 20, D-8012 Ottobrunn, Germany, tel. 089-6088-3303

G. Scheckeler

The system levels are structured as follows:

A: Long-Term Security System: A security system with long term interactions of sociological, economical, ecological, political and military aspects in international correlations. The time dimension is measured by years.

B: Short-Term Security System: A security system with relatively short term interactions of political and military forces in international correlations for crises management. Also mobilisation and reconstitution are seen in this category. The time dimension is measured by months.

C: Multilateral Military System: Military system of forces in a coalition. The military part of a war theatre may be seen here. Time is measured in weeks.

D: National Military System Force Structures: National armed forces, i.e. army, air force and navy in typical sub scenarios of a greater conflict. A corps of the army or a ATAF of the air force are typical. Time is measured in days.

E: Combined Weapons: National units of a service within typical - may be - generic sub scenarios derived from D. A brigade is typical for the army. The time is in the magnitude of hours.

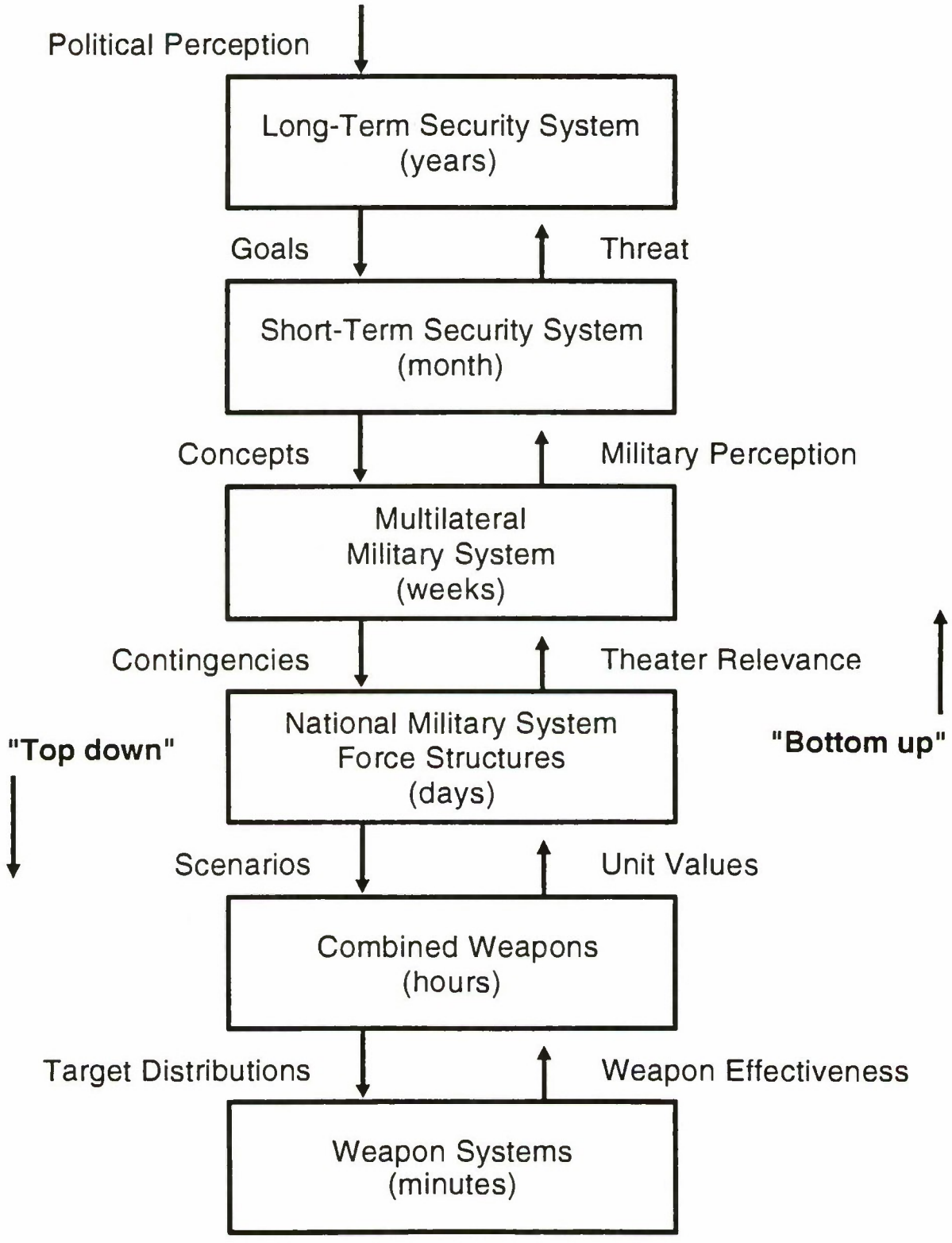
F: Weapon Systems and Functions: Weapon systems of a single service or single functions like reconnaissance, command and control at the lowest level. The time is measured in minutes.

In this structure, the following principles are of basic importance for model development and model application. (Picture 2-2):

- Using the top-down-approach, objectives and assumptions for an analysis as well as scenarios can be derived for lower levels (A --> F).

- Using the bottom-up-approach, data for a model can be derived by aggregating the results from the lower level. This can be done in away where the variety of the micro processes is sufficiently represented (F --> A)

Picture 2-2: Levels of the Security System



km124_ye.gem/30.04.92

The model types as the other category are structured as follows:

(1) Free forms of play within the systems analysis work. Examples are lowly structured dialectical debates, the brainstorming, the path-gaming method or games in the course of which conflicts, coalitions and even rules are worked out.

(2) Model games or wargames, which use computer based models or very rigid frameworks of rules. The computer based models usually are simulation models. This category often is called interactive or open simulations.

(3) Closed simulations which should represent many functions in a detailed and most realistic way. Human decisions are represented by rule driven systems or algorithms rather than by humans as in the open simulations.

(4) Closed simulations which represent the essential functions and their interactions in a relatively abstract way. They are sometimes called fast simulations

(5) Analytical methods from statistics or operations research. Examples are mean value methods, optimisation methods and cost-benefit methods. Usually analytic models are composed of static units.

(6) Expenditure models which predict costs, needed infrastructure or bulk supply. Often they use extrapolations on the basis of historic and empiric time series.

(7) Automated data processing techniques. Examples are data base management systems, netflow techniques and expert systems.

(8) Methods for personnel planning and deployment, but mainly methods for training and education.

The simulation methods (2) to (4) explicitly represent the dynamics of the related systems and are often called dynamic methods.

In analogy to the system levels we have a principle of basic importance for building an architecture of models.

- With growing degree of abstraction i.e. (2) --> (5), its easier and faster to use the models, but they need the results and knowledge from the more detailed models to represent the system level correctly. We also call it methodical aggregation.

- With growing degree of detail i.e. (5) --> (2) the models become more realistic but also slower and costlier. It is getting more difficult to investigate sufficient numbers of variations and alternatives. Therefore the more abstract models are used to limit the alternatives to the reasonable ones, the most interesting of which being examined with the more detailed models.

In that way the models complement themselves mutually. None is a complete substitution for an other one. Within a sufficiently complete model architecture, the basic data flow is clearly defined.

2.2 Examples of the Realised Hierarchy of Models

In this presentation we concentrate on simulation models. The interrelation between these models is shown in picture 2-3 and is called the flow of aggregation. This principle fully realised is an ideal situation which may never be fully reached. But it is a valuable guideline for the aggregation process and was realised in some parts as shown by the bold arrows. This principle is sometimes broken by requirements for details which belong to a lower level. The consequence usually is a unnecessary and more than proportional growing expenditure.

An aggregated model can be considered as a model of the detailed model. In the following the flow of aggregation is described in more detail.

Picture 2-3: Flow of Aggregation for Simulation Models (examples)

Methods / System Levels			Simulation						
			interactiv	closed					
		Free Games (1)	Model Games (2)	Detailed Simulations (3)	Aggregated Simulations (4)	Analytical Methods (5)	Expenditure Methods (6)	ADP Techniques (7)	Personnel Planning (8)
Long-Term Security System (years)	A								
Short-Term Security System (months)	B								
Multilateral Military System (weeks)	C		AGATHA C2	AFSIMO C3	C4 TRIAMOS				
National Military System Force Structures (days)	D		AGATHA AFRA AIDA TRAMP KORA GINA IKOMO D2	GINA IKOMO TRAMP D3 KOSIMO	TRIAMOS D4 KOLOG				
Combined Weapons (hours)	E		E2 KORA	E3	E4				
Weapon Systems, Functions (minutes)	F			F3 PABST	F4				

Industrieanlagen-Betriebsgesellschaft mbH, Einsteinstr. 20, D-8012 Ottobrunn, Germany, tel. 089-6088-3303

G. Scheckeler

(F3): Detailed closed simulation model representing several weapon categories i.e. direct fire weapon systems. The model PABST is an example, representing single vehicles, a very detailed terrain, line of sights etc. and is driven by a script.

(F4): The processes represented in PABST are also represented in the models of category (F4), but in a more aggregated way. During the transition from (F3) to (F4) adequate data and algorithms are worked out. Usually the data aggregation process is done with conventional methods like regression analysis. In the last years methods of the evolution theory were used to derive Lanchaster coefficients representing the loss curves of the direct fire weapon systems. This shows clearly that developing data and algorithms separately breaks the flow of aggregation.
The aggregated models of category (F4) are still closed models, in relation to the time frame at that system level. Mainly they are used to build the interactive model families at the next higher system level. At that level they need interactive command and control.
During the last effort several hundred battalion size scenarios had been simulated and the data base for the next higher level, i.e. KORA, had been produced using automated aggregation processes, as mentioned.

(E2): The model games at level E use the models of the lower level F, mainly aggregated models from (F4). All functions and elements are now introduced, which are needed for the more comprehensive scenarios at that higher system level. Simultaneously processes like reconnaissance, command and control, attachment of forces, movements without combat, result in time delays. Combat processes as they occur at the lower level, are mere events at level E.
The main task of model games is to introduce human command and control to combine all functions and processes of that level.
An example is the wargame model KORA, which consists mainly of aggregated models from (F4).

(E3): The transition from (E2) to (E3) requires the introduction of modules, which represent the command and control processes at that level. This allows faster applications of the model, because the time consuming human interactions are substituted. (E3) hence consists of clearly identifiable command and control modules linked with the model family of (E2). Presently we have no running systems at (E3).

(E4): The transition from (E3) to (E4) is similar to the aggregation from (F3) to (F4). Having no models yet at (E3), we use KORA from (E2) in what we call "laboratory games" to generate the data base for the aggregation process. Considering the model architecture, this clearly is feasible, but more time consuming than using models from (E3). The data ag-

gregation process again is supported using evolutionary methods.
The aggregated army model at (E4) in essence is AFRA.
During the last effort some 50 brigade size games with KORA produced the data base for the next higher level models, i.e. AFRA

(D2): An analogy to (E2) in this area of model games is built mainly with aggregated models of level E. The model family AGATHA consisting of AFRA, the air war model AIDA and the follow on forces model TRAMP, is located here. The model family KORA as also used at that level, mainly at the "lower boundary" of cell (D2).

(D3): Here again command and control modules are linked with the models of (D2). For the army AFRA together with the C^2 model KOGEN builds the model KOSIMO.
The models GINA (supply) and IKOMO (maintenance) represent logistics mainly for the army. It should be noted, that these models without their command and control part are also used within the model family KORA at (D2)

(D4): An aggregation in analogy to (E4) and (F4) is required at this level. The works are not so advanced yet, as at the lower levels. First results are TRIAMOS, a strategy driven aggregated simulation model, covering army, air force and navy.
Using GINA and IKOMO, the aggregated logistics model KOLOG was developed, which is "plug compatible" with these models within the KORA family. A standardised protocol was developed, which links either model. Usually, dependent on the purpose of the investigation, one side(i.e. BLUE) may play with the detailed models GINA and IKOMO, while the other side may use the aggregated model.

(C2): The model games at this level still use models from (D2), i.e. AGATHA.

(C3): At this level we momentarily use a slightly different approach. As no highly aggregated army models are presently available, a two level command and control module was developed to drive the AFRA model. The first level is KOGEN, the same module as in (D3), the next higher one AFGEN. The air force is represented with SSM (developed by IASFOR) and its C^2 module. Altogether they build the AFSIMO model.

It should be mentioned, that the development and application of models is not so distinct as shown in the picture. Especially on the borders of two system levels, models of both levels may be used.

Considering the command and control modules, there is a continuum of pos-

sible interactions with the model, as i.e. using KOGEN or AFGEN, generated orders can be monitored and changed by the human operators.

AN APPROACH TO HIERARCHIES OF MODELS: PROCESS INDEPENDENCE

Edward R. Harshberger
Bart E. Bennett
David R. Frelinger

June 1992

R A N D

DRAFT

INTRODUCTION

This paper, prepared for a RAND-sponsored conference on variable resolution and hierarchical modeling, discusses an approach to hierarchical modeling that focuses on maintaining "process" independence between different hierarchical levels. The observations and recommendations in this paper are motivated by our experience performing integrated analysis performed with several near-independent models at varying levels of fidelity. As such, it is a type of "make do" approach, and grows out of the sometimes competing needs to both develop new (and hopefully better) methodologies and to use and modify existing approaches.

RAND's Recent Air Vehicle Survivability Modeling Efforts

The concepts discussed in this paper are based on recent experiences in air vehicle survivability modeling. At RAND, two major analytic efforts undertaken over the last several years have had significant air vehicle survivability components. The first project, The Future of U.S. Strategic Aerodynamic Forces, examined the role, missions and force structure mix of the U.S. bomber and cruise missile forces over the next 30 years. The second project, Low Observables/Counter-Low Observables, examined the relative role and importance of stealth and other design features for air superiority aircraft.

Although these studies examined vastly different components of our nation's air forces, from heavy bombers to air superiority fighters, they had some common emphases. Both had as their purpose increased understanding of the broad, policy-oriented implications of technological advancements impacting high fidelity, phenomenology-oriented processes. Both projects examined a range of evolving air vehicles and evolving air defense threats and networks. Both performed high resolution assessments of system-level capabilities, then aggregated and translated this detailed understanding into meaningful statements on force-level issues. This type of analysis is an ambitious undertaking; the analytic issues span the range from engineering-level performance through concepts of operations to force allocation strategy decisions.

DRAFT

DRAFT

In addition, the two studies shared a methodological approach that relied heavily on computer modeling. A mix of models at several levels of fidelity and scope were used to inform various portions of the analysis. Some of the models used are shown in Table 1, along with brief descriptions. The precise purposes to which these models were put will be discussed in more detail later.

Critical to this discussion, however, is the fact that explicit attempts were made to tie the models together in an hierarchical manner. As noted above, we needed to explicitly tie high-level (mission, campaign) outcomes to high-fidelity (engagement, encounter, subsystem) attributes of the systems and forces in question. The form of the hierarchy was largely driven by the form of the available modeling methodologies - not, in general, a happy circumstance. The varying degrees of success encountered in these attempts and the observations and suggestions we can make in light of our relative successes and failures are the subject of this paper.

Table 1

SELECTED RAND AIR VEHICLE SURVIVABILITY MODELS

Model	Source	Primary Focus
STRAPEM	Community	Mission/Campaign Survival
Suppressor	Community	Mission Survival
TAME	RAND/Hughes	Air-to-air Mission Survival
RJARS	RAND	Surface-to-air Engagement & Mission Survival
Brawler	Community	Air-to-air Engagement & Encounter Survival
ESAMS	Community	Surface-to-air Encounter Survival
Numerous subsystem models	Community	Physical & Environmental Phenomenology

HIERARCHIES AND THEIR RELATION TO ANALYTIC MODELING

Before discussing specific modeling issues, we begin by describing just what we mean by hierarchies. The question is not as straightforward as it seems, and at least part of the difficulty is semantic. Table 2 outlines the terms and concepts used throughout the remainder of this paper and defined in this section.

Too often the term "model hierarchy" is very loosely used to refer to some kind of connectivity among various models. In our example, we have seen at

DRAFT

DRAFT

least three explicit ways our models can be hierarchically linked: by object groupings, process scope and fidelity, and analytic focus. These three categorizations of hierarchical modeling form the taxonomies shown as the headings in Table 2.

Table 2
SEVERAL VIEWS OF HIERARCHIES

Type of Model	Objects	Processes	Analytic Focus
Campaign	Multiple, disparate, aggregate players	Force allocation strategies	Campaign-level outcomes
Mission	Multiple, disparate, individual players	Many-on-many operational concepts	Mission-level effectiveness
Engagement	Multiple, similar individual players	Few-on-few cooperative tactics	Engagement-level effectiveness
Encounter	Single individual players	One-on-one tactics	System-level performance
Physical Effects	Subsystems & components	Phenomena and environmental interactions	Subsystem performance

These notional hierarchies linking both the analytic and modeling tasks are, in truth, the heart of the matter. The models are tools to formalize and organize our understanding of objects and processes, in an attempt to answer analytic questions. How, then, do our current models deal with object, process, and analytic hierarchies?

In Table 2, the object hierarchy taxonomy is the most generally familiar and exercised. When we speak of the objects and players in a simulation, we generally have a straightforward basis for understanding. These objects are represented by individual components, like "receivers" or "transmitters", by composite systems, like "planes" or "tanks", or by organizational units, like "squadrons" or "divisions". In addition, object hierarchies are the easiest to describe and program into a computer simulation. Modern computer languages such as C or MODSIM use data-structure techniques to facilitate the description of objects, their attributes, and their groupings. Combinations or aggregations of objects can be easily described and intuitively understood.

DRAFT

DRAFT

On the other hand, process hierarchies are much more difficult. A process hierarchy describes the relationship between the various functional representations among objects at a given level. Thus, they include, for example, environmental and physical processes among objects at the sensor and component level and operational concepts (including command and control) for the multiple, disparate individual players at the mission level. Perhaps the greatest problem with process hierarchies is describing and differentiating the distinct processes; where do the interactions between two objects change character enough to require a different process description?

A more fundamental problem occurs for modeling processes at various levels of aggregation within the hierarchy. It is a common practice to represent or approximate within a model the processes that are more properly modeled with higher fidelity at lower levels in the hierarchy. For example, instead of relying on analysis of the detailed processes performed at the sensor level, higher level models try to model this process in a simplified manner, perhaps using only some of the key variables used in more detailed calculation. In general, the models approximate more detailed algorithms at one or two levels below in the hierarchy using only a subset of the total output measures from yet lower levels.

Unfortunately, it is rarely possible to maintain consistent outcomes with representations of the same process at different levels of fidelity. If one acknowledges the need for the higher level of detail and fidelity (and we assume here that such a need exists,) too few variables exist to describe the process at the more aggregate level. If the attempt is made, the result is an under-specified model that is not only likely to come up with a different answer than the more detailed model but is also likely to contain a fundamentally different causal structure. In practice, an enormous amount of time and energy is spent on attempts to induce similar behavior between detailed models of processes and lower fidelity representations of the same processes. This effort is largely wasted, and in many instances more of the modeling effort is directed at trying to simplify the higher fidelity processes (and "validate" their credibility) than addressing those issues more germane to the specific level in the hierarchy.

We are not questioning here the utility of parallel approaches to modeling the same process as long as these efforts are undertaken at the same hierarchical level and level of fidelity. Neither are we suggesting that analytic products or theories from one process hierarchy cannot be used in the models at

DRAFT

DRAFT

higher or lower levels of the hierarchy. Indeed, science are replete with theories addressing distinct levels of aggregation in both natural and man-made processes. What we are questioning is the mixed, often blurred levels of process fidelity that are a part of most models. Additionally, we feel that as a community the focus of modeling enhancement efforts (particularly in the computer and information science fields) has been on object hierarchies while the difficulties and inconsistencies generated by inconsistent, overlapping process hierarchies have largely been ignored. Specific examples of this problem are discussed in the next section.

But where does this leave us for what is arguably the most important hierarchy, that of analytic focus? The "analytic focus hierarchy" is a formal way of asking "what questions and analytic issues must be addressed?" For air vehicle survivability analyses, the questions can range from "what is the detection range of this radar?", a subsystem performance question, to "how does the survivability of different aircraft in different missions affect day-to-day allocation of aircraft sorties?", a campaign-level allocation question. Modeling is only a part, although perhaps the most important quantitative part, of the analytic hierarchy of issues. We argue that a consistent analytic hierarchy requires *both* a consistent object hierarchy *and* a consistent process hierarchy. In any analysis, the point is *to understand both the process and the effects of that process upon the objects within it.* Inconsistency either in processes or objects yields confusion and severely limits the effectiveness and utility of modeling. Although of great importance in the definition, structure, and success of the modeling effort, we leave further discussion of the analytic hierarchy to focus on our recommendation to design modeling hierarchies with independent processes.

PROCESS-INDEPENDENT HIERARCHY

A potential solution to the problems noted above is to construct models according to what we term a "process-independent hierarchy." What we prescribe is a combined emphasis on both the object hierarchy and the process hierarchy in our models, with the form and divisions in the hierarchy driven by the analytic focus.- the questions being asked. In particular, we need to *avoid overlapping process modeling between hierarchical levels.*

DRAFT

DRAFT

An Example Model Hierarchy

To illustrate what we mean by this, we will use examples from RAND's recent studies. Figure 1 below shows a simple hierarchy of models with three potential levels: mission, engagement, and encounter. Arranged along the left of the figure is a list of functions, by no means complete, that are associated with calculations of air survivability at one or more of these levels. These functions are arranged roughly by operational sequencing order.

The bars on Figure 1 are meant to indicate the scope of the functional representations in each of the models. The mission-level model (in this case STRAPEM) integrates air and ground defenses, covering the entire range of issues associated with the survivability of aircraft when performing a mission within a stated operational concept.[1] The few-on-few or engagement-level models (RJARS and Brawler) also contain a subset of the processes found in the mission level model, beginning when aircraft enter engagements with assets that can actually shoot them down. At the lowest level, the one-on-one or encounter level models (ESAMS and Brawler) focus on one aircraft versus one defense element.

[1] We make no comment here at how well each of these models represents the specified process, only that it accounts for it in some way.

DRAFT

DRAFT

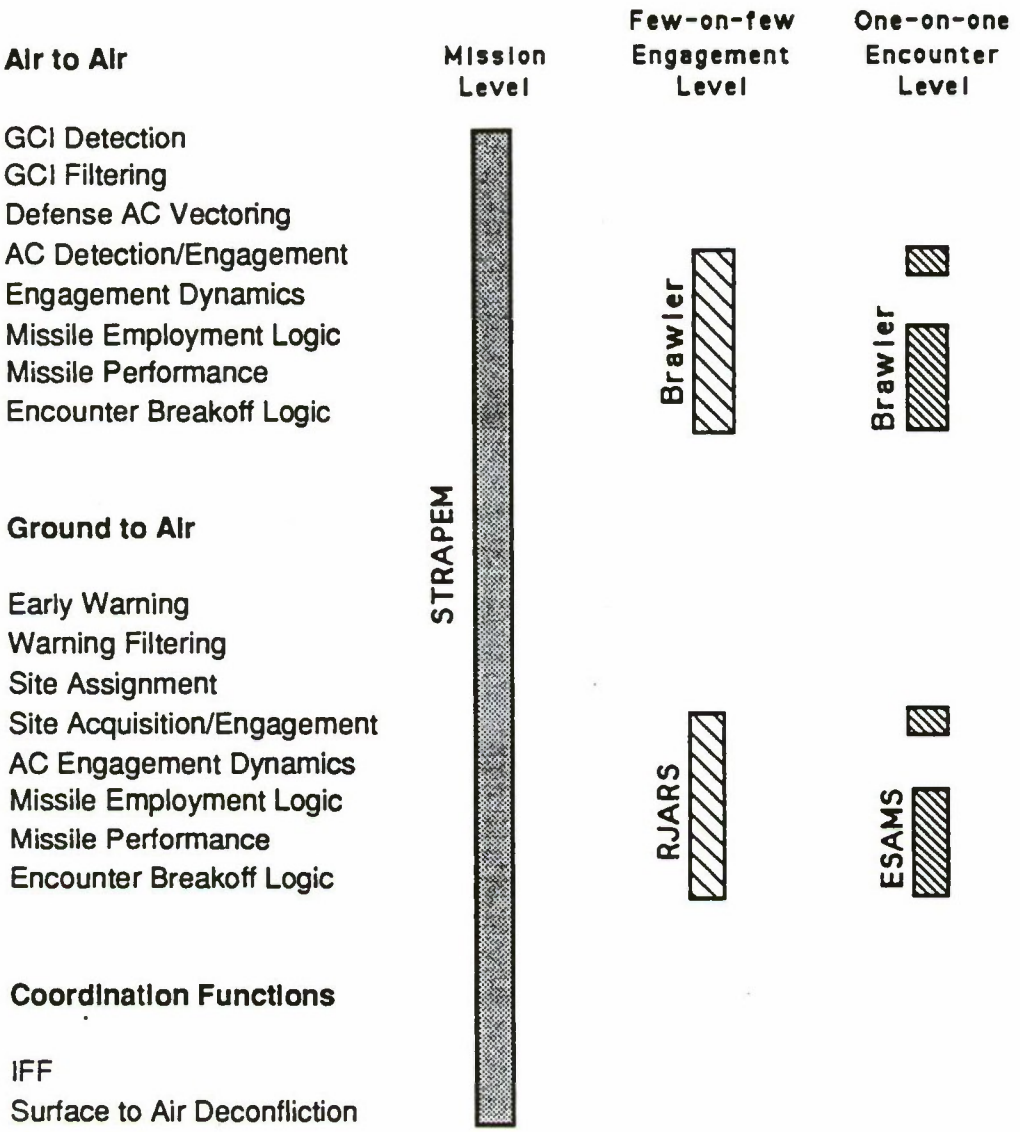


Fig. 1 — An air vehicle survivability hierarchy

As one moves to the right on Figure 1, down the hierarchy, the models have increasingly narrow scope. Note also that models do not always address only one level of the hierarchy. Brawler is a relatively high fidelity representation of air-to-air vehicle processes and operates at both the few-on-few engagement level and the one-on-one encounter level.

These three models and their hierarchy can be tied (albeit somewhat loosely) to a conceptual analytic hierarchy. One-on-one simulations are

DRAFT

DRAFT

generally aimed at investigating performance of specific systems in terms of defensive envelopes, detection footprints, etc. The external conditions for the analysis are generally artificial and fairly static (e.g. simple grids or straight-line flights).

At the engagement level, the analytic emphasis is on survivability of aircraft per engagement, as well as ancillary measures such as missiles expended, intercept distances, fuel used, etc. In contrast to the one-on-one level, dynamic tactics and reactive maneuvers are an important component of the engagement-level simulation, since the emphasis is on estimation of effectiveness of combined, like-systems rather than performance.

Finally, at the mission level, the analytic emphasis is on understanding the overall aircraft survival effectiveness when performing a given mission. The primary measures relate directly to mission effectiveness; for example, "successful bombs on target" is a primary measure of effectiveness when analyzing fighter aircraft in an escort mission, since their primary purpose is to ensure the survival of the bombing aircraft. Of course, the survivability of the escorts, the bombers, and the character and frequency of engagements would be of interest as well. At the mission level, one is actually analyzing the competing effectiveness of the defensive and offensive operational concepts. Hence, issues such as surveillance, assessment, command, and control become more important as macro determinants of the outcome than the more micro results of specific engagements.

Making Connections

The models shown above can be organized and discussed in terms of the conceptual analytic hierarchy, but how are these models actually connected together hierarchically? In a word, poorly, and the reason stems from the lack of process independence. In practice, the actual connection of these models into an "hierarchy" follows a widespread pattern: emulate processes one or more analytic levels below, and take inputs from two or more levels down in the hierarchy.[2] In the case of the Future Strategic Forces Study, the mission-level

[2] The term "emulate" is used here in a precise way, namely "to strive to equal or excel, especially through imitation." Typically, the attempt is made to simplify algorithmic complexity while maintaining predictive quality for the physical effect under examination. The obvious paradox is if this could be done with fidelity, the detailed representation would be uncesseary, and if it can't be done, why try?

DRAFT

model (STRAPEM) takes inputs from one-on-one models in the form of detection/performance footprints, while ignoring or attempting to represent the processes that are modeled at the few-on-few engagement level. This practice is shown graphically in Figure 2.

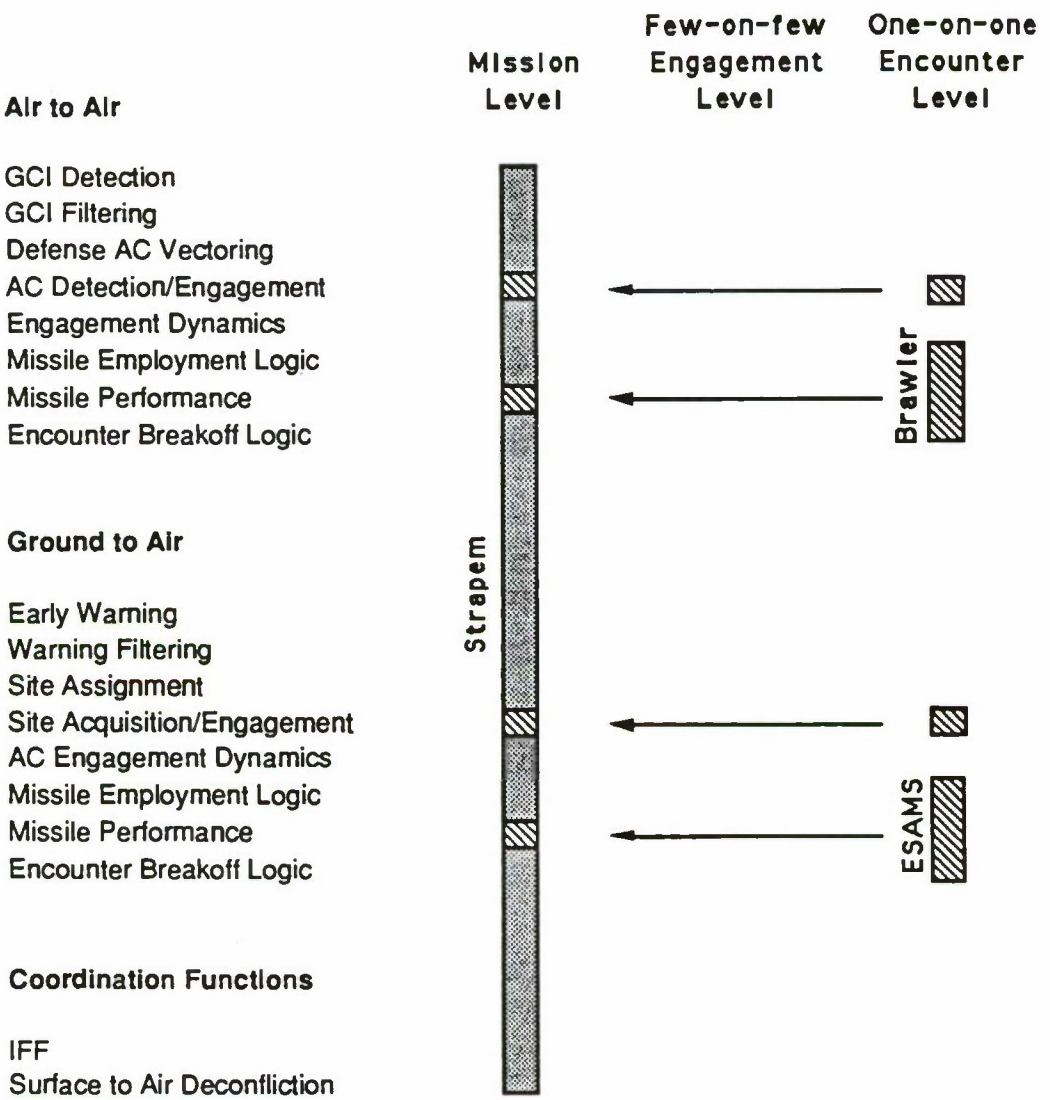


Fig. 2 — A model "hierarchy" in practice

There are more examples of this "represent one below, inputs from two below" approach, most notably in theater modeling that requires engagement

DRAFT

level results as inputs and ignores or inaccurately represents mission-level operational concept processes.

This approach causes a number of problems to occur. First, the higher-level models tend to do a poor job when representing the processes of models below them. In our example, STRAPEM contains none of the logic for reactive, cooperative tactics that drive the dynamics of air-to-air engagements and employed rudimentary missile employment tactics and break-off logic. This condition, poor representation, may be theoretically unavoidable;[3] it is certainly unavoidable in practical terms. Models are and should be designed to investigate different levels of the analytic hierarchy; as such they will inherently operate at different levels of fidelity. Models with a broader scope in the hierarchy and hence at a lower level of fidelity, simply lack the inputs to describe the details of lower-level processes.

This style of modeling actually produces a false sense of fidelity. By delving down into the higher fidelity levels and supplying some (but not all) of the variables needed to address them, model developers can "include" higher fidelity processes. This creates both a warm feeling for the developer (who, if he is like us, enjoys the details a great deal) and a more sellable model product, one that "includes" issues that users believe are important. The price paid for this "comfort" is, however, distressingly high. By overlapping processes in the hierarchy at different levels of fidelity, we lose connectivity between adjacent hierarchical levels. In the example we show, there is no means of consistently translating engagement-level results into the mission-level model, since the models do not share a common input-output structure. So, for instance, one has no means of discerning at all how a higher engagement exchange ratio (the ratio of defenders killed versus attackers killed in an engagement) translates or does not translate into higher mission effectiveness. The mission-level model has no concept of "exchange ratio" as an input.[4]

---

[3] Actually, one solution would be to subsume the more detailed model within the high-level model and call the detailed model as a function. This solution is not emulation as we have defined it, however, it is duplication. Although restrictive from a practical stand point, it is, in fact, a logical extreme of the solutions we propose.

[4] For those unfamiliar with exchange ratio as a measure, it is perhaps the most commonly used metric for comparing aircraft air-to-air combat effectiveness. It is therefore exceedingly unfortunate that it has no place in current mission-level models.

DRAFT

DRAFT

What is even more distressing from the analytic perspective is that any tractability or paper trail is lost when we step from one level of the hierarchy to the next. The model outputs simply do not flow between levels. This situation leads to a tendency towards "bait and switch" presentations. Solid, detailed analysis is shown at one level of fidelity, followed by solid, careful model results with a broader scope. The unspoken implication is that the broader analysis is at the same level of fidelity as the more detailed analysis, or at the very least, was directly informed by the results of the more detailed analysis in some consistent way. In fact, the two *can not* be reconciled in any consistent way and usually there has been little or no attempt to so. The authors have observed such presentations at all levels of the hierarchy discussed here - and may even have presented a few themselves.

Process Independence: A More Logically Consistent Approach

A more logically consistent approach to the problem is to maintain independence of processes between levels of fidelity - stop overlapping representation. In concrete terms, the model developers need to leave a "hole" in their model when they encounter a process that requires examination at the next lower level in the hierarchy. The shape of the "hole" is defined by the outputs of the next level down. In the case above, the mission-level model needs to stop and request inputs when it determines that an engagement has begun. The requested inputs should be the outputs of the engagement-level analysis: exchange ratios, attrition per engagement, fuel use, weapons expended, etc. These inputs may be generated quite simply (to assess the impact of various hypotheses) or in great detail directly from the model at the next lower level.

An example of this type of hierarchy is shown Figure 3. Note that the mission-level model requires inputs at the engagement level and does not overlap engagement-level processes. Likewise, engagement-level processes do not overlap encounter-level processes.[5] In such a structure, consistent and traceable connections can be made between levels in the hierarchy.

[5] Although as noted before such processes may be *duplicated* in the engagement level, a situation we discuss at some length later in this paper.

DRAFT

DRAFT

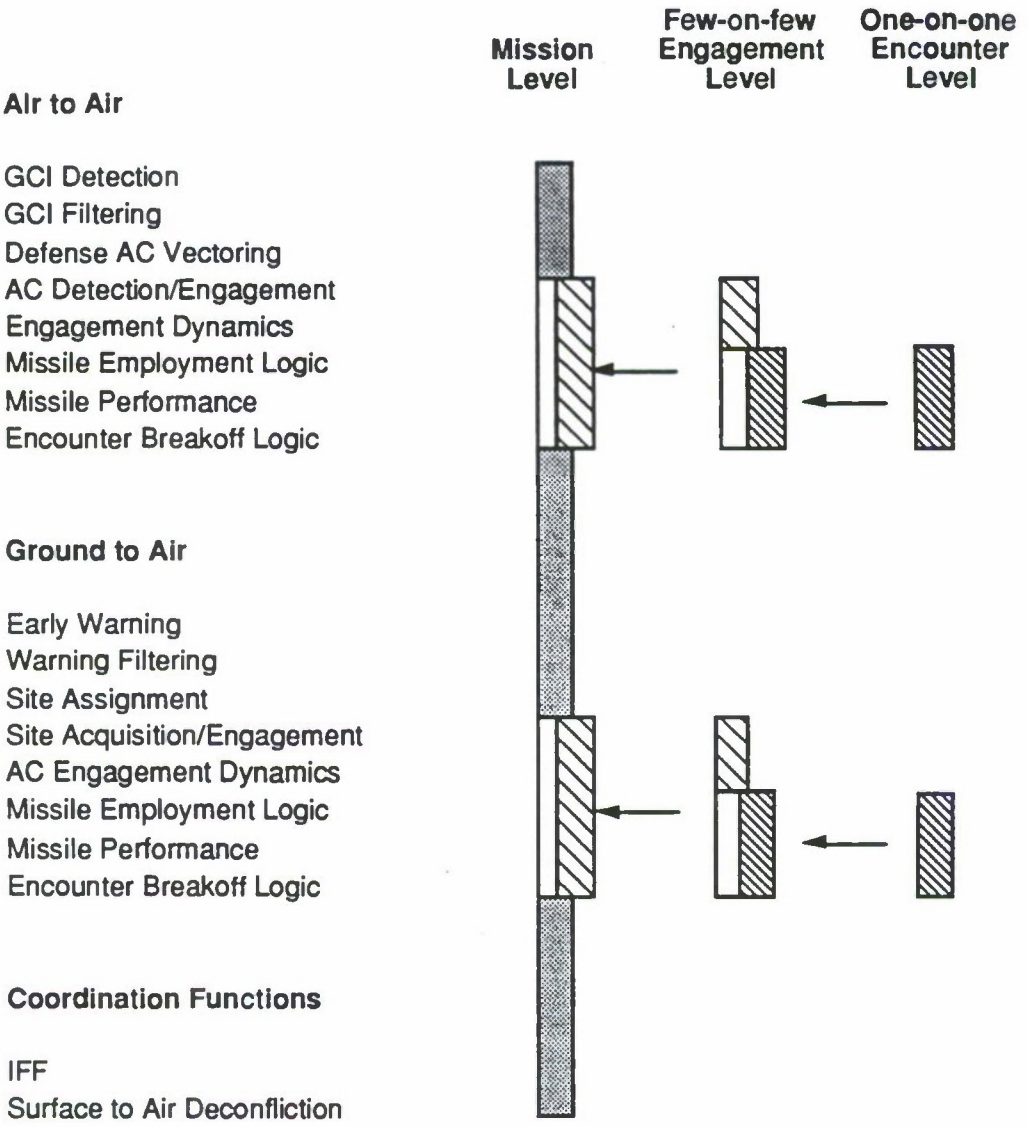


Fig. 3 — A process-independent hierarchy

Difficulties with Process Independence

Although we would argue that the process-independent hierarchy shown in Figure 3 is more logically consistent than hierarchies involving overlapping processes, such a hierarchy of models is truly difficult to operationalize. At least three things are required to make such a system work: situational awareness, adjudication, and open-mindedness.

DRAFT

DRAFT

Situational Awareness. "Situational awareness" means that a model must be designed to recognize when a process is beginning that is handled by a higher-fidelity approach. In addition, the model needs to recognize the operative starting conditions of the process. In the case of our mission-engagement model example, the mission-level model must recognize the fact that an engagement is beginning and correctly characterize its nature (e.g. 2 F-15Cs versus 4 Su-27s). *This is an extremely subtle modeling problem.* To the best of our knowledge, such internal model situational awareness (in terms of processes) has not been attempted on any serious scale in our field. It requires new perspectives and new coding approaches.

Adjudication. Adjudication describes the resolution of a process modeled at a higher fidelity. For example, in the mission model, the adjudication of an engagement includes such elements as the type and number of surviving aircraft, expenditure of resources (weapons, fuel, and countermeasures), and the end-of-engagement geometries. Adjudication is critical, since all of the information required by higher-level models is unlikely to be supplied from lower-level models. In real terms, too many variables are involved to fully parameterize the process. Therefore, once the higher-level model has identified a situation and the starting conditions, some form of defensible (not perfect) method for adjudicating the outcome is required. We have identified at least five ways of accomplishing such an adjudication: 1) duplication, 2) sparse table interpolation, 3) fitted curves, 4) bounded table lookup, or 5) invention.

Duplication is just what it appears to be: complete duplication of the lower-level model within the higher-level model. An example of this can be seen in Figure 3, where the same model, Brawler, is used for both the engagement-level and encounter-level modeling. A more ambitious example of this situation would be, for example, calling Brawler as a subroutine of the mission-level model. As most readers are aware, however, such a system of duplication swiftly breaks down as run-time and code complexity increase. It is nonetheless applicable in some situations.

Table interpolation and fitted curves are, in fact, functionally equivalent. Although it is difficult to believe that one could completely fill a parametric matrix and never exceed the table bounds, one can imagine a sparsely filled matrix with suitable interpolation and limited extrapolation techniques. Likewise, a set of model runs at the more detailed level can be approximated by series of curves or

DRAFT

simple multivariate regressions or probability distributions. In a perfect world, the functional representations used for table interpolation and simple regressions would be equivalent. In fact, they would probably differ is some respects. Both of these methods are less than perfect, as they sacrifice some information and can be highly skewed at the table boundaries. Nonetheless, these methods are far superior to the type of representation discussed earlier, since they are tied explicitly to the outputs of the more detailed models. The challenge is to develop robust and straightforward methods of developing such interpolation, fitting or distribution schemes.

Another approach, what we term bounded table lookup, is to develop more densely defined tables that are applicable over only a given range of initial conditions. For this approach it would be necessary for the interface between the upper-level model and the table to recognize when the boundary conditions are exceeded.

A final method of adjudication is, quite simply, invention. "Invention" might be a needlessly provocative term; one would hope that inputs with uncertainty attached would be parameterized. Nonetheless, a legitimate use of any model is to generate "what if" cases, without the benefit of more detailed modeling at lower levels. Such is the case for assessing the implications of hypothesized conditions.

Again, development of these approaches into workable procedures is a difficult analytic and computer science problem, requiring a significant amount of research and effort.

Open-mIndedness. Pursuit of process-independent hierarchies calls for a gestalt shift on the part of model developers and users, a serious departure from the way we currently do business. From the developer's perspective, it requires a leap of faith: someone, somewhere will fill in the "holes" of the model. For the user, a rigorous analytic design, rather than an ad hoc model mix, must be created. This requirement runs counter to the natural inclination of all analysts and modelers to create a completely self-contained (and apparently self-sufficient) piece of work.

Furthermore, we see a need to more closely join the modeling user community, which specifies the demand for models in terms of the analytic hierarchy, and the model developer community, which focuses on creating generality and flexibility in their object and process representations. Too often

DRAFT

users must take whatever is on the shelf and somehow make do, as if they were shopping in a State run market. Developers, on the other hand, struggle to ensure the applicability of their models to a broad enough audience and range of problems.

Finally, process-independence requires modelers and analysts to "just say no" to ever-increasing detail. As we have established, adding some detail in an attempt to represent high fidelity process modeling does nothing but create confusion, however much it appears to add "realism" to the simulation. Modelers typically wish to second-guess the user community and fill their model with flexibility ("sure, it can do that"). This in and of itself is not problematic, but it can be the driving factor towards adding unnecessary detail. The user community bears as much responsibility for this problem, if not more.

Our feeling is that there will always be plenty of areas to expand the breadth of process modeling consistent with the analytic hierarchical level of the model without increasing the level of detail into deeper levels of either the object or the process hierarchy. What this really means, however, is that modelers need to change their focus. The y should concentrate on correctly modeling difficult processes at the chosen level, not on attempting to represent detailed lower level models. For the example we have used, the emphasis at the mission level would be on such issues as surveillance and assessment, command and control, and operational concepts along with development of robust situation awareness and adjudication algorithms.

The Payoff

So, the approach we are advocating, process independence, presents a range of difficulties - practical, theoretical, and human. It requires trust and self-discipline on the part of modelers and analysts, and a new theoretical and structural focus in modeling.

But the payoff could be immense. The process-independent approach provides a logical structure that enables consistent and intuitive connections between hierarchical levels - analytic, process and object. It provides an opportunity for more structured, repeatable analysis and a clearer audit trail from one set of models to the next, a critical factor in developing and maintaining the credibility of modeling efforts in general.

DRAFT

DRAFT

Finally, and most importantly, the structure we have discussed here provides for a better match and connection between our modeling, our analysis, and the critical decision making we desire to influence. To the extent that we can clearly link the levels in our modeling hierarchy, both objects and processes, to the levels in our analytic hierarchy, we enhance the opportunity for clearer, more responsive, and more effective analysis.

DRAFT

RAND

A DISTRIBUTED NETWORK APPROACH TO VARIABLE RESOLUTION MODELING

Keith W. Brendley and Jed Marti
April 1992

ABSTRACT

This paper describes recent RAND work to selectively enhance the resolution of tactical level force-on-force models. Our basic approach is to link program modules over a Local Area Network (LAN) through a master program called the Seamless Model Interface (SEMINT). The resulting architecture enables permits disparate models of varying resolution to be invoked. In the pilot case, we linked the Cartographic and Geographic Information System (CAGIS) to a UNIX version of Janus-A in order to generate a higher resolution depiction of background terrain and target objects. We overlaid and correlated Defense Mapping Agency topographic data and Landsat satellite imagery data in CAGIS. Key scenario elements (objects passed from Janus-A) and relatively detailed target descriptions are used to develop detection probabilities that are passed from CAGIS to Janus-A through SEMINT. This system, which we have demonstrated, permits analysis of alternative future weapon systems to be evaluated through the application of models designed to represent key features of future systems at the required level of detail.

INTRODUCTION

Computer simulations of military operations range from out-the-window cockpit simulators to worldwide strategic wargames. Between these extremes, force-on-force simulations model brigade and division engagements at the weapons system level. These latter types of simulations, sometimes known as engagement models, are often used for estimating the battlefield effectiveness of new technologies.

In this paper, we examine the capability of current engagement models to represent advanced technologies with sufficient fidelity. As a case study, we examine the effectiveness of engagement models in correctly representing target acquisition. Target acquisition is a critical component of any battle since a target that can be acquired can be killed. Also, advanced sensor and signature reduction technologies significantly stress current target acquisition algorithms, creating increasing difficulties in predicting the effectiveness of future systems.

310-393-0411

1700 Main Street, PO Box 2138
Santa Monica, CA 90407-2138

CONTEMPORARY WARGAMES

Three engagement models that have been used by the U.S. Army for various analyses are SIMNET[1,2], JANUS[3], and CASTFOREM[4]. These models, particularly JANUS and CASTFOREM, provide input to the Army's Cost and Operational Effectiveness Assessment (COEA) studies that often determine whether or not a new technology is to be introduced in to the force. Engagement models therefore perform a critical role in the Army's technology development policy. As a minimum, they must represent the technology that they purport to measure to a sufficient degree to analyze its contribution to the battle.

SIMNET is a network of manned and automated simulators that fight in a virtual battlespace. At the simulation stations, operators acquire simulated targets as they appear on screen. Resolution and scene realism are limited, but operators can slew sensors, change fields of view, etc. SIMNET is normally used with only a small number manned simulator platforms. In most cases, autonomous agents within SIMNET, called Semi-Automated Forces (SAF), represents the actions of systems for which there are no manned simulators. SAF does not really model target acquisition. It uses two look-up tables that give detection as a function of range: one table for all ground-based sensors and one for all airborne sensors. Therefore, SAF is currently a poor system for determining differences between sensors since all systems use the same two tables.

JANUS and CASTFOREM use nearly identical target acquisition algorithms, developed by the Army's Night Vision and Electro-Optics Division (NVEOD)[5,6], for modeling imaging sensors . This model has maintained the same form for nearly two decades. Three great simplifications are assumed in the construction of the NVEOD model. First, the performance of sensors can be represented by a minimum resolvable temperature (MRT) curve. Second, the only target dimension of importance is its "critical" dimension, usually its height. Third, the background is a uniform field, allowing the target to background contrast to be defined by a single number. Essentially, these simplifications can be categorized as a reduction of the information that characterizes a real scene and sensor.

---

[1]R.E. Garvey, Jr. and P. Monday, "SIMNET (Interactive SIMulator NETworking," *Phalanx*, Mar. 89, pp. 25-29.

[2]W. Schneider, "SIMNET: A Breakthrough in Combat Simulator Technology?" *IDR*, Apr. 89, pp. 489-491.

[3]Janus reference goes here.

[4]M.C. Douglas, et al., *CASTFOREM: METHODOLOGIES*, TRAC-WSMR-TD-4-88, Mar. 90.

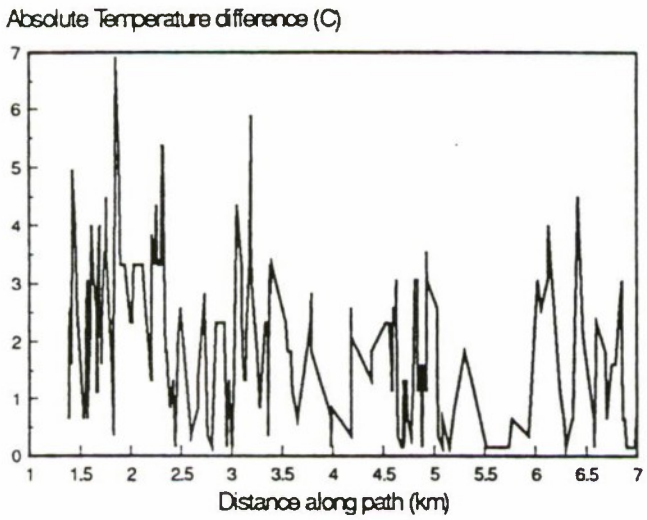
[5]J.Ratches, et al., "Night Vision Laboratory Static Performance Model for Thermal Viewing Systems," ECOM-7043, Apr. 1975.

[6]L.B. Scott and L. R. Condiff, "C2NVEO Advanced FLIR Systems Performance Model, VIS-89-001, Apr. 1990.

The NVEOD target acquisition model is used in most contemporary wargames, but it was developed under the assumption that little background and target data could reasonably be incorporated into force-on-force simulations. Important target characteristics such as radiance structure across the surface, hot spots, glint, movement, and firing signatures are either ignored or modeled poorly. Sensor technologies such as automatic target recognition (ATR), sensor fusion, and image enhancement receive similar treatment.

Most problems associated with target acquisition models can be traced to the paucity of data with which they work. Why bother to model signature phenomena when only a single contrast value is used? Why develop a methodology for simulating sensor fusion when it must eventually be distilled into an MRT curve? What is the use of a clutter model when one does not even know the local contrast of the vehicle to the background? The list goes on.

Since these models are actively employed by the military to make material acquisition and R&D decisions, this is not a mere academic exercise. For example, suppose that a company develops a new sensor fusion device, and the Army wishes to decide whether to pursue this avenue or simply try to make a current sensor better. At some point in the decision cycle, a wargame will almost certainly be invoked.



In a similar vein, suppose a camouflage pattern juxtaposed against an "average" background deceives observers quite well. Current wargames cannot determine if this pattern works well in conditions of a varying background, i.e. the real world. For example, Fig. 1 shows the absolute temperature difference of a vehicle moving against a typical background in Germany.[7] Any one temperature difference cannot sufficiently capture this considerable fluctuation.

[7]This figure was generated using CAGIS with Landsat imagery. An armored vehicle assumed to have constant temperature moves along a 7 km path of varying temperature.

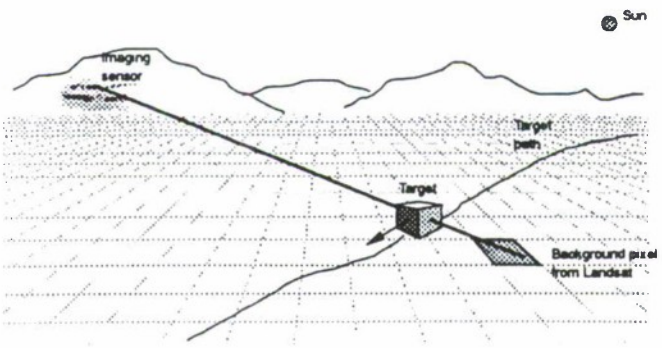
HIGHER RESOLUTION MODEL FOR TARGET ACQUISITION

The key, then, to making target acquisition representations within wargames more realistic is redesigning them to accommodate higher resolution data. For example, background data can be brought in from satellite imagery by a Geographic Information System (GIS). RAND developed the Cartographic and Geographic Information System (CAGIS) to support military operational research[8].

CAGIS can incorporate data from a wide variety of sources such as digital terrain-elevation data (DTED) from the Defense Mapping Agency (DMA), Landsat imagery, SPOT imagery, or imagery from any number of aerial platforms. These data may be and warped ("rubber sheeted") to correlate projections. They may then be used for any number of operations research applications.

Satellite imagery in conjunction with a GIS can be used to enrich the data available for target acquisition representations in a wargame. In Fig. 2, the general approach used at RAND is depicted. An imaging sensor on a tank searches over a scene built up from DMA and satellite imagery data. Since these data are too coarse to accurately represent local clutter in the immediate target surround, there is little to be gained by a detailed target description, and the target is modeled simply. It is composed of a small number of plates that can take on many different spectral characteristics. Important phenomena such as glint angles[9], hot spots, and varying thermal characteristics are modeled. The presence of a coarse background also allows the effects of confusing objects (scene congestion) to be modeled. These effects are important in calculating search times.

Although the above approach brings vastly more data to bear on the problem, it does not capture the effects of specific target shapes, detailed target mottling patterns, or near-field clutter. All of these phenomena are important for determining the probability of acquiring a target given that the observer looks in its direction. Phenomena such as these are normally modeled on



[8]A.L. Zobrist and L.J. Marcelino, *LHX Mission Analysis Using MOSF Sun Terrain Procedures: An Overview*, RAND, N-2760-A, 1989.

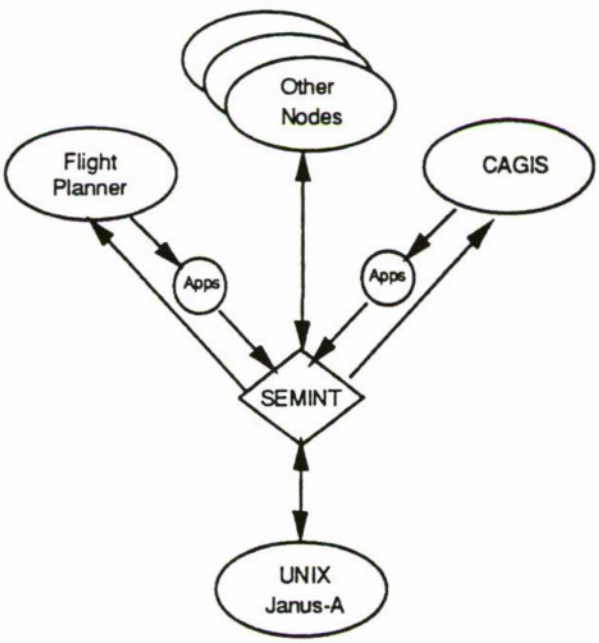
[9]Glint angles are associated with near-specular solar reflection.

photorealistic image display systems.[10] A CAD system is used to design the vehicle and build a wire frame model. Other models convert the wire frame diagram into a faceted image and degrade that image as a function of sensor and atmospheric characteristics.

There are two reasons that GIS and imaging systems currently have not previously been incorporated in wargames. First, these systems tend to be very large and are dependent upon specialized system architectures and programming languages. CAGIS for example consists of tens of thousands of code lines and relies upon an "executive" program which is larger still. It would therefore be prohibitively expensive to actually code these systems into a wargame. Second, wargames call target acquisition routines many thousands of times during a typical run. This has forced algorithms to be very fast, greatly limiting their complexity.

THE SEAMLESS MODEL INTERFACE FOR NETWORKING SIMULATION MODULES

RAND is experimenting with a fundamentally different approach to alleviating these problems. Rather than recoding a wargame, it is possible to link various programs through a local area network (LAN), with each program contributing what it does best to the overall simulation.



An example architecture is shown in Fig. 3. RAND is currently using this object-oriented networking approach to incorporate an advanced target acquisition module into an engagement level wargame called Janus-A[11]. Each network module communicates with other modules through a master program called the Seamless Model Interface (SEMINT). The objects of the simulation are tanks, helicopters, and other systems, each of which has its own attributes, which are updated from time to time in the simulation to reflect physical processes such as movement, detection, and weapon effects.

---

[10]P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*, Springer-Verlag, NY, 1990.

[11]We use the UNIX version of Janus-A, developed by RAND under contract to the Army. The UNIX environment is much more amenable to the networking approach taken here than, VMS, the system under which Janus previously operated.

SEMINT orchestrates portions of this updating to permit the introduction of higher resolution data than the original simulation was designed to use, and to assure that applications (Apps) run at the right time using the right data. We have successfully demonstrated this system with Janus-A, SEMINT, and a single CAGIS module accurately passing objects and target acquisition in the proper timing sequence.

This architecture is inherently amenable to multiplexing modules, allowing parallel processing. For example, if a single target acquisition node is too slow, a number of such nodes may be run simultaneously, each waiting to give up its data to SEMINT on command.

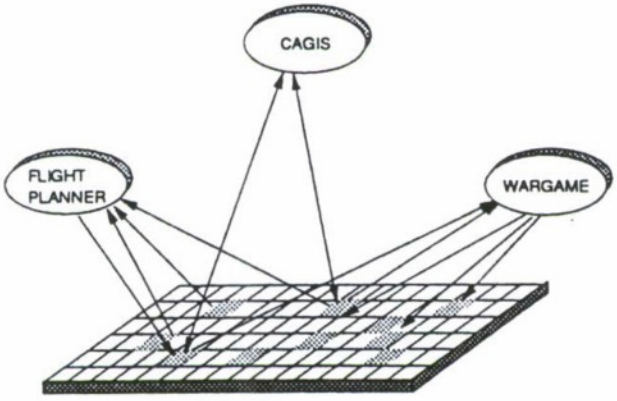


Fig. 4 depicts another, more complicated, simulation network involving a wargame (such as Janus-A), a target acquisition module (based on CAGIS), and a manned helicopter flight planner. The flight planner would be an example of another simulations module which could be added to the system in addition to the target acquisition module.

The SEMINT program maintains a database of the location and status of all significant objects. The locations are provided by the wargame, or in the case of the helicopter, by the flight planner. In turn, SEMINT provides the location of these objects to the other simulation components such as the target acquisition model. SEMINT provides the synchronization mechanism to maintain time consistency among the various components.

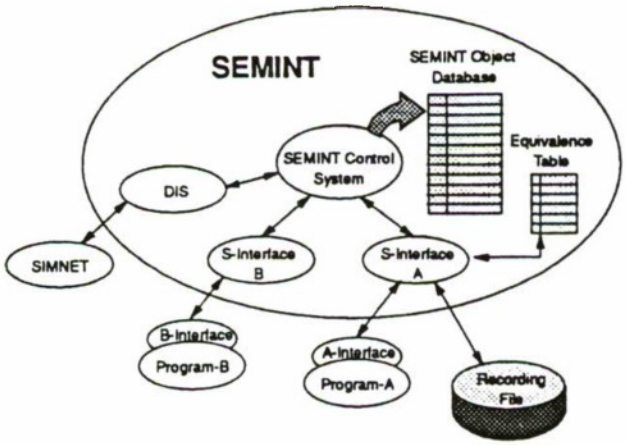
Unlike the Aggregate Level Simulation Protocol (ALSP)[12] system, SEMINT is based on a distributed version of the Virtual Time mechanism, Time Warp.[13,14] This allows various components of the simulation, such as the flight planner, to plan and execute somewhat in advance of the global state of the world as maintained by the SEMINT database. The level of aggregation is sufficient that the full Time Warp protocol need not be implemented on most of the processes.

---

[12]R. Weatherly, et al., "Aggregate Level Simulation Protocol," *Proceedings of the Summer Computer Simulation Conference*, Soc. Comp. Sim., Jul. 91, pp. 953-8

[13]D. Jefferson, "Virtual Time," *ACM Trans. on Prog. Lang. Sim.*, vol. 7(3), Jul. 85.

[14]B. Gates and J. Marti, "An Empirical Study of the Time Warp Mechanism," *Distributed Simulation* (ed. B. Unger and D. Jefferson), Simulation Councils, Inc., San Diego, vol. 18(3), 1988.



We designed SEMINT, shown in Fig. 5, to support analysis with some emphasis on actual man-in-the-loop interfaces. A major goal was to simplify the overall design and minimize requirements for sophisticated programming support. For example, rather than standardize on a large protocol such as DIS[15], each programmer provides an application specific interface. The SEMINT programmer provides appropriate interfaces to the SEMINT database and maintains the file of protocols.[16] This requires additional work building SEMINT, but very little effort applied to interfacing at the applications level.

Through the use of TCP/IP[17], we run SEMINT simulations on one or more processors. This parallelism allows us to optimize the processing of specific applications such as graphics, or target acquisition .

To facilitate analysis, the system can also record application interactions for future playback. This allows applications modules to be run "off line" in a recorded battle, creating a powerful tool for analyzing details of a specific application. For example, the signature technology of systems could be modeled and modified parametrically. The target acquisition module would then be used to calculate probabilities of detection within the given force structure, vehicle movements, and terrain. Once an interesting set of parameters was defined, the complete simulation could be invoked to determine the battlefield effectiveness of the technologies.

An advantage variable resolution modeling using a distributed network via SEMINT is the compartmentalization of models outside the wargame. This permits:

- Using different programming languages such as C, C++, FORTRAN, and LISP
- Employing application specific hardware such as SUN workstations, Silicon Graphics display hardware, and high speed floating point hardware
- Allowing modelers to work in a large scale environment without specific knowledge of other simulation modules

---

[15]*Military Standard: Protocol Data Units for Entity Information and Entity Interaction in a Distributed Interactive Simulation*, Inst. for Sim. and Training, Orlando, Oct. 1991.

[16]Of course, this does not preclude the use of DIS as one of the protocols.

[17]E.J. Feinler, et al., *DDN Protocol Handbook: Volume 1*, USDCA, 1985.

- Encouraging multiprocessing by low granularity and efficient protocols.

CONCLUSIONS

In summary, the networking approach saves time and money, while greatly increasing the flexibility of simulations. These attributes, particularly increased flexibility, will allow modelers to vary the resolution of their simulations in order to capture system attributes that are important to the study question at hand.

Submitted for publication to *Naval Research Logistics.*

CUTTING SOME TREES TO SEE THE FOREST: ON AGGREGATION AND DISAGGREGATION IN COMBAT MODELS[1]

RICHARD J. HILLESTAD AND MARIO L. JUNCOSA[2]

ABSTRACT

Most models of air and land combat use schemes of aggregation and disaggregation in representing combat systems, in spatial configuration, and in depicting progress of a battle. For example, the use of firepower "scores" is an extreme case of aggregation of weapons into a single measure. Combining like systems into weapons categories, partial aggregation, is a common approach to representing a large number of aircraft or ground weapons types. This paper explores different approaches to aggregation and what is known theoretically about aggregation and disaggregation in combat models of the Lanchester type commonly called "square law" in two dimensions. It defines requirements for consistency between aggregate and higher-dimensioned models of this type. Some important conclusions are that aggregation should take into account the specific capabilities of the opponent (raising concern about many "scored" approaches which attempt to evaluate force components in isolation), and that partial aggregation (grouping "like" systems) and disaggregation of previously aggregated results can only be done consistently when certain restrictions on the relative attrition capabilities of weapon systems hold. When this is the case, specific weightings for the force resources can be determined for the partial aggregations.

---

[1] This research was performed for DARPA in the project, *Military Science and Advanced Simulation* under contract number MDA903-90-C-0004.

[2] Both authors work at RAND, Santa Monica, CA.

I. INTRODUCTION

A common problem among military strategists and analysts is that of estimating the "strength" of a military unit. This estimate is needed to judge the success of operations, to compare the military forces of allies and opponents, and to determine "how much is enough" in defense budgeting. Yet military forces are composed of many distinct types of weapons and capabilities. Furthermore, success in battle is a function of many other factors including training, tactics, morale, terrain, command and control, etc. Ultimately an assessment is made on the basis of a score, that is, an aggregation of the force values into a single measure. Clearly, military experience and training permit some commanders to make good assessments of strength. However, this must be highly situation-dependent and the commander must have the required experience base.

When analysts evaluate new and perhaps undeveloped capabilities against possible threat forces they are frequently forced to create an aggregate estimate of the strength of these forces. Thus, a fundamental problem of military analysis is how to aggregate or "score" a military force. Usually, the weights come from what is considered experience, judgment, perhaps engineering or proving ground tests, etc.

Aggregation is required for other reasons as well. Historical data is often available in less detail than might be desired (overall losses rather than losses of specific weapons systems may be all that is known and certainly the exact causes of loss are usually unknown or unrecorded). Comprehension and understanding are frequently better served by more aggregate descriptions (the forest versus trees argument). Finally, analysis may require more efficient computation than is available with detailed battle simulations.

In general, the ability to aggregate and disaggregate combat forces and their processes is necessary and desirable-yet little

theory and science underlay most approaches taken in this regard.[3] Aggregation in linear dynamical systems has been a topic of interest, e.g., in economics[4] as well as in combat simulations, for some time. But, the typical focus has been on nearly decomposable systems or weakly coupled systems that may be amenable to approximated methods for solution wherein the system is partitioned into smaller subsystems for individual treatment followed by a reassembly for solution of the given system.[5] However, the current authors are unaware of a systematic treatment of the conditions necessary to effect a consistent aggregation in the sense defined here. Combat models exhibit numerous approaches but most are ad hoc. This paper examines some of these approaches, illustrates some of the problems, and then attempts to describe what is known and possible in a theoretical sense for basic square law Lanchester systems in dimensions higher than two.

The paper is organized to first illustrate some common approaches used in aggregation or scoring, particularly in military conflict simulation models. Then, some of the problems with these approaches are shown. Finally, using Lanchester theory as a basis, the requirements for theoretically "consistent" aggregation, disaggregation, and partial aggregation are described. An appendix provides the theorems and other mathematical considerations underlying the results in the main body of the text.

II. COMMON APPROACHES TO AGGREGATION AND DISAGGREGATION

Consider a military force composed of multiple types of weapons with different capabilities, the "combined arms" army for example. Each weapon type and supporting system is important to

---

[3] See Davis, P.K, and D. Blumenthal(1991), The Base of Sand Problem: A White Paper on the State of Military Combat Modeling, RAND, N-3148-OSD/DARPA.

[4] Simon, H.A., and Ando, A., "Aggregation of Vaiables in dynamic Systems," *Econometrica*, 29,pp.111-138, Reprinted in: Ando,A.,Fisher,F.M., and Simon,H.A., *Essays on the Structure of social Science Models, pp.64-91, MIT Press, Cambridge, Massachusetts,1963.*

[5] Gabriel Kron, "Solving highly complex elastic structures in easy stages," J.Appl.Mech.,22, pp 235-244, 1955.

prevent weakness (armor without infantry for example) or because of dependencies (artillery requires adequate fire control) and to provide synergism in battle (rolling artillery barrages to reduce the defender advantage as attacking armor moves into range). Yet these weaknesses, dependencies, and synergisms are influenced by terrain, opponent capability, and tactics to be employed. How can one aggregate or score such a situation dependent process?

The simplest form of aggregation involves assigning a value to each weapon type, multiplying by the number of each type and adding these values to obtain a force score or total aggregation. This is the approach taken by the WEI/WUV method[6] which, although once used widely, is not currently favored by the U.S. Army as an approach to modeling. Nevertheless it continues to be used in various models and academic debates[7]. Some uses of scoring have gone further. For example, scores have sometimes been disaggregated and the results used to estimate the losses of individual weapons systems or weapon categories and the ratios of force scores have been used to predict movement of forces in combat.[8] Almost all evaluations of military capability in models or exercises use some form of aggregation because the number of different systems is too large to consider directly.[9] The number of different systems in a typical mechanized army division may number 25 or more. Thus, partial aggregation of similar systems is a common approach. The subject of this paper is what can be said about the theoretical "correctness" of these various approaches to aggregation and disaggregation. We first take up the issue of total aggregation.

---

[6] Weapon Effectiveness Indices/Weighted Unit Values III(WEI/WUV III), US Army Concepts Analysis Agency, November 1979.

[7] For example, some models used in the public debate on conventional arms control in Europe and to estimate outcomes prior to the Desert Storm Operations used scored forces. See Bracken, J. Stability of Ground and Air Forces Without and With a Buffer, Phalanx, Volume 24, No. 2, June 1991.

[8] B. W. Bennett, C. Jones, A. Bullock, P. Davis(1988), Main Theater Warfare Modeling in the RAND Strategy Assessment System(3.0), RAND N-2743-NA.

[9] Ibid.

III. AGGREGATION TO SCALARS - SCORING

Scores or strength estimates have been necessary from the beginning of organized combat. Initially, with largely homogeneous forces, the scores could be based on the quantity of personnel available, and an estimate of individual strength. Even then the number that could engage at any one time was important and tactics and terrain could allow a smaller force to defeat a larger one. As forces became more heterogeneous it was necessary to give a relative evaluation of various components. The Soviets carried this to a scientific extreme with the *Correlation of Forces Methodology*[10] which attempted to predict the success of operations by evaluating force scores against time, position, and attrition objectives. A U.S. counterpart to this approach is the *Quantified Judgment Model* of Colonel T. N. Dupuy[11] which evaluates the outcome of battles based on a force score that is situation dependent and draws data from historical battles. The WEI/WUV scoring approach estimates strength in terms of division equivalents, and, more recently, an approach called Situation Adjusted Scores[12] attempts to estimate force strength with more attention to the situations of combat. Various models attempt to use these force aggregations for purposes of predicting campaign outcomes in terms of attrition, force movement and battlefield success.[13]

A number of problems arise in attempting to score complex military forces. First, the value of a force should depend strongly on the opponent and situation. A technically sophisticated armored force may not be of great military value in low intensity urban or jungle conflict. Advanced armored forces

---

[10] Hines, J,(1990),"Calculating War, Calculating Peace: Soviet Military Determinants of Sufficiency in Europe", in Reiner Huber(ed.), Military Stability, nomos Verlagsgesselschaft, Baden-Baden

[11] Dupuy, T. N.(1987), Understanding War, 1987, Paragon House Publishers.

[12] P. Allen, Situation Force Scoring: Accounting for Combined Arms Effects in Aggregate Combat Models, RAND N-3423-NA,(forthcoming).

[13] See for example RSAS description referenced above, or Epstein, Joshua M., The 3:1 Rule, the Adaptive Dynamic Model, and the Future of Security Studies, International Security, Spring 1989.

may be sitting ducks for an uncontested air force. The situational and opponent dependent aspects of scoring may have been implicitly included during the cold war when the opponent was assumed to be the Soviets and Warsaw Pact in Central Europe. But, when the opposition could be any of a number of possible enemies with widely varying capability, one scenario's aggregations would seem to be inappropriate for many others. Clearly, Soviet tanks in the hands of Iraqi soldiers during the recent Gulf War did not have the lethality (score) that they might have had when operated with trained Soviet or East German soldiers.

Determination of scores by subjective judgment has been unscientific to say the least. Experts, when used, must relate to experience. Yet, the experience base is limited to either specific contrived exercises which cannot be expected to represent warfare realistically and/or to historical experience with known weapons and situations. The evaluation of proposed or experimental weapon systems or new situations and opponents must therefore be done with guesswork by the experts.

As if the aggregation of a combined arms forces was not difficult enough, the attempt to use the aggregations to predict more than the likelihood of a single battle outcome is even more difficult. This requires a prediction of movement of forces in combat and, in order to predict the next battle, the composition of forces surviving. This means that the attrition results of earlier battles must be disaggregated. In mathematical parlance, this is a one-to-many mapping and it simply cannot be done uniquely without additional information. Given a 10% attrition of an aggregated force, what components of the force survive? Should the losses be evenly distributed according to number of systems, or should they be distributed by relative vulnerability, or relative lethality, or what? The following examples illustrate some of the problems with aggregation to scores and disaggregation.

Air/Ground Tradeoffs Using Scores

The first example is a simplification of a serious debate which took place during the development of NATO's conventional arms control position in the late 1980s.[14]

The basic question was whether tactical aircraft helped or hindered stability in the central region conventional force balance. It is assumed that defensive predominance is good and attacker predominance bad from the standpoint of stability. Let A be the attacker ground force score and D be the defender ground force score. The ground force ratio is then $FR = A/D$. In one analysis the air forces are added and considered to be killers of ground forces and therefore reduce the number of ground weapons and resulting scores. Assuming equal air forces for both attacker and defender, the forces removed by air attacks from each side is a. The resulting force ratio is then

$$FR = \frac{(A-a)}{(D-a)}$$

which favors the attacker under the doctrinal assumption that, as the attacker requires the greater force, we have $A > D$.[15] In this evaluation the air forces are considered to be destabilizing as they permit the attacker to gain more of an advantage.

In a countervailing evaluation the air forces are added and are considered to add firepower to each side. Thus, again assuming equal air forces, the firepower added is b to each side. The resulting force ratio is then

$$FR = \frac{(A+b)}{(D+b)}$$

---

[14] Bracken, op. cit. See also the comments following this article in the same journal.

[15] We assume that A > D otherwise the potential attacker would probably not attack.

which favors the defender under the same assumption that A > D and in this evaluation tactical aircraft could be seen to be stabilizing. A similar paradox is easily developed for helicopters, long range ground fires, etc. The important point is that the use of aggregation (of air power to ground forces) has somehow eliminated information that might be necessary in the evaluation of airpower and stability.

Disaggregation of Scored Results

There are two commonly used approaches to disaggregation in combat models. The simplest approach apportions the losses of individual weapon systems based on the initial proportion of the systems of each type. This means a 10% loss in aggregated score causes a 10% loss in the quantity of each type of system. This type of disaggregation is illustrated in the left curve in Figure 1 which is based on a square law Lanchester attrition calculation described in the next section. Note that this type of disaggregation keeps the proportion of weapons constant in time. On the right of Figure 1 the disaggregation has been done in proportion to the relative weight of the particular weapon used to create the score. That is, let w_1 and w_2 be the weights ("scores") of two weapon systems and let $N_1(t)$ and $N_2(t)$ be the number of those systems at time t. The score of this two weapon unit is

$$S(t) = w_1 N_1(t) + w_2 N_2(t).$$

Let the losses of this cored unit in and interval dt be denoted by dS and let these losses be computed by the same Lanchester square law. In this type of disaggregation the fractional losses of systems 1 and 2 in the interval dt are computed to be

$$\frac{dN_1}{N_1(t)} = \frac{w_1}{w_1 + w_2}\frac{ds}{S(t)} \quad \text{and}$$

$$\frac{dN_2}{N_2(t)} = \frac{w_2}{w_1 + w_2}\frac{ds}{S(t)}.$$

After extracting these losses for the interval, the system is re-scored and the Lanchester square law attrition is recomputed for

the next time interval. This results in a disproportionate drawdown and, of course, a different overall solution. The weaker component of the forces takes a proportionately larger amount of the loss as time advances. Either case could occur. Without additional information about tactics, how fire was allocated, etc. it is not possible to say which, if either, is correct. This is an illustration of the one-to-many mapping problems that are prevalent in attempts to disaggregate results in combat models. When these results are used in scoring the forces in the next stage of battle, the predicted campaign outcomes can be dramatically different depending on the approach taken.

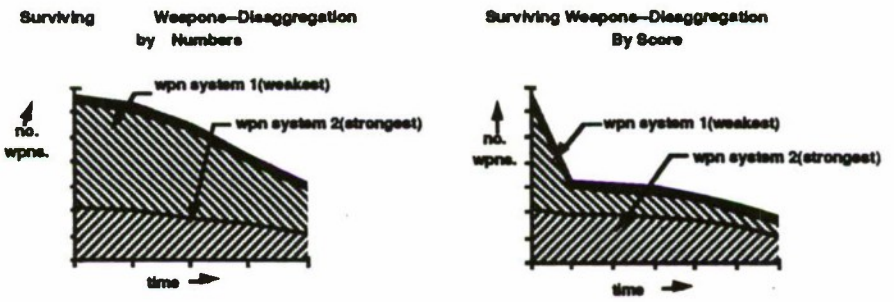


Fig. 1 Alternative Disaggregations of Scored Combat

IV. THEORETICAL RESULTS ON AGGREGATION AND DISAGGREGATION

Some important questions about aggregation and disaggregation are: How does an aggregation of a system relate to its more detailed representation? Given consistency requirements between two representations at different resolution, is there a "correct" way to aggregate? When is a consistent partial aggregation possible? When can aggregated combat results be disaggregated?

It is necessary to define the notion of consistency first. Figure 2 demonstrates the consistency between 2 models as defined in terms of the model output measures. At any time of interest the analysts should be able to compare the outputs of two models and, after an appropriate "mapping" the outputs should match within some small amount. The mapping is required because the analyst may only be interested in some aggregated outputs and, on the other hand, the outputs of one model may need to be converted into the similar measures produced by the other model. The differences between the outputs, because they represent a vector, must be measured in terms of a scalar norm of the differences. If this norm is small, then the models can be considered to have epsilon consistency with respect to the measures at the times of interest.

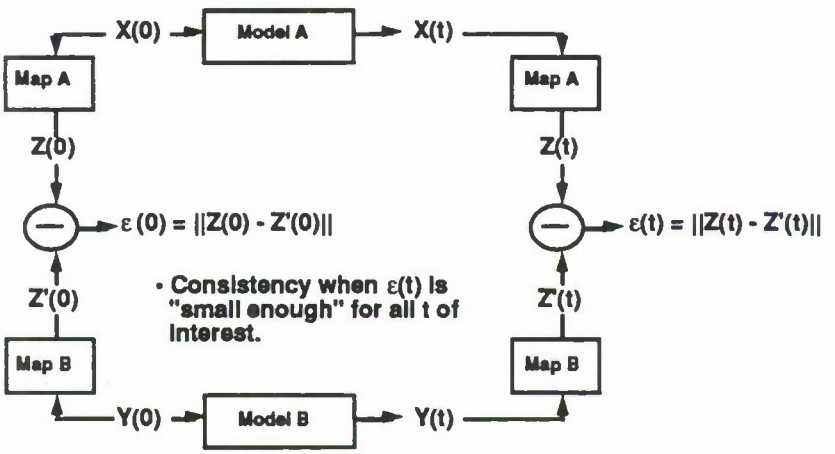


Fig. 2 Consistency in Dynamic Models

Figure 3 illustrates a slightly simpler representation of consistency. In this case we desire absolute consistency ($\varepsilon = 0$) between an aggregated model and a more detailed model. The mapping goes only one way—the outputs of the detailed model are

aggregated to match the aggregate model. When this mapping can be done with the same mapping or aggregation functions at all times and the outputs match exactly then we say the two representations have absolute consistency and are commutative. We now ask, under what conditions can we aggregate Lanchester square law combat models partially and completely and maintain absolute consistency?

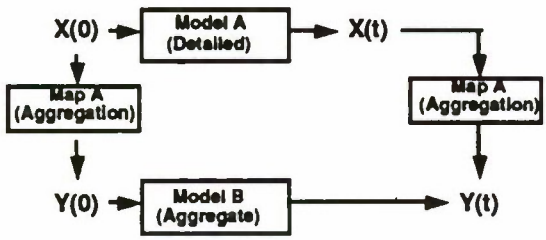


Fig. 3 Absolute Consistency in Aggregation

Because the constant coefficient, heterogeneous Lanchester square law[16] models lead to a system of linear differential equations, it is possible to derive and state strict theoretical results with respect to aggregation and disaggregation.[17] The square law Lanchester system is described by the pair of vector differential equations

---

[16] See James G. Taylor, Force-on-Force Modeling, Military Operations Research Society of America, 1981.

[17] It is clear that combat does not strictly follow a Lanchester Square law but understanding the requirements for aggregation and disaggregation of this "ideal" system is an important first step to understanding how it might be done in more complex models, say, possibly in ones represented by nonlineare differential equations that may be equivalent to higher-order ones as one-dimensional and some higher-dimensional Riccati equations are. This matter requires further study.

$$\frac{dX(t)}{dt} = -AY(t)$$

$$\frac{dY(t)}{dt} = -BX(t)$$

where $X = [x_1, x_2, ..., x_m]$ and $Y = [Y_1, Y_2, ..., Y_n]$ are the vectors of side X and side Y weapons systems. $A = [A_{ij}]$ and $B = [B_{ij}]$ are the Lanchester coefficient matrices defining the rate at which Y systems destroy X systems and vice versa, respectively.

An aggregation of this system is a reduced-dimension system

$$\frac{dU(t)}{dt} = -CV(t)$$

$$\frac{dV(t)}{dt} = -DU(t)$$

where $U(t)$ and $V(t)$ are aggregations of X and Y such that $U = RX$ and $V = SY$. The vectors U and V are of length r and s which are less than m and n (otherwise U and V would not be aggregations according to our definition). R and S are aggregation mappings (matrices) which are nonnegative (It's not clear what negative weights of weapons systems would imply). Thus, if R is a matrix of dimension 1 X n then the resulting U is a scalar. If R is 2 X n then the vector of X systems is reduced to 2 aggregate components that comprise U. It is not necessary that X and Y or U and V have the same dimensions. U might be a scalar in the aggregated system and V a vector of systems.

Consistent Scalar Aggregation

Consider the case in which the aggregation matrices R and S are vectors and therefore map X and Y to scalars. This, in effect, weights the components for X and Y into scalar "scores" for the two sides. Results from linear algebra and systems theory dictate the requirements on R and S such that consistency between

the resulting models is achieved. By consistency we mean that the result obtained from aggregating the solution of the unaggregated differential equation system is the same as the solution of the system of aggregated differential equations; i. e., in mathematical parlance, the operations of differential equation system solving and aggregation commute. We state the results here and provide the mathematical propositions and their proofs in the Appendix.

First, the eigenvalues of the product matrices AB and BA are desired to characterize the time response of the systems. That is, the time response for X and Y is the solution of the system of differential equations as a function of time, subject, of course, to initial conditions, $X(0) = X_0$, $Y(0) = Y_0$,

$$\begin{bmatrix} X(t) \\ Y(t) \end{bmatrix} = \begin{bmatrix} F(AB) & -G(AB)A \\ -G(BA)B & F(BA) \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix},$$

where

$$F(z) = \sum_{k=0}^{\infty} t^{2k} z^k / (2k)!$$

and

$$G(z) = \sum_{k=0}^{\infty} t^{2k+1} z^k / (2k+1)!,$$

both analytic functions over the finite complex plane and recognizable as $\cosh(t\sqrt{z})$ and $(\sinh(t\sqrt{z}))/\sqrt{z}$ respectively. Now, if we replace AB and BA by similarity transformations on their respective Jordan forms,

$$AB = W\Lambda W^{-1} \qquad \text{and} \qquad BA = Z\Gamma Z^{-1},$$

where the diagonal of Λ contains the eigenvalues of AB and the diagonal of Γ contains the eigenvalues of BA, we have a more computationally palatable form

$$\begin{bmatrix} X(t) \\ Y(t) \end{bmatrix} = \begin{bmatrix} WF(\Lambda)W^{-1} & -WG(\Lambda)W^{-1}A \\ -ZG(\Gamma)Z^{-1}B & ZF(\Gamma)Z^{-1} \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix}.$$

Fact *The matrices AB and BA are nonnegative matrices with non-null columns and non-null rows and, consequently, have maximal positive eigenvalues and corresponding nonnegative left eigenvectors whose components are the weights that yield commutative or consistent scalar aggregations of X and Y* [18].

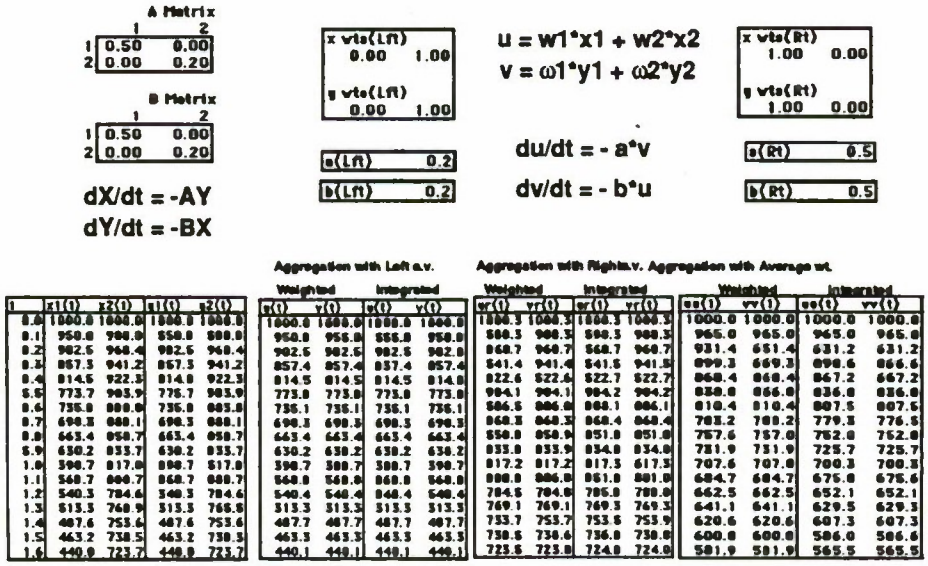
It is best to give an example of such an aggregation. Table 1 illustrates a consistent aggregation for a 2 X 2 Lanchester system reduced to a single component on each side. The original *A* and *B* matrices are shown at the upper left, possible *C* and *D* matrices (scalars in this case) are shown with the weights derived from the eigenvectors. There are two sets of weights because there are two possible eigenvalues and associated vectors.[19] The left part of the table shows the direct integration of the heterogeneous system in *X* and *Y* as a function of time, the columns labeled as **weighted** are obtained by weighting the timewise values of *X* and *Y*. The columns labeled as **integrated** u(t) and v(t) are obtained by using the *R* and *S* obtained with the left eigenvalues and integrating initial values u(0) and v(0). The fact that these match at each time t shows the consistency of the aggregation. The columns labeled ur(t) and vr(t) demonstrate that consistency is also achieved with weights based on the other (right) eigenvectors.[20]

---

[18] The reason the product matrices, AB and BA, are important is seen by differentiating the equation for dX(t)/dt. This gives $d^2X(t)/dt^2 = -AdY(t)/dt = ABX(t)$, with a similar argument for BA. Thus, the solutions depend on AB and BA, rather than on A and B separately.

[19] Consistency applies only as long as no force components become negative. When this happens the original systems and aggregations must be changed as negative weapons have no meaning. In the original systems one simply deletes those components of X or of Y that go to zero and the corresponding rows and columns in A and B and proceeds with the reduced system of equations with, of course, new aggregation operators R and S.

[20] There is an optional scale factor that can also be applied to change the overall magnitude of the aggregated U and V components.



t	x1(t)	x2(t)	y1(t)	y2(t)	X(t)	Y(t)	X(t)	Y(t)	Xr(t)	Yr(t)	Xr(t)	Yr(t)	uu(t)	vv(t)	uu(t)	vv(t)
0.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.3	1000.3	1000.3	1000.3	1000.0	1000.0	1000.0	1000.0
0.1	950.0	980.0	950.0	980.0	950.0	956.0	956.0	950.0	980.3	980.3	980.3	980.3	965.0	965.0	965.0	965.0
0.2	902.5	960.4	902.5	960.4	902.5	902.5	902.5	902.5	960.7	960.7	960.7	960.7	931.4	931.4	931.2	931.2
0.3	857.3	941.2	857.3	941.2	857.4	857.4	857.4	857.4	941.4	941.4	941.5	941.5	899.3	899.3	898.6	866.6
0.4	814.5	922.3	814.0	922.3	814.5	814.5	814.5	814.0	922.6	922.6	922.7	922.7	868.4	868.4	867.2	867.2
0.5	773.7	903.9	776.7	903.9	773.8	773.8	773.8	773.8	904.1	904.1	904.2	904.2	838.8	866.8	836.8	836.8
0.6	735.8	885.8	735.8	885.8	736.1	736.1	736.1	736.1	886.5	886.5	886.1	886.1	810.4	810.4	807.5	807.5
0.7	698.8	868.1	698.5	868.1	698.3	698.3	698.3	698.3	868.3	868.3	868.4	868.4	783.2	788.2	779.1	776.5
0.8	663.4	850.7	663.4	850.7	663.4	663.4	663.4	663.4	850.8	850.9	851.0	851.0	757.6	757.0	752.0	752.0
0.9	630.2	833.7	630.2	833.7	630.2	630.2	630.2	630.2	833.8	833.9	834.0	834.0	731.9	731.9	725.7	725.7
1.0	598.7	817.0	598.7	817.0	598.7	598.7	598.7	598.7	817.2	817.2	817.3	817.3	707.6	707.0	700.3	700.3
1.1	568.7	800.7	568.7	800.7	568.8	568.8	568.8	568.8	800.8	800.8	801.0	801.0	684.7	684.7	676.8	676.6
1.2	540.3	784.6	540.3	784.6	540.4	540.4	540.4	540.4	784.5	784.6	785.0	785.0	662.5	662.5	652.1	652.1
1.3	513.3	768.9	513.3	768.9	513.3	513.3	513.3	513.3	769.1	769.1	769.3	769.3	641.1	641.1	629.5	629.3
1.4	487.6	753.6	487.6	753.6	487.7	487.7	487.7	487.7	753.7	753.7	753.8	753.9	620.6	620.6	607.3	607.3
1.5	463.2	738.5	463.2	738.5	463.3	463.3	463.3	463.3	738.5	738.6	738.8	738.8	600.8	600.8	586.0	586.6
1.6	440.0	723.7	440.0	723.7	440.1	440.1	440.1	440.1	723.6	723.8	724.0	724.0	581.9	581.9	565.5	565.5

Table 1 Consistent Scalar Aggregation

Consider what has been lost by the aggregations when one chooses such a consistent set of weights. The two-dimensional Lanchester system in Table 1 has only positive components on the diagonals. These are actually the equations for two separate battles. X_1 shoots at Y_1 and Y_1 shoots X_1. X_2 shoots at Y_2 and Y_2 shoots at X_2 but there are no shots between X_1 and Y_2, X_2 and $Y1$ or vice versa. As a result of the preceding, we have that the consistent aggregations weight the components of X with R = [0 1] or R = [1 0] and similarly S = [0 1] or S= [1 0]. Thus, either one battle or the other can be represented in a consistent aggregation and there is no consistent aggregation for an "average" battle incorporating components of both.[21] The columns labeled uu and vv represent an attempt to approximate an average battle by averaging the weights. Comparison of the columns

[21]This can be understood intuitively by considering the fact that the different components will generally attrit at different rates leading to a continuously changing mix of systems.

labeled weighted with the integrated columns indicates that consistency has not been maintained in this attempt.

The point of this example is to show that something is lost in an aggregation, namely some aspect of the response of the detailed system. And as the illustration shows, the components left off may be important (in this case, the other battle). Figure 4 illustrates this example, showing the aggregated response matching the X_1, Y_1 components with one set of weights and matching the X_2, Y_2 components with the other set of weights.

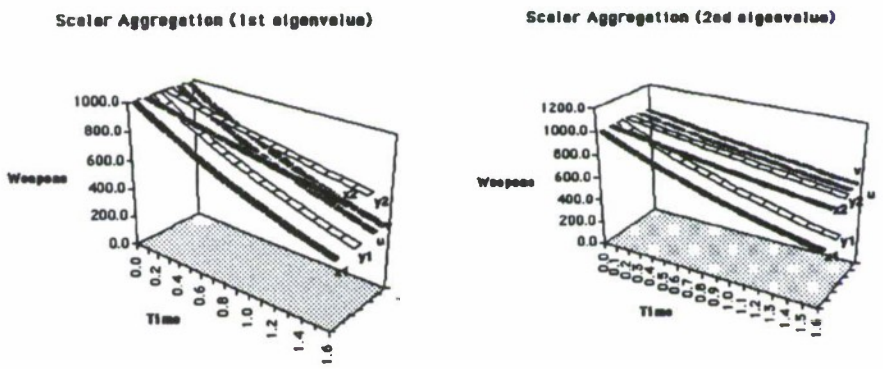


Fig. 4 Consistent Scalar Aggregation Alternatives

Partial Aggregation and Disaggregation of Lanchester Systems

It is common to group systems by type in a combat model and create an aggregated weapon type for each such grouping. Fighter aircraft of similar capabilities might be grouped into equivalent fighter systems or all tanks might be aggregated into an "equivalent" tank. This implies a partial aggregation of the system and, in terms of the earlier described aggregation mappings, we define R to have the canonical form

$$R = \begin{bmatrix} [\ \rho_1\ ][0.....0] \\ [0.....0][\ \rho_2\ ][0.....0] \\ \cdots \\ [0.....0][\ \rho_r\ ] \end{bmatrix}$$

where each ρ_i is a nonnull vector of nonnegative weights on specific components of X. We rule out aggregation of a component into two different aggregate components although it could be treated; but the treatment in what follows, particularly in the appendix, would be laborious in its details and not likely to be illuminating.

For the heterogeneous Lanchester system we can state the following two facts about the ability to do this and maintain consistency between the two models. These results are established in the Appendix.

Fact 1 (Partial Aggregation): One cannot generally partially aggregate a square law Lanchester system so that U and V are vectors and consistency is maintained without applying additional restrictive conditions on the matrices A and B.

Fact 2 (Disaggregation): It is not generally possible to disaggregate a previously aggregated system and obtain a system consistent with the original disaggregated system without additional restrictive conditions on the matrices A and B.

The condition required to partially aggregate or disaggregate a square law Lanchester system is that there is a constant relative effectiveness or vulnerability between the various components of X and Y. For example, the following is shown in the Appendix.

Fact 3 (Partial Aggregation): Partial aggregation is possible (consistency maintained) when there is a constant proportionality in the effects of some of the weapons of a side with respect to all of the weapons of the other side.

For example, if weapon 1 of side 1 is twice as effective as weapon 2 of the same side against all systems (components of Y)

then weapon 1 and weapon 2 can be aggregated into a single representative weapon and this aggregated model will correctly predict the square law attrition of each component of *Y*. This partially aggregated system model is consistent with the non-aggregated model if the weighting of the two weapons is proportional to their relative effectiveness.

Figure 5 illustrates this case in which components X_1 and X_2 have been aggregated while X_3, Y_1, Y_2, and Y_3 remain disaggregated. The trajectories of the components of *Y* remain the same regardless of whether the aggregated or disaggregated *X* is used in the equations.

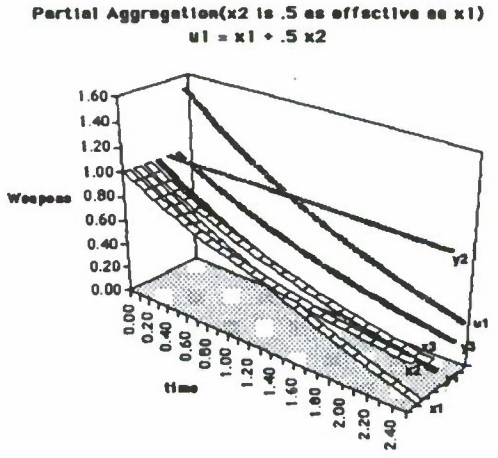


Fig. 5 An Example of Partial Aggregation

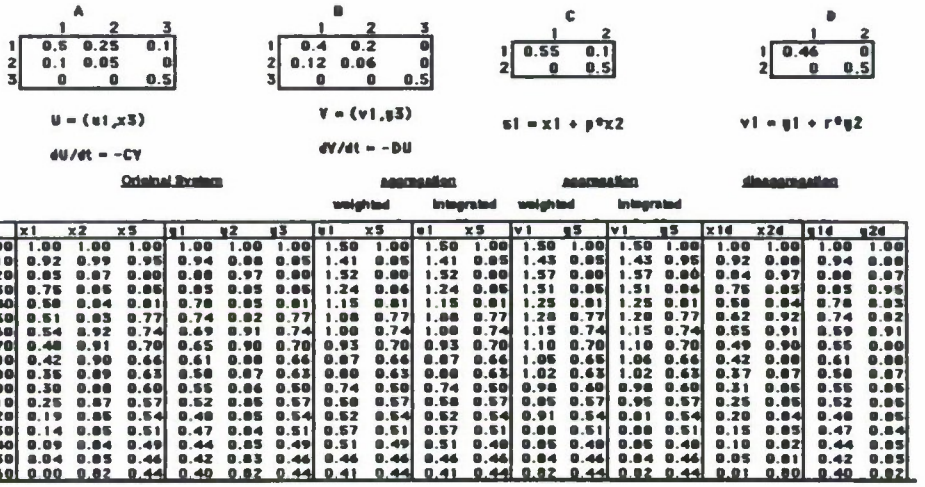
This condition is not enough to be able to restore X from its aggregated state at some later time, however, This is because the system may have differential attrition of the components of X (X_1 and X_2 in the example) so that the number of surviving elements varies over time. Once aggregated, without additional information, there is no way to turn U back into X_1 and X_2. What

is a condition that would permit this? In the appendix we show the following is true.

Fact (Disaggregation): If, in addition to the previously stated relative effectiveness of two or more systems, the same systems also have a proportionality in vulnerability that is constant with respect to all systems of the opponent, then it is possible to both aggregate these systems consistently and at any point in time this aggregated system can be disaggregated back to the original more detailed model.

In other words, if we can say that weapon 1 is twice as effective against all weapons as weapon 2 and that weapon 1 is say, one third as vulnerable as weapon 2 to all opposing systems, then the detailed model can be partially aggregated and later disaggregated while satisfying the consistency requirement.

These are the conditions under which the scoring of weapons and later disaggregation of those scores make the most sense because the aggregated values can be used without any loss of information. The information needed to restore the system is in the knowledge about relative effectiveness and vulnerability. As a final example, consider the 3 component model shown in Table 2. The columns of Table 2 shows the consistency between the aggregation of the first two components of a 3 dimensional example(3 weapon systems) and the original system. It also shows the reconstruction of the original system from the aggregated components (possible only because the respective components in the differential equations are linearly related) Note that consistency is achieved up until t = 1.6 at which time x_1 goes to zero. At this point the original Lanchester system must be reduced to leave out x_1 and all aggregations recalculated to maintain consistency. In this example we stopped the calculations at this point.



t	x1	x2	x5	y1	y2	y3	y1	x5	y1	x5	v1	y5	v1	y5	x1d	x2d	y1d	y2d
0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.50	1.00	1.50	1.00	1.50	1.00	1.50	1.00	1.00	1.00	1.00	1.00
0.10	0.92	0.99	0.95	0.94	0.98	0.85	1.41	0.85	1.41	0.85	1.43	0.85	1.43	0.95	0.92	0.98	0.94	0.89
0.20	0.85	0.97	0.90	0.88	0.97	0.80	1.32	0.80	1.32	0.80	1.37	0.80	1.37	0.90	0.84	0.97	0.88	0.87
0.30	0.78	0.95	0.86	0.83	0.95	0.85	1.24	0.86	1.24	0.85	1.31	0.86	1.31	0.86	0.76	0.95	0.83	0.95
0.40	0.69	0.94	0.81	0.78	0.93	0.81	1.15	0.81	1.15	0.81	1.25	0.81	1.25	0.81	0.69	0.94	0.78	0.83
0.50	0.61	0.93	0.77	0.74	0.92	0.77	1.08	0.77	1.08	0.77	1.20	0.77	1.20	0.77	0.62	0.92	0.74	0.82
0.60	0.54	0.92	0.74	0.69	0.91	0.74	1.00	0.74	1.00	0.74	1.15	0.74	1.15	0.74	0.55	0.91	0.69	0.91
0.70	0.48	0.91	0.70	0.65	0.90	0.70	0.93	0.70	0.93	0.70	1.10	0.70	1.10	0.70	0.49	0.90	0.65	0.80
0.80	0.42	0.90	0.66	0.61	0.88	0.66	0.87	0.66	0.87	0.66	1.06	0.65	1.06	0.66	0.42	0.88	0.61	0.88
0.90	0.35	0.89	0.63	0.58	0.87	0.63	0.80	0.63	0.80	0.63	1.02	0.63	1.02	0.63	0.37	0.87	0.58	0.87
1.00	0.30	0.88	0.60	0.55	0.86	0.60	0.74	0.60	0.74	0.60	0.98	0.60	0.98	0.60	0.31	0.86	0.55	0.86
1.10	0.25	0.87	0.57	0.52	0.85	0.57	0.68	0.57	0.68	0.57	0.95	0.57	0.95	0.57	0.25	0.85	0.52	0.86
1.20	0.19	0.86	0.54	0.48	0.85	0.54	0.62	0.54	0.62	0.54	0.91	0.54	0.91	0.54	0.20	0.84	0.48	0.85
1.30	0.14	0.85	0.51	0.47	0.84	0.51	0.57	0.51	0.57	0.51	0.88	0.51	0.88	0.51	0.15	0.85	0.47	0.84
1.40	0.09	0.84	0.49	0.44	0.85	0.49	0.51	0.49	0.51	0.48	0.85	0.48	0.86	0.48	0.10	0.82	0.44	0.85
1.50	0.04	0.85	0.46	0.42	0.83	0.46	0.46	0.46	0.46	0.46	0.84	0.46	0.84	0.46	0.05	0.81	0.42	0.85
1.69	0.00	0.82	0.44	0.40	0.82	0.44	0.41	0.44	0.41	0.44	0.82	0.44	0.82	0.44	0.01	0.80	0.40	0.82

Table 2 Consistent Partial Aggregation and Disaggregation

V. CONCLUSIONS

We have shown that aggregation and disaggregation should be performed carefully in models of combat if consistency is to be preserved; but it can be done, provided that the attrition matrices satisfy certain additional reasonable conditions when the aggregations are not to be total. Although we do not suggest that the Lanchester square law is a realistic depiction of actual conflict, the fact that, even for this linear system of equations, consistent aggregation and disaggregation cannot be done without severe restrictions on the attrition matrices *A* and *B* implies that ad hoc approaches to varying resolution may not lead to consistent models. The aggregation weights cannot be arbitrary. Once the attrition matrices *A* and *B* are specified, the aggregation weights, when consistent aggregation is possible, are dictated by mathematical considerations, as is shown in the appendix.

Clearly, the absence of general conditions of the type shown here for more complex models implies that empirical tests

comparing models of different resolution should be made before conclusions can be drawn regarding the goodness of an aggregation.

We do not mean to argue that detailed models are always better. Often the aggregate results of combat (overall losses and advances of front lines) are known from history, but the details of specific forces, conditions, tactics, effectiveness, fire allocation, etc. are missing. This means that an aggregate model can be tested or fit to the data but an attempt to extrapolate to detailed losses is highly subjective. Thus, the most correct model based on empirical data could very well be the aggregate, low resolution one. On the other hand, the frequent absence of any empirical data regarding how forces or weapons might fare in battle has often forced analysts to build models in high detail in hopes that engineering test data can be extrapolated to combat outcomes. Frequently, however, this approach amounts to compounding assumptions upon assumptions regarding interactions in conflict that are completely subjective.

The material in this research sheds very little light on this problem. It does suggest, as noted earlier, that aggregation and disaggregation cannot be done arbitrarily and that fairly strong requirements are necessary to obtain consistent high and low resolution models.

Further research in this area should examine when models and aggregations can be made partially consistent. That is, suppose the conditions for aggregations and disaggregations stated in this paper are approximately satisfied. What does this imply regarding how far apart the aggregate solution of the unaggregated differential equations are from the solution of the aggregated differential equations as time advances?

Another area of investigation to consider is the possibility of extension of this research to Lanchester models with nonlinear differential equation formulations that may be equivalent to higher order linear ones as, for example certain higher-dimensional Riccati differential equation systems with special structure are.

APPENDIX

MATHEMATICAL CONSIDERATIONS

I. PRELIMINARIES

We consider here a rather sharply defined Lanchester System model of continuous combat between two opposing forces whose resources fall into different classes or types, e. g., tanks of one or more types, missile launchers, personnel vehicles, howitzers, helicopters, troops of different types, etc. The numbers of each resource type form the different components of a strictly positive m x 1 vector X for one side and of a strictly positive n x 1 vector Y for the other side.

The model assumes that the state of the battle at a time t > 0 is represented by the pair (X, Y) which are obtained as the solutions of the system of ordinary equations mentioned earlier

$$\begin{aligned} \frac{dX}{dt} &= -AY, \quad X(0) = X_0 > 0 \\ \frac{dY}{dt} &= -BX, \quad Y(0) = Y_0 > 0 \end{aligned} \tag{1}$$

where A: m x n and B: n x m are the nonnegative attrition matrices. The elements a_{ij} of A represent the time rate of attrition of the i-th resource x_i by one unit of the j-th resource y_j of the opponent. The elements b_{ij} of B are similarly defined with x and y interchanged.

We note that the components of the vectors, X and Y, in the solution of the equation (1) are monotone decreasing functions of time. When any component, or components, of these vectors reaches zero, we consider the time duration of this system, but not necessarily the battle, to be terminated. A new system, if the battle is to continue, is created with its dimensions reduced by the number of components that have reached zero; these components

and those rows and columns of the matrices that are associated with them are removed.

The new system is still a Lanchester system with the same generic form and properties as has the system (1) with initial conditions for the positive vectors being their respective values they attained at the end of the previous time period. Although the same battle continues, we simply consider it as continuing under a new system, new only in the sense that its dimensions have been reduced and its initial conditions changed.

This process of "culling" the system of variables and associated rows and columns in the attrition matrices (that could, if not culled, lead to the absurdity of negative numbers of resources) can be continued so long as there are positive components left on each side or earlier, if it is deemed that the battle is over.

We further make a reasonable assumption of choice that each resource on either side has some attriting effect on at least one resource of the opponent and each resource on either side is vulnerable to attriting effects of at least one resource on the opponent's side. Thus, not only are A and B matrices of nonnegative components, but we require that they have no completely null column and no completely null row. This property of no null columns and no null rows in the attrition matrices with their nonnegativity is sufficient for their respective products to satisfy the hypotheses of one or the other of the Perron-Frobenius theorems on the existence of a maximal positive eigenvalue and corresponding positive or nonnegative eigenvector for nonnegative matrices that we will use later on[22]. (Of course, one can easily conceive of resources that have value that is not of an attriting nature, e.g., petroleum supplies, or of an invulnerable nature, e.g., aircraft against an enemy with no air defenses; and, one can deal with such cases). However, for some mathematical convenience later, namely, to avoid the potential for zero Perron eigenvalues

[22]See, e.g., Richard S. Varga, Matrix Iterative Analysis, Prentice Hall Inc., Englewood Cliffs, N.J. 1962 [ch. 2].

and concomitant complications, we exclude these cases by our assumptions above.)

As is to be expected from the general theory for systems such as (1), its solution is an analytic vector function of the initial condition vectors. We define

$$(2) \qquad F(z)=\sum_{k=0}^{\infty}\frac{t^{2k}z^k}{(2k)!} \qquad \text{and} \qquad G(z)=\sum_{k=0}^{\infty}\frac{t^{2k+1}z^k}{(2k+1)!}.$$

Then, we have for the solution of (1)

$$(3) \qquad \begin{bmatrix}X(t)\\ Y(t)\end{bmatrix}=\begin{bmatrix}F(AB) & -G(AB)A\\ -G(BA)B & F(BA)\end{bmatrix}\begin{bmatrix}X_0\\ Y_0\end{bmatrix},$$

for as long as no component of the solution vector becomes negative. For $z = AB$ or for $z = BA$ computing a solution in the form (3) can be laborious; consequently, employing the similarity transformations,

$$AB=W\Lambda W^{-1} \qquad \text{and} \qquad BA=Z\Gamma Z^{-1},$$

we have a much more computationally tractable form

$$\begin{bmatrix}X(t)\\ Y(t)\end{bmatrix}=\begin{bmatrix}WF(\Lambda)W^{-1} & -WG(\Lambda)W^{-1}A\\ -ZG(\Gamma)Z^{-1}B & ZF(\Gamma)Z^{-1}\end{bmatrix}\begin{bmatrix}X_0\\ Y_0\end{bmatrix},$$

where the diagonals of the Jordan canonical form matrices (even possibly diagonal), Λ and Γ, contain the eigenvalues of AB and BA, respectively.

However, it may be that the dimensionality of the problem is still too high either for computational reasons or, more importantly, because the detail is too fine for understanding the progress of the battle at various levels of generality or resolution. Thus, to relieve these objections to the size of the dimensionality of the original problem, one frequently employs

weighted linear aggregations of the resources on either side either into a single scalar value for each side (total aggregation) or into smaller numbers of groups of resources than the original numbers (partial aggregation) or, perhaps, even total scalar aggregation on one side and no aggregation on the other side (unilateral aggregation). Having presented some illustrative examples earlier, our purpose here is to examine mathematically some aspects of aggregation and conditions for potential disaggregation.

We define linear dimension-reducing aggregation operators, R and S, to be r x m and s x n nonnegative matrices, respectively, with r and s strictly less than m and n, respectively, such that

(4) $$U = RX \qquad \text{and} \qquad V = SY$$

form a reduced dimension Lanchester system in the above defined sense. Thus, we have a system of ordinary differential equations for U and V of similar form to (1) for X and Y

(5)
$$\frac{dU(t)}{dt} = -CV(t), U(0) = U_0 > 0,$$
$$\frac{dV(t)}{dt} = -DU(t), V(0) = V_0 > 0,$$

where C and D are new r x s and s x r nonnegative attrition matrices, respectively, with the same properties of having nonnull rows and nonnull columns as A and B.

The solution to the reduced-dimension aggregated Lanchester system (5) is of exactly the same form as the solution (3) of (1), namely,

(6)
$$\begin{bmatrix} U(t) \\ V(t) \end{bmatrix} = \begin{bmatrix} F(CD) & -G(CD)C \\ -G(DC)D & FDC) \end{bmatrix} \begin{bmatrix} U_0 \\ V_0 \end{bmatrix}$$

The aggregation operators R and S appearing in (4) are defined to be in canonical form if they are in the forms illustrated below

(7)
$$R=\begin{bmatrix}[\ \rho_1\ ][0.....0] \\ [0.....0][\ \rho_2\ ][0.....0] \\ \cdots \\ [0.....0][\ \rho_r\ ]\end{bmatrix}$$

and

(8)
$$S=\begin{bmatrix}[\ \sigma_1\ ][0.....0] \\ [0.....0][\ \sigma_2\ ][0.....0] \\ \cdots \\ [0.....0][\ \sigma_s\ ]\end{bmatrix},$$

where the row vectors ρ_i, i=1,...,r, and σ_i, i=i,...,s, are nonnull and nonnegative and are of respective dimensions m_i and n_i. For convenience, we shall consider aggregation operators in the above canonical forms (7) and (8). (Conceivably one could presume aggregations that allow a resource to be in two or more aggregate groups; but, because of the additional complexity of details in their treatment, we conveniently refrain from examining such cases.)

A central desideratum for aggregation operators, one that makes for a sense of consistency, is that the operations of differential equation solving and aggregation commute. That is, the result obtained by first solving the ordinary differential equation system (1) to obtain the solutions given by (3), or of any other equivalent form, and then applying the dimension-reducing aggregation operators, R and S, to the solutions is identical to the result of first applying the operators, R and S, to the differential equations (1) to get the differential equation system (5) in U and V and then solving this system (5) to get the reduced-dimension result (6).

In the following sections we shall be concerned with presenting conditions on *A, B, R, S, C,* and *D* that will produce the desired commutativity in the different situations of total (scalar) and partial aggregation; verification that the matrices *AB* and *BA* satisfy the hypotheses of one or the other of the Perron-Frobenius theorems mentioned below, which are useful in determining *R* and *S* that preserve the desired commutativity; and conditions for the inversion of aggregation, i.e., disaggregation.

II. GENERAL CONDITIONS FOR THE PRESERVATION OF LANCHESTER SYSTEM PROPERTY AND COMMUTATIVITY OF AGGREGATION AND DIFFERENTIAL EQUATION SOLVING

Theorem: *If aggregation operators R, r x m, and S, s x n, and nonnegative matrices C and D of compatible dimensions exist such that*

(9) $$RA = CS \quad \text{and} \quad SB = DR,$$

then U and V defined by (4) will produce a Lanchester system (5) from the Lanchester system (1). Moreover, with compatible initial conditions, the solution (6) of (4) will be identical to the pair, (RX, SY), where (X, Y) is the solution (3) of (1).

Proof: Let *X* and *Y* satisfy a Lanchester system of differential equations (1). Suppose that nonnegative *R, S, C,* and *D* exist such that (9) holds. Then, we have

$$\frac{dU}{dt} = R\frac{dX}{dt} = -RAY = -CSY = -CV$$

and

$$\frac{dV}{dt} = S\frac{dY}{dt} = -SBX = -DRX = -DV\,.$$

Hence, with positive initial values, U_0 and V_0, for *U* and V, respectively, *U* and *V* form a Lanchester system pair as in (5).

Now, consider any analytic function, *f*(z) (particularly *F*(z) and *G*(z) defined in (2)), representable by an absolutely convergent power series in z in some disk of the complex plane

with a radius exceeding the spectral radii of AB, BA, CD, and DC (Those of AB and BA are equal and those of CD and DC are also.). Then, applying the associative law for matrices successively, we have that the relations (9) imply that $R(AB) = C(SB) = (CD)R$, $S(BA) = (DC)S$, $R(AB)^k = (CD)^kR$, $S(BA)^k = (DC)^kS$, and, finally,

$$(10) \qquad Rf(AB) = f(CD)R \quad \text{and} \quad Sf(BA) = f(DC)S.$$

Let the U and V satisfy (5) with initial values of $U_0 = RX_0$ and $V_0 = SY_0$, respectively. Then, applying R to X, S to Y, and (10) with f replaced by F and G as defined in (2) in the solution (3) to equation (1) completes the proof of the theorem, Q. E. D.

III. BILATERAL TOTAL (SCALAR) AGGREGATION

In this section, we are concerned with defining the row vectors R and S which totally aggregate X and Y down to scalars and, consequently, make U, V, C, and D positive scalars. We will make use here of the two Perron-Frobenius Theorems:[23]

(i) *Let M be a nonnegative, irreducible square matrix. Then it has a positive eigenvalue (called the Perron eigenvalue) that is simple and is not exceeded by the absolute value of any of its other eigenvalues. Furthermore, corresponding to this eigenvalue is a positive eigenvector (left eigenrow or right eigencolumn).*

(ii) *Let M be a nonnegative, reducible, square matrix, but for which there exists no permutation matrix, P, such that PMP^T is a strictly upper triangular matrix. (The superscript T denotes transpose.) Then it has a positive eigenvalue that is equal to its spectral radius and a corresponding nonnegative, nonnull left eigenrow and right eigencolumn.*

---

[23] Ibid.

(A square matrix, *M*, is reducible if there exists a permutation matrix, *P*, (i.e., a square matrix whose only nonzero elements are a single 1 in each row and in each column) such that

$$PMP^T = \begin{bmatrix} N_1 & N_2 \\ 0 & N_3 \end{bmatrix},$$

where N_1 and N_3 are square matrices and 0 is a null matrix; *M* is irreducible if no such a permutation matrix exists. A square matrix is strictly upper triangular if all of its elements are zero except those strictly above the main diagonal, which are unrestricted.)

Theorem: *If no row or column of either of the nonnegative matrices A or B is a null vector, then there exist no permutations, P or Q, such that either* $AB = PABP^T$ *or* $BA = QBAQ^T$ *is similar to a strictly upper triangular matrix.* (Hence, *AB* and *BA*, being nonnegative, also satisfy the hypotheses of one or the other of the Perron-Frobenius theorems, depending on whether they are irreducible or reducible.)

Proof: For the sake of a contradiction later, let us assume that, by permuting the rows and columns of *AB* (via the same permutation), *AB* is transformed into a strictly upper triangular matrix. Then this strictly triangular matrix has a null first column and a null last row. Let that column of *AB* that becomes the first column under the permutation be the j-th column for a particular j. Let *K* be that set of integers k such that $b_{ij} > 0$ whenever i = k. *K* is not empty by the hypothesis of the theorem. Since $a_{ik}b_{kj}$ must = 0 for every i = 1,...,m and k=1,...,m (*AB* being m x m), the k-th columns of *A* must be null whenever k is in *K*, contrary to the hypotheses of the theorem.

Similarly, let that row of *AB* that becomes the last under the permutation be the i-th for a particular i. Anew, let K be that set of integers k such that $a_{ij} > 0$ whenever j = k. Arguing similarly to the above, we must have that the k-th rows of B must

be null whenever k is in K. Again, we have a contradiction to the hypotheses of the theorem.

Reversing the roles of *A* and *B* in the arguments above and applying them to the n x n matrix BA, we again arrive at the findings that A must have some null row or rows and *B* must have some null column or columns, contrary to the hypotheses of the theorem. Therefore, neither *AB* nor *BA* is similar under permutation to a strictly upper triangular matrix, Q. E. D.

Thus *AB* and *BA* satisfy the hypotheses of the Perron-Frobenius theorems. Now, it is easy to demonstrate that if *A* and *B* are square matrices then they have the same set of eigenvalues; but, what can be said if they are not?

Let λ and ρ be the Perron eigenvalue and corresponding left eigenrow for *AB* and μ and σ be the same for *BA*. For convenience, though it is not necessary, let the norms of these two eigenrows be unity.

Lemma: *With the above definitions, we have that* $\lambda = \mu$ *and, if these eigenvalues are simple, then* $\rho A/|\rho A| = \sigma$ *and* $\sigma B/|\sigma B| = \rho$.

Proof: The assumption that λ and μ are Perron eigenvalues of the respective matrices *AB* and *BA* implies that λ is also an eigenvalue of *BA* and therefore $\lambda \leq \mu$. Similarly, μ is also an eigenvalue of *AB* and therefore $\mu \leq \lambda$. Hence, $\lambda = \mu$.

If *AB* and *BA* are irreducible then their Perron eigenvalue is simple. If they are reducible then we must assume additionally that the maximum of all of the Perron eigenvalues of the submatrices on the main diagonal is unique. Then, multiplying $\rho(AB) = \lambda\rho$ by *A* on the right on both sides and applying the associative law, we have that $(\rho A)(BA) = \lambda(\rho A)$. Hence, since λ is simple, we have that $\rho A/|\rho A| = \sigma$. Similar argument yields that $\sigma B/|\sigma B| = \rho$, Q.E.D.

Therefore, for total aggregation on both sides, we define

$$R = k\rho/|\rho A| \quad \text{and} \quad S = \kappa\sigma/|\sigma B|.$$

In order to achieve our desideratum of consistency, i. e., commutativity of aggregation and differential equation solving, we impose the hypothesis of the theorem in section II of this appendix and the requirement that the Perron eigenvalues of *AB* and *BA* are simple if so needed when these matrices are reducible. Then, from the above lemma, we have that

$$RA = \frac{k(\rho A)}{|\rho A|} = CS = \frac{\kappa C \sigma}{|\sigma B|} = \frac{\kappa C(\rho A)}{|\sigma B||\rho A|}$$

and

$$SB = \frac{\kappa(\sigma B)}{|\sigma B|} = DR = \frac{kD\rho}{|\rho A|} = \frac{kD(\sigma B)}{|\rho A||\sigma B|},$$

from which we get

$$k = \frac{\kappa C}{|\sigma B|},$$
$$\kappa = \frac{kD}{|\rho A|},$$

and

$$CD = |\rho A||\sigma B|.$$

Furthermore, from this, $\rho AB = \lambda\rho$, $\sigma BA = \lambda\sigma$, and the lemma, when the eigenvalues are simple, we have that $\lambda\rho = \sigma B|\rho A|$ and $\lambda\sigma = \rho A|\sigma B|$. Taking absolute values of either, we conclude that $\lambda = |\rho A||\sigma B|$ and $CD = \lambda$. Some freedom of scaling in aggregation exists; but *C*, *D*, *k*, and κ are not entirely arbitrary. They must satisfy the above relations.

This concludes the definition of the total (scalar) aggregation operators (row vectors, in this case) sufficient to produce commutativity of the aggregation and the solution of the Lanchester differential equations.

IV. UNILATERAL TOTAL AGGREGATION

For the sake of definiteness, let us assume that it is the resources in the vector *X* that are to be aggregated into a single somehow representative resource, *U* = *RX* and that the vector *Y* is

not to be aggregated; i.e., the aggregation operator S is replaced by the identity operator and $V = Y$. At the outset, we assume the aggregation row vector R is determined from considerations external to the mathematical ones above. Of course, we should have the desired previously mentioned commutativity. This is not possible with arbitrary A and B as it was in the case of bilateral total aggregation, as we shall see.

The relations, $U = RX$ and $V = Y$, the equations (5) with U being a scalar, and

$$\frac{dV}{dt} = -BX = -DU = -DRX$$

imply that every row of B must be proportional to R where the constants of proportionality are the elements of D. We note further that if all the rows of a matrix are proportional to some common row, here R, then also all the columns must be proportional to some common column, here D, the constants of proportionality being the elements of R. That is, the elements of B must be of the form $p_i q_j$.

Conversely, if the elements of B are of this form, then, to preserve the desired commutativity, R must be proportional to a row vector whose elements are q_j, j = 1, ... ,m. Consequently, we have proved the following

Theorem: *Unilateral aggregation of X with no aggregation of Y is possible if, and only if, the elements of B are of the form* $p_i q_j$, $i = 1, \ldots, n$, $j = 1, \ldots, m$, *and then the aggregation vector, R, is proportional to any row of B.*

The attrition operation operator C is then determined from $RA = C$.

Clearly, interchanging X and Y, m and n, and replacing B by A and R by S, we have the same unilateral aggregation theorem for the opponent's resource vector.

Incidentally, we note that the above form for B is not unreasonable from the point of view of modeling. It essentially

postulates an attrition matrix element b_{ij} against Y due to X which is the product of a generalized overall "average" vulnerability factor p_i for the resource Y_i and a generalized overall "average" lethality factor q_j for the resource Y_j.

V. UNILATERAL PARTIAL AGGREGATION

Again for definiteness, we consider aggregation on the resource vector X with the aggregation operator, R, where the dimension r exceeds unity and with no aggregation on Y. Here we have that $U = RX$ and $V = Y$. Consequently, $RA = C$ and $B = DR$ are the sufficient conditions to achieve the desired commutativity and Lanchester consistency. Since D here is of dimension n x r and R is of dimension r x m, it is clear that it is necessary that there be blocks of columns in B whose numbers of columns correspond to the lengths of the component subvectors in the canonical representation of the aggregation operator R shown in (7).

If we assume that a permutation of the indices of the resources is made such that such blocks become blocks of contiguous columns, a block for each aggregation subvector in the canonical form of R, we then see that these blocks in B have the same form $[p_i q_j]$ as B had in the previous section on unilateral total aggregation but over a restricted integer interval of values of j for each block. Then the component subvectors in R become row vectors proportional to the particular row vectors $[q_j]$ that are common to the pertinent block of columns in B. And, the corresponding columns of D become column vectors proportional to the pertinent common column $[p_i]$, their constants of proportionality being the reciprocals of the constants of proportionality in the subvectors of R. Thus R and D are determined up to scale constants that are reciprocally related.

The attrition matrix, C, on the right hand side of the equation for dU/dt for the reduced Lanchester system (5) is obtained directly from $RA = C$. Once again, as in the section IV, we arrive at necessary and sufficient conditions, $RA = C$ and $B =$

DR, with a particular structure for *B*, that permits a unilateral aggregation and either defines *R*, or if *R* is determined exogeneously, defines rows of *B*. *B* cannot be arbitrary. If it is arbitrarily chosen, aggregations with the desired consistency and commutativity properties cannot be achieved.

VI. BILATERAL PARTIAL AGGREGATION

Here we revert to the earlier theorem in section II giving sufficient conditions on the relevant matrices, *A*, *B*, *R*, *S*, *C*, and *D*, that provide the desired commutativity and consistency, that is, of course, when *A* and *B* have the appropriate structure. That is, we require that $RA = CS$ and $SB = DR$.

The arguments are similar to those in the preceding section and will not be carried out in detail. However, there are a few points that are different or that need some minor modification.

We have seen in the preceding section that the prior specification of the lengths of the blocks of resources in *X* that are to be aggregated determines the numbers of columns in B that are proportional to a common column as well as the common column up to a multiplicative constant or, conversely, the appearance of sets of columns in B that are proportional to a common column determine the aggregation subvectors in *R* again up to a multiplicative scale factor.

We also note that if there are sets of columns of *B* or of *A* that are proportional to a common column then *SB* and *RA* respectively have the same respective sets of columns that are proportional to a common column (not the identical columns, but ones that are those columns multiplied on the left by *S* or *R* respectively). Consequently, again *R* will be composed of subvectors which will be proportional to row vectors whose components will be those constants of proportionality in each block in *B* sharing a common column vector. Conversely, if *R* is specified then the column indices of blocks of columns in *B* that have to be proportional to a common column are identified and, up to a multiplicative scalar factor, the row vectors of the

constants of proportionality in these blocks are proportional to the corresponding components in R.

Similarly, corresponding properties of A are determined by a priorly determined S, and conversely, properties of S by a priorly determined A. The attrition matrices C and D are then determined by solving the linear equations $RA = CS$ and $SB = DR$ for the elements of C and D respectively. This should always be possible to do uniquely if the full aggregation possibilities afforded by the blocks of proportional columns in A and in B are effected. If this is not done then we have undetermined systems and degrees of freedom for some choices in C and D exist.

VII. DISAGGREGATION

Obviously, disaggregation to the unaggregated X and Y directly from U and V in the solution (6) of the system of aggregated differential equations (5) without using other information is impossible, any more than one could, for example, determine the value of two numbers from their arithmetic average without another piece of information, such as their difference, another differently weighted average, or some other functional relationship between these two numbers.

In the case of completely general A and B that, of course, still satisfy our original condition of not possessing any null columns or rows, there are two options (whose full description is outside the intent of the current work) that are possible to provide the additional information. They are both based on solving the eigenvalue problems for AB and BA (really only one eigenvalue problem if A and B are square).

Omitting complicating details that can occur when either or both of the matrix products AB and BA are not diagonalizable, we know that the solution (X, Y) of the original system (1) can be written in terms of two linear combinations of exponentials involving square roots of the eigenvalues of AB and BA whose coefficients can be determined from the solution (U, V) of (5) in either of two ways.

The solution (U, V) of the reduced Lanchester system (5) can also be considered in terms of linear combinations of the very same exponentials although some of the coefficients are zero because the system is reduced.

The unknown coefficients for two linear combinations of exponentials are related through two underdetermined systems of linear equations to the now known coefficients in the linear combination coefficient representations of U and V, where the two matrices defining the two systems have the exponentials evaluated at a given time, say, the current. The systems can now be determined by either storing the solution (U, V) of (5) for a succession of values of time sufficient to make the two systems of linear equations in the unknown coefficients determined, or by differentiating U and V a sufficient number of times for the current or any other value of t to enlarge the two systems of equations in the unknown coefficients to become determinate.

Now, for the not so general cases, suppose that X is either totally aggregated or partially so. Now, suppose that, in addition, either all rows of A in the case of total aggregation, are proportional to some common row or to blocks of rows of A coinciding with the blocks of components of X that are being aggregated into a single component of U in the case of partial aggregation.

Then, it is essentially obvious that each component of X in the totally aggregated X case is linearly related to the solution U, the vector of slopes of these linear relations being proportional to the vector of constants of proportionality among the rows of A. The intercepts being determined from the differences between the initial conditions for U and the initial conditions for the particular component of X of immediate concern. The arguments in the case of partial aggregation are identical, except that they are applied individually to each aggregated block of components of X and the associated component of U. Of course, the same arguments are applied to any aggregation, total or

partial, of Y provided that all or else the relevant blocks of rows of B are proportional to common rows respectively.

Submitted for publication to *Naval Research Logistics*

EXPERIMENTS IN VARIABLE RESOLUTION COMBAT MODELING[1]

RICHARD HILLESTAD,[2] JOHN OWEN,[3] AND DON BLUMENTHAL[4]

ABSTRACT

This paper examines the differences in combat outcomes predicted by models of different resolution applied to identical combat situations. First, hypothetical combat situations are posed, then several models of varying degrees of resolution in the spatial representation, aggregation of forces, and time step are used to predict losses and battle winners. Both stochastic and deterministic simulations are used. The comparisons of outcomes provides important insights into the problems of aggregation. Some observations from this set of experiments are that intuition regarding outcomes, causes and effects are frequently wrong, leading to bad approximations in the aggregate. Scaling for different levels of resolution is possible but a method of predicting the appropriate scaling technique and factors in advance has not been found. The differences in outcomes between stochastic and deterministic models are most pronounced in the "fair fight" regime in which the force balance (accounting for situational factors) is nearly even. Because defense analysis frequently operates in this regime (getting "just enough" force to a theater or because constrained defense budget allocations may not permit overwhelming odds) this result implies that great care should be taken to understand the possible variance in outcomes.

---

[1]Part of this research was performed for DARPA on the project, *Military Science and Advanced Simulation* under contract number MDA903-90-C-0004

[2]RAND, Santa Monica, California.

[3]British Defence Operational Analysis Establishment, UK.

[4]Lawrence Livermore National Laboratory, Livermore, CA.

1. INTRODUCTION

Aggregate, low resolution combat models are needed for a number of reasons. The fact that much historical data is of an aggregate nature without the details to totally reconstruct battles means that models based on such data must either operate at an aggregate level of resolution consistent with the data or the modeler/analyst/gamer is forced to make up interactions such as fire allocation, detailed acquisition predictions, small unit objectives, etc. Command and control decisions frequently require information at an aggregate level in which the commander or his intelligence branch estimates the "strength" of opposing and friendly forces to decide on the commitment of reserves, when to move, etc. Aggregation is often necessary in analysis to comprehend and explain phenomena (the forest vs. trees argument). Other issues such as cost, the need to produce results to meet deadlines and repeatability also force the analyst away from detailed weapon on weapon analysis. At the same time, these aggregations should not be done arbitrarily. Unfortunately there is very little theory and science in most approaches that have been taken to aggregation in combat modeling.[5] In a companion paper[6] one of us has examined some of the theoretical issues involved in aggregation and disaggregation. In this paper we look at the problem more empirically, comparing the results of simulations at various levels of aggregation.

The approach used was to perform a controlled set of simulation experiments on the same combat problem using simulations with differing degrees of resolution. The simulation results reported here are derived from two basic models. The first model was a detailed weapon on weapon event stepped, stochastic model which simulated each individual firing decision and round fired. The second model was a time stepped,

---

[5]Davis, Paul K. and Donald Blumenthal, *The Base of Sand Problem: a White Paper on the State of Military Combat Modeling*, RAND, N-3148, May 1991.

[6]Hillestad, Richard J., and Mario L. Juncosa, *Cutting Some Trees to See the Forest: On Aggregation and Disaggregation in Combat Models*

deterministic spreadsheet with fighting units aggregated into company, battalion, and regiment groups. Using the two models we compared results when changing the spatial resolution, configuration[7], and object (unit) resolution. We also examined how the results differed when simulated deterministically from those obtained with the stochastic model. The following sections describe the experimental frame, the models used, the results, some possible approaches to developing consistent aggregations, and some general observations drawn from the research.

2. DETAILED STOCHASTIC SIMULATION

The first model used is detailed (high resolution) in that it represents the individual systems and shots of those systems. On the other hand, it was used simply in the analysis in that it did not consider acquisition and movement was constrained to straight paths on flat terrain with no obstacles. Each individual tank or other vehicle is described in terms of its side, type, position, speed and direction of movement, number of rounds remaining, and state - alive or dead. Systems are configured in groups, each of which moves along a straight path until an objective line is reached.

The ground is assumed to be flat and free of obstacles; all other vehicles can always be seen, with no delay for searching, but a dead vehicle may not be recognized as such until a given time after it is killed, or until a given number of hits have been scored on it. The perceived state of a vehicle is the same for all observers.

In the model each system always fires at the closest enemy in range which is perceived to be alive at the moment of firing. Each system type has a rate of fire; the time between shots is not affected by the need to switch targets. Furthermore, effectiveness and accuracy are not affected by the motion of the

---

[7] See Horrigan, Timothy, *Configurational Theory and the Mathematical Modeling of combat: Realizing the Potential of Models and Simulations of Combat to Improve the Effectiveness of Weapons, Tactics, and Training*, Horrigan Analytics, HAS 91-17-1 (1991), for other discussion of the problems related to configuration and aggregation.

firer or target. Probability of kill is range-dependent, as is time of flight. Projectiles are assumed to be unguided after launch, or fire-and-forget; the projectile time of flight is not added to the time to the firer's next shot. When the projectile reaches the target, a random number is compared with the single-shot kill probability (Pk) to determine whether a kill has occurred; any delay in perceiving a kill is measured from this time.

Because of the stochastic nature of the model, the battle is repeated a number of times. There are two types of output - statistical, showing the cumulative results over all replications, and graphical, showing the movements and fate of systems in one selected replication.

The model is implemented in the MODSIM object-oriented simulation language[8].

3. THE SCENARIO AND RESULTS OF DETAILED SIMULATION

Figure 1 shows the start of the main scenario used in this analysis; the grid lines are at a 500 meters spacing in both axes, but in order to fit the scenario on the display, different scale factors have been used for the x and y directions. The 33 tanks of the three defending companies are shown individually. The attacking tanks (99 in number) are also individually marked, but because they are closer together, the symbols overlap at this scale. The attackers are organized in three battalions, two forward and one back, each composed of three companies, two forward and one back. Figure 2 shows the simplified probability of kill curves used in the scenario. Comparing this figure with the initial positions in Fig. 1, it is seen that all companies are initially out of range of each other. The attacker will move all companies forward in the formation simultaneously at a constant speed. In the simulation, as systems are killed they stop on the battlefield. When they have taken a certain number of shots, or

[8]Anyone wishing to know more about this model should contact R. Hillestad or J. Owen at RAND.

after a specified time has passed, they are perceived by all other systems on the battlefield to be dead. Up until that time they can draw fire. In the initial cases we have set the perception delay time to zero so that a system is instantaneously perceived to be killed when it happens. Later results will show the result of this perception delay.

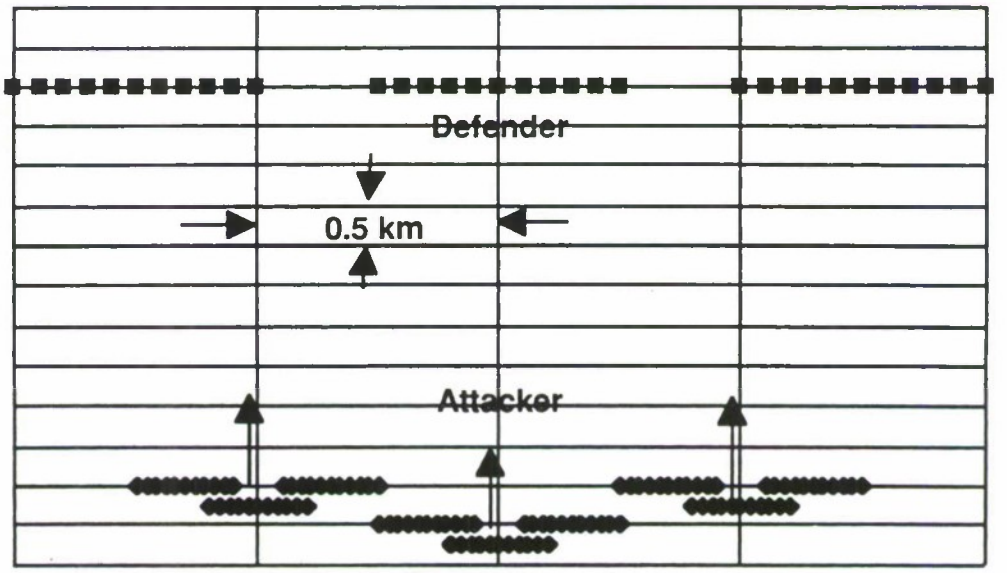


Fig. 1—Initial Positions in Scenario

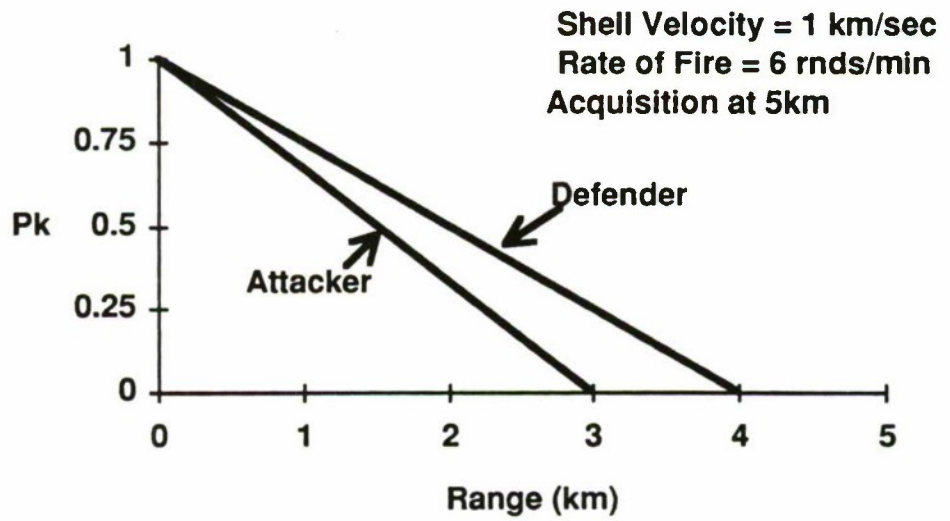
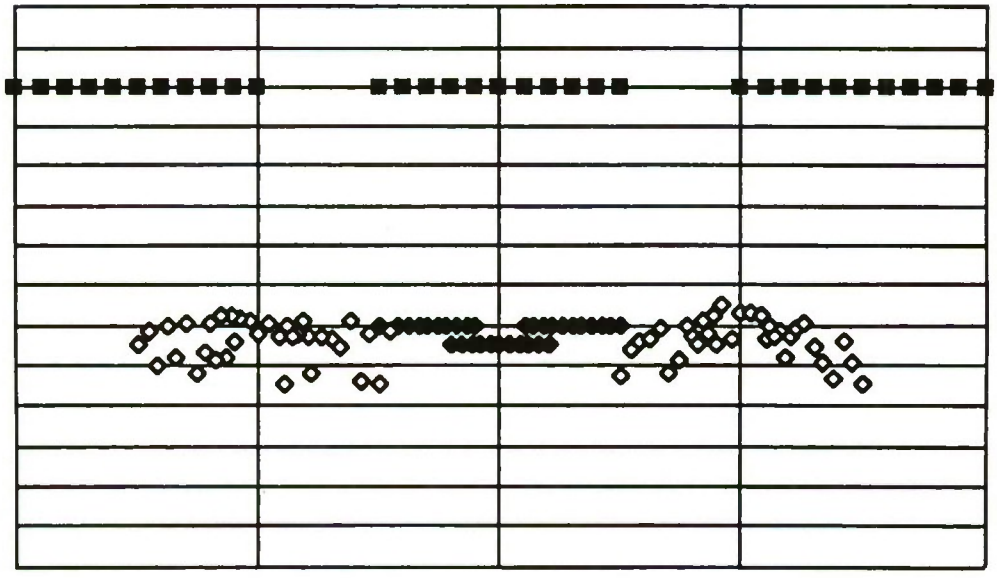


Fig. 2—Simplified Probability of Kill Curves

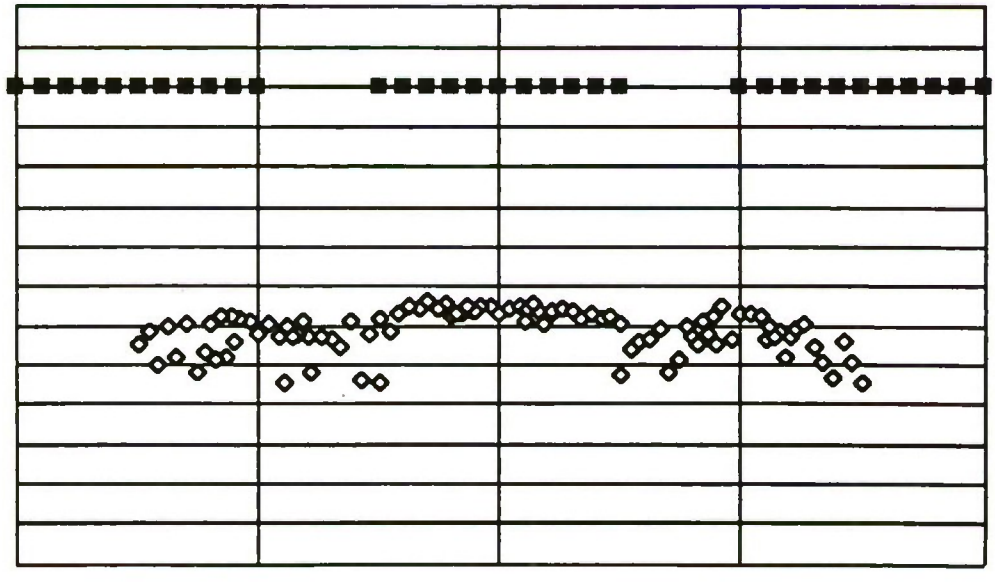
This scenario gives a 3 to 1 numerical advantage to the attacker and a range/Pk advantage to the defender. The 3:1 rule[9] suggests that the outcome in this battle should favor neither the attacker nor the defender. As we will show, this is far from true. Figure 3 shows snapshots of the battle at two later stages in one of 30 replications run. The attacker is closing on the defender's position, in this case at 30 kilometers per hour (km/hr). The defender has a 1 kilometer range advantage, and is using it. The left diagram shows that the attacker's forward companies have already been decimated - the hollow symbols are dead attackers - while the attacker is not yet in range to retaliate. The second diagram shows the end of this battle. The attacker has been wiped out at no loss to the defender - indeed the attacker barely manages to get into range to fire a few shots.

[9]See the debate about this rule in Mearsheimer, John J., *Assessing the Conventional Balance: The 3:1 Rule and Its Critics*, International Security, Spring 1989, and in Epstein, Joshua M., The 3:1 Rule, the Adaptive Dynamic Model, and the Future of Security Studies, International Security, Spring 1989.

This outcome is explained by the curve in Fig. 4 which has been derived from the Pk versus range curves in Fig. 2. In the region from 3 to 4 kilometers the defender can shoot at the attacker with some effectiveness but the attacker cannot return fire with any effectiveness. At less than 3 kilometers the ratio of Pk advantage decreasingly favors the defender. The attacker does have a numerical advantage so that once in range the advantage quickly switches to the attacker. The problem for the attacker is to cross the "gauntlet" of defender fire and get into his effective fire zone before the defender defeats him. The problem for the defender is to destroy enough of the attacker before he can get into range and use his numerical advantage. In the case shown, the attacker crosses the zone of infinite defender advantage too slowly so that all systems are destroyed before they get into range. It is relatively easy to calculate this. At 30 km/hr it takes an attacker system 2 minutes to cross the zone between 3 and 4 kilometers. The attacker battalions are spread 1 km in depth as well so that it takes about 4 minutes for all attacker systems to cross the zone. In 4 minutes the defender can fire 33(Systems)*6(Shots Per Minute)*4(Minutes) = 792 shots. The average Pk in the zone is 0.125 and the expected number of kills as the attacker crosses the zone is 0.125(kills/shot)*792(shots) = 99 kills.



State After 5 Minutes



State After 7 Minutes - End of Battle

Fig. 3—Battle Results at a 30 km/hr Closing Speed of Attacker

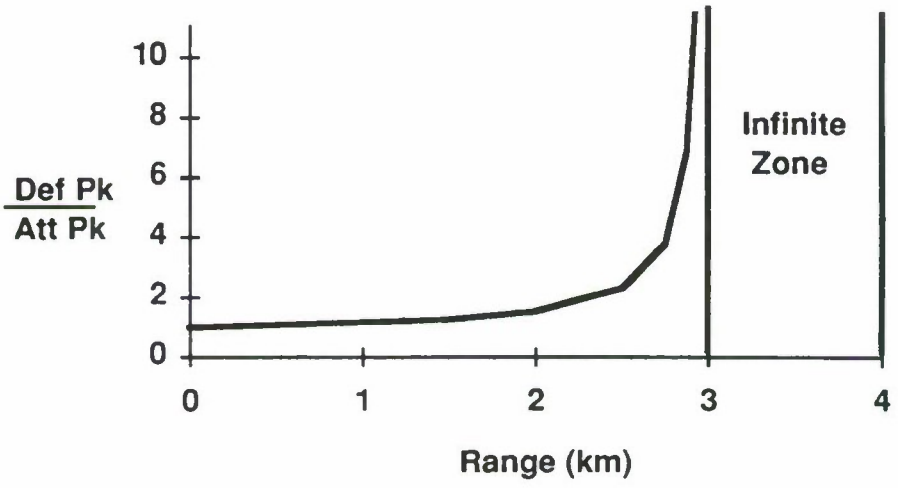
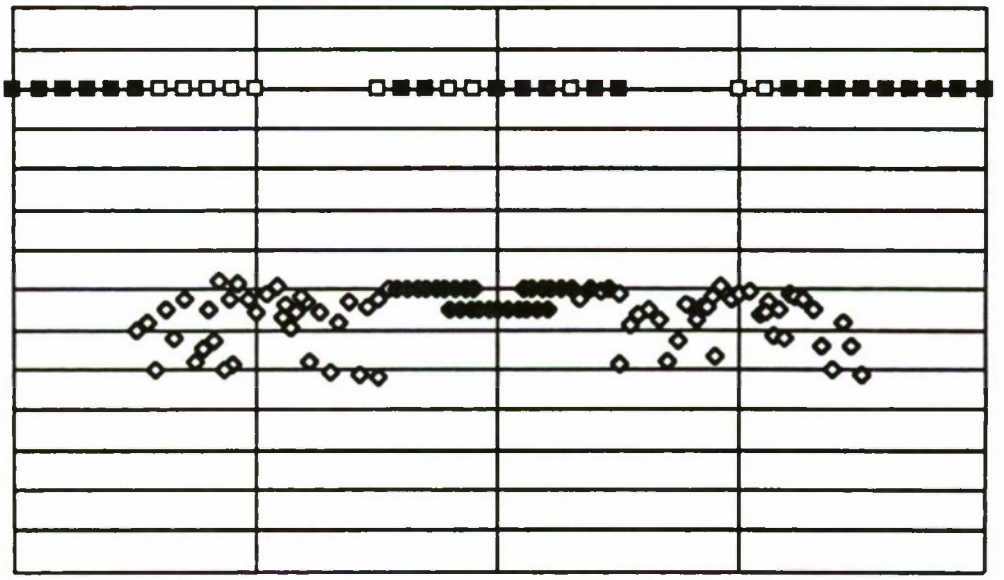
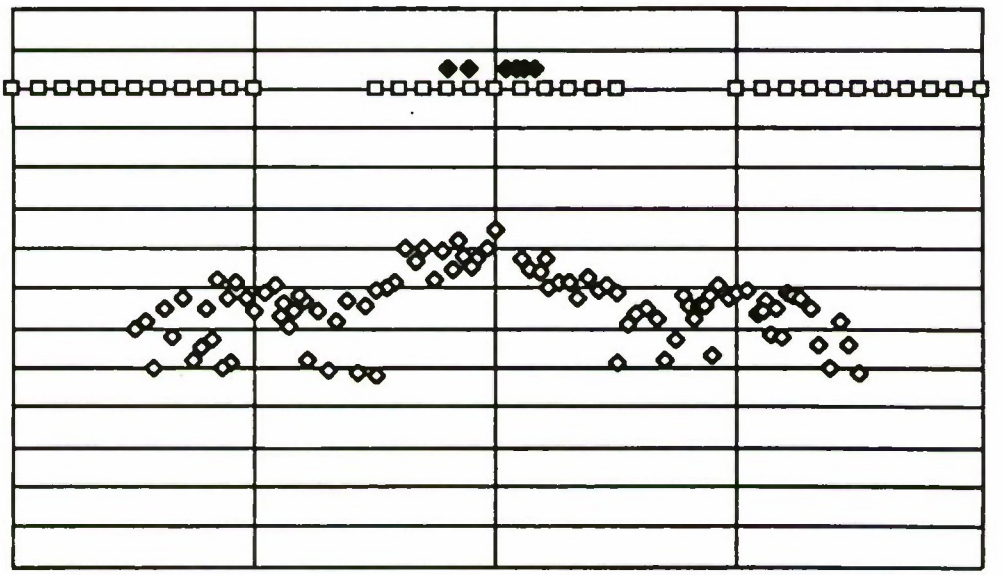


Fig. 4—Defender/Attacker Pk Ratio as a Function of Range

One option for the attacker is to get across the disadvantageous zone faster. Figure 5 shows a run with the same initial deployments but with a greater attacker speed - 45 kilometers per hour. As stated earlier, the greater speed has no effect on either side's gunnery performance. At the intermediate stage, the forward attacker battalions have been wiped out, but not before getting far enough to do some damage to the defender. By the end of the battle, in this specific replication, the defender has been wiped out, and a few surviving attackers are past the defender's position. The higher speed has allowed the attacker to cross the zone of defender advantage fast enough to win the battle, although it is somewhat of a Pyrrhic victory. We have not considered the implications such large losses would have for either side's willingness to continue in the battle.



State After 4 Minutes



State After 8 Minutes; Defender Wiped Out by 6 Minutes

Fig. 5—Battle Results at a 45 km/hr Closing Speed of Attacker

To further illustrate the point about the attacker needing to get across the lethal zone we varied the initial attacker configuration. Figure 6 shows an attacker deployment with all three battalions in line. The attack is therefore more concentrated in an attempt to get more weapons into range faster. The intermediate snapshot in Fig. 7 shows that if this formation advances at 45 km/hr, the lead companies of each battalion suffer heavily, but the defender also takes losses. The attacker wipes out the defense in this run, with a greater number of survivors than with a two-up one back formation at the same speed.

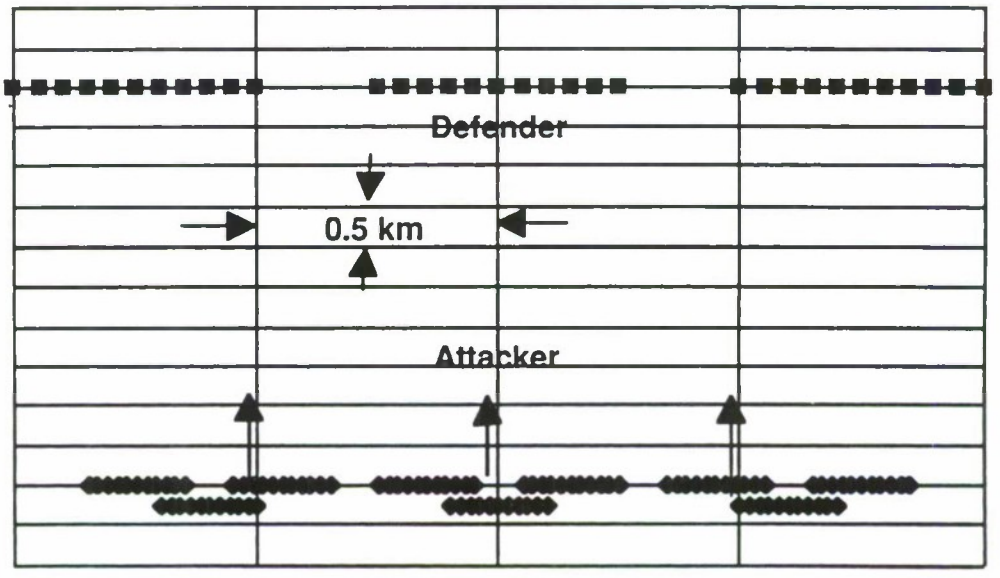
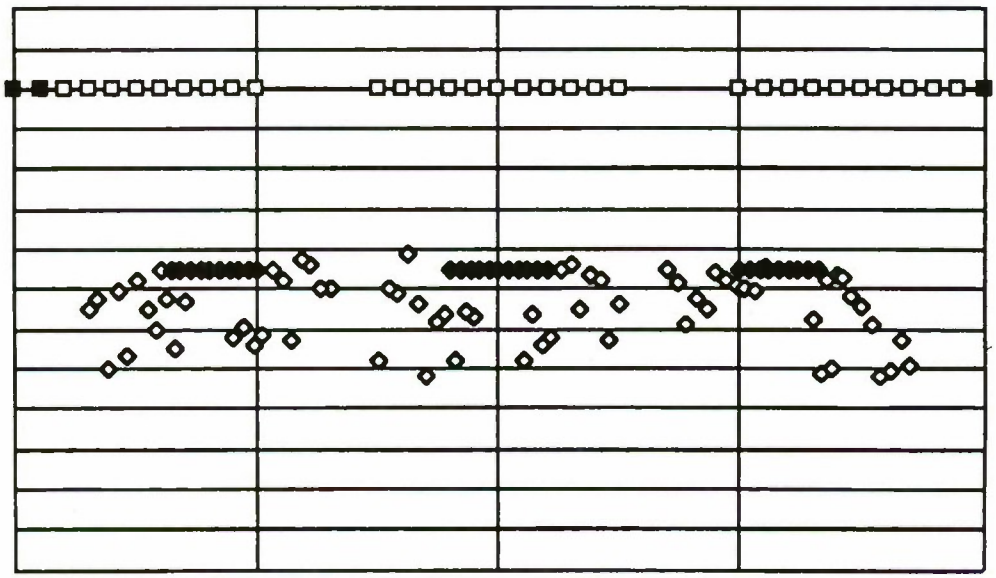
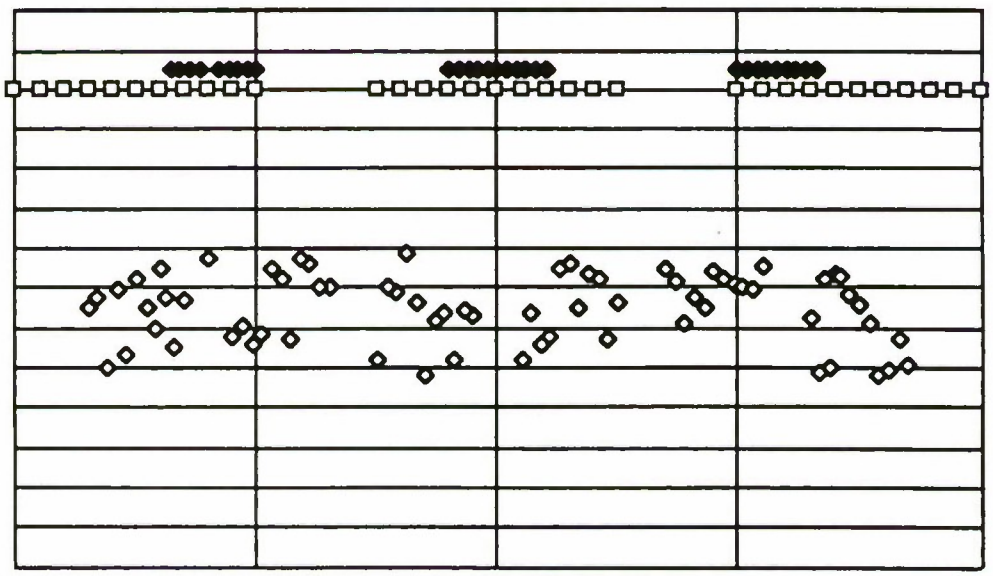


Fig. 6—Initial Position: All Attacker Battalions Forward



State After 4 Minutes



State After 8 Minutes; Defender Wiped Out by 5 Minutes

Fig. 7—Battle Results at 45 km/hr and All Attacker Battalions Forward

To test the model for any bias in structure which might favor one side or the other, we also created a meeting engagement scenario - a completely symmetrical battle. There is no range advantage to either side; both use the "attacker" Pk curve of Fig. 2. Figure 8 shows the initial positions in the battle; both sides move forward to attack the other from these positions. In Fig. 9 the intermediate snapshot shows each side's forward companies almost wiped out, and by the end of the battle, in this particular case, by luck of the dice, BLUE is left with six survivors out of ninety-nine. We won't be showing any more results for this scenario, but it has been run for 30 replications, which resulted in 14 BLUE wins and 16 RED wins - a win being defined as having at least one survivor at the end of the battle. Other statistics gathered from these replications indicates the model does not have any apparent biases. The number of survivors on the winning side ranged from 3 to 37. We will discuss the implications of such a large variance in survivors later.

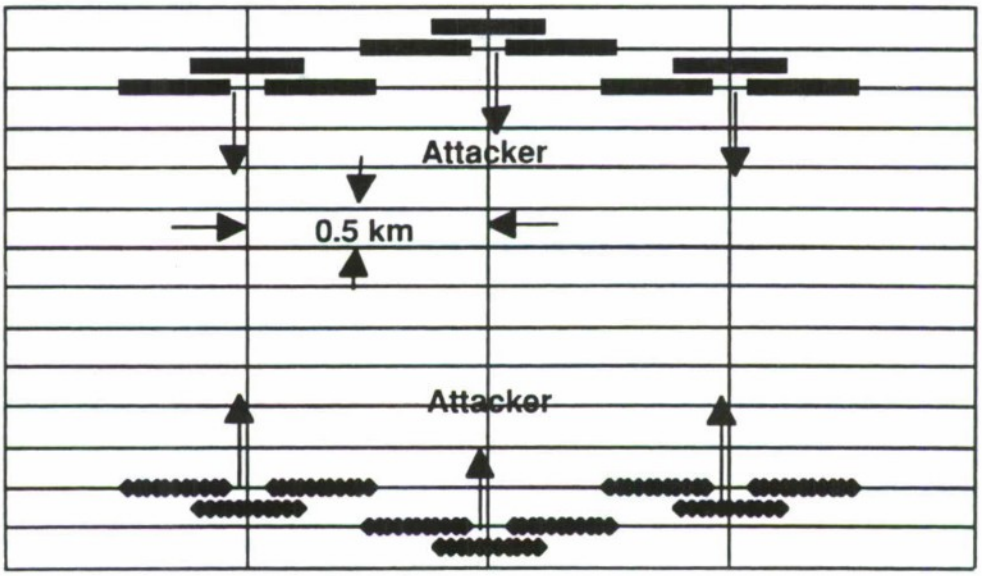
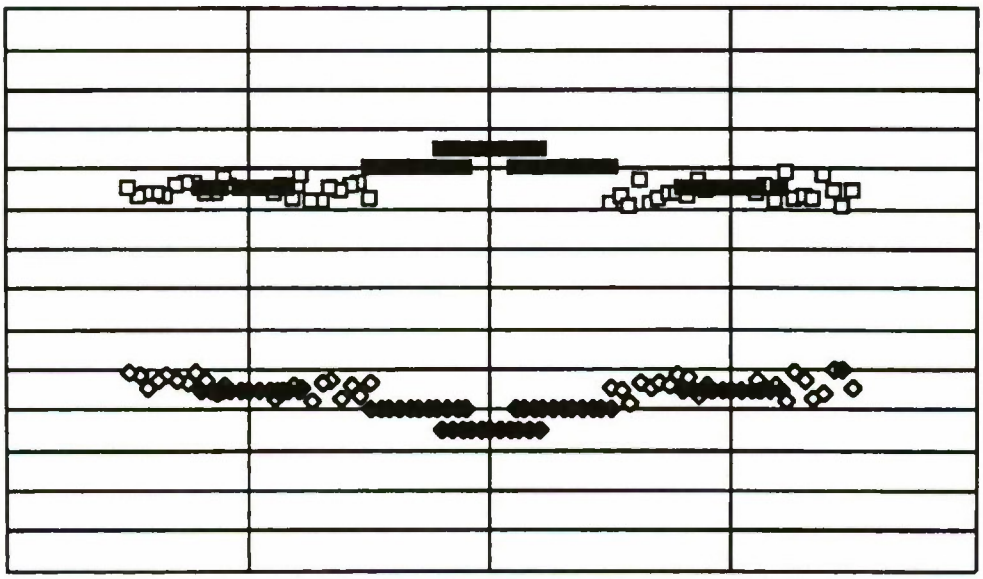
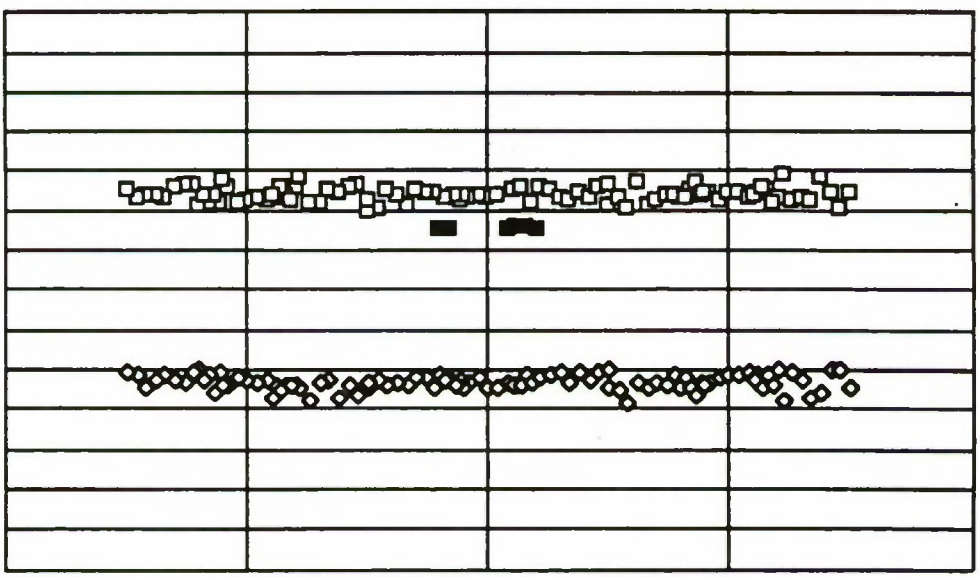


Fig. 8—Initial Positions in a Meeting Engagement



State After 6 Minutes



State After 10 Minutes; One Side Wiped Out by 9 Minutes

Fig. 9—Battle Results for a Meeting Engagement

The previous battlefield graphic displays showed single replications; the positions in those replications are dependent on the particular values drawn for random numbers. Figure 10 summarizes some statistical results for the initial scenario. The quantities plotted here and in the figures that follow are the number of defender and attacker tanks surviving at the end of the battle, where the battle is always fought to the annihilation of one side or the other. The results are for 33 defending tanks against 99 attacking tanks. All cases were run for thirty replications. The shaded part of the column shows one standard deviation about the mean. Above each column is shown the number of replications (out of 30) won by defender and attacker, where "winning" means having at least one tank left. Many of these victories are in fact Pyrrhic.

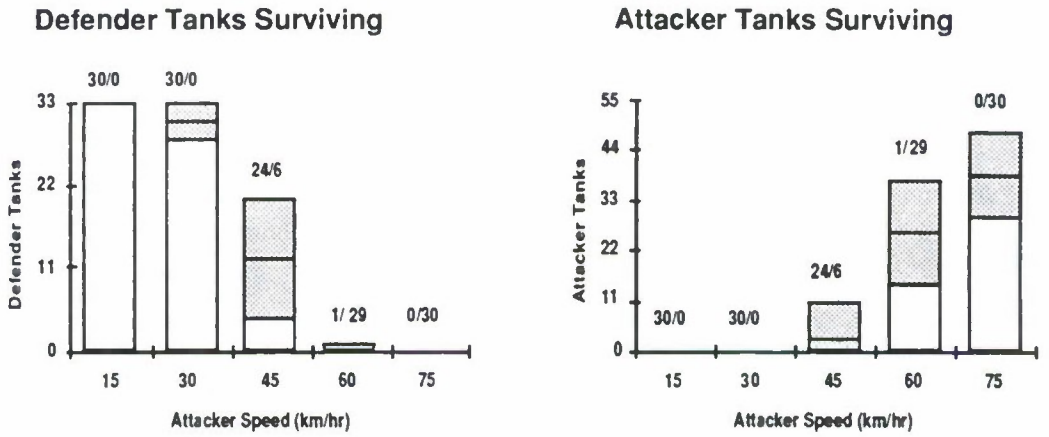


Fig. 10—Stochastic Model Results: Effect of Attack Speed

In the figure the effects of changing the speed with which the attacker closes on the defender's position are shown. At 15 km/hr the defender wins in all replications, without losing a

single tank; the attacker is not moving fast enough to cross the zone where the defender can fire but he cannot. At 30 km/hr, the defender always wins, but the attacker occasionally gets close enough to do a little damage. At 45 km/hr, there is a wide variability in the number of survivors. The defender wins 24 out of 30 replications, but takes heavy losses. Where the attacker wins, he has few survivors. At higher speeds, the advantage swings decisively to the attacker, though he always takes substantial losses. One observation is that the variance is largest when the fight is "fair" or nearly equal. As should be expected, when one side or the other has a predominance of force, the variance in outcome is relatively small. This type of result has been reported elsewhere.[10]

All the results thus far have assumed perfect perception - when a tank was destroyed, this was immediately known by all enemy tanks. Figure 11 shows the effects of varying perception. All results are for a speed of advance of 45 km/hr.

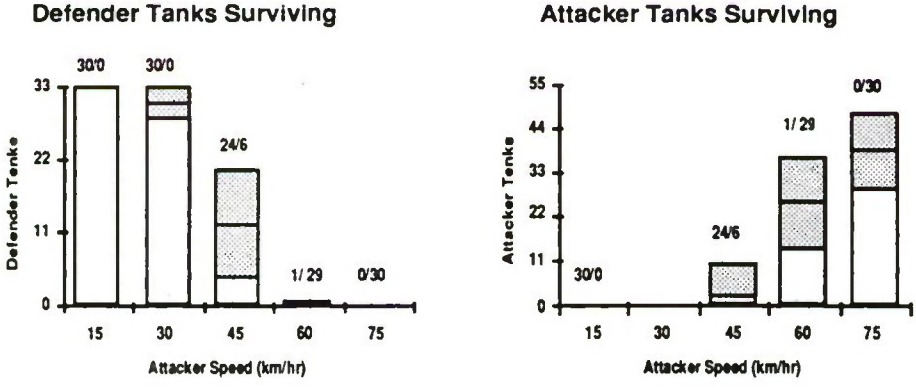


Fig. 11—Stochastic Results: Effect of Perception

[10]Hans W. Hofmann, *On an Approach to Stochastic Modeling of Combat at the Corps/Army Level*, Paper presented at the TIMS/ORSA/MAS Meeting, Nashville, May 1991.

The first case is with perfect perception, as in the previous figure. In the second case, there is a 10 second delay following a kill of a tank on either side, before the enemy realizes that it is dead. Shots will therefore be wasted on dead targets. This favors the attacker because in the early stages of battle many of the forward attackers are killed, and they draw fire away from those behind or beside them, allowing more attackers to move into firing range. Note that the attacker/defender "win" ratio changes around dramatically. The defender cannot afford to waste these shots as the attacker moves across the zone where the defender has an advantage.

The third case delays perception of kill not for a fixed time but until three hits have been recorded on a target. This also favors the attacker, but not by as much as the 10 second delay; several rounds can hit a target in less than 10 seconds.

The final case shown imposes a 10 second delay on attacker perception of defender death, but no delay on defender perception of attacker death. This case, which may be more realistic, favors the defense.

Figure 12 summarizes the effect of changing the attacker's formation. All results assume perfect perception and a speed of advance of 45 km/hr. The first case shown is for the two-up one-back formation, as in previous figures. The second case has three battalions in line, but the companies within each battalion are in a two-up, one-back formation, as illustrated in Fig. 6. As can be seen, this more concentrated attack favors the attacker, who wins in the majority of replications. The defender is less able to cope with one wave of attackers before the arrival of the next and the attacker gets more of his systems across the gauntlet and into firing range. The third case puts all nine attacker companies in line. As might be expected this is even more favorable to the attacker.

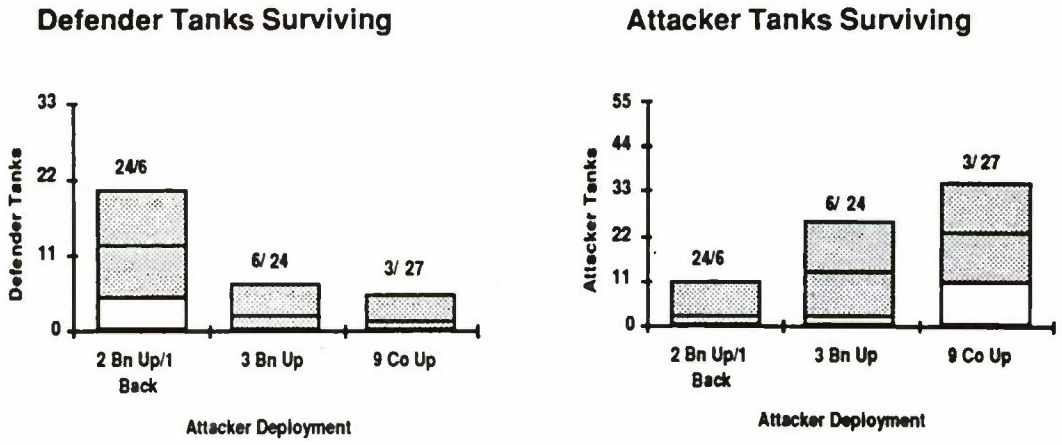


Fig. 12—Stochastic Results: Effect of Deployment

4. AGGREGATE SIMULATION RESULTS

At this point it is interesting to start comparing results with a more aggregate model. This model is a deterministic time-stepping simulation which calculates attrition rates between groups of weapon systems on each side.

The model fights the battle through a succession of regular time steps. In each time step, group positions are updated, and hence the ranges between them and the probabilities of kill change.

Each group directs all its fire at the closest living enemy group within range. There is no acquisition problem; all enemy groups are always visible, and perception is also perfect; dead groups cannot be mistaken for live ones.

In each time step, the kills by a group of its target group are calculated as the product of group rate of fire, the duration

of the time step, the strength of the firing group at the start of the time step, and the probability of kill at the range between the groups. If these calculations result in a group taking losses in a single time step exceeding its strength, the shots fired and kills obtained by each enemy group firing at that group are reduced proportionately so that the total number killed is equal to the number of targets. Other than this, there is no rounding; group strengths during the battle will generally not be integer. The calculation takes no account of time of flight of projectiles, and assumes perfect distribution of all hits within a time step. In other words, all hits are on distinct targets within the target group - there is no overkill.

The model is implemented on the Microsoft Excel spreadsheet[11].

The size of groups can be varied in the model. The original scenario has been represented, and results will be shown at three levels of aggregation: company size groups, battalion size groups, and battalion versus regiment. Figure 13 depicts these different levels of resolution.

---

[11]Anyone wishing to know more about this model should contact R. Hillestad or J. Owen at RAND.

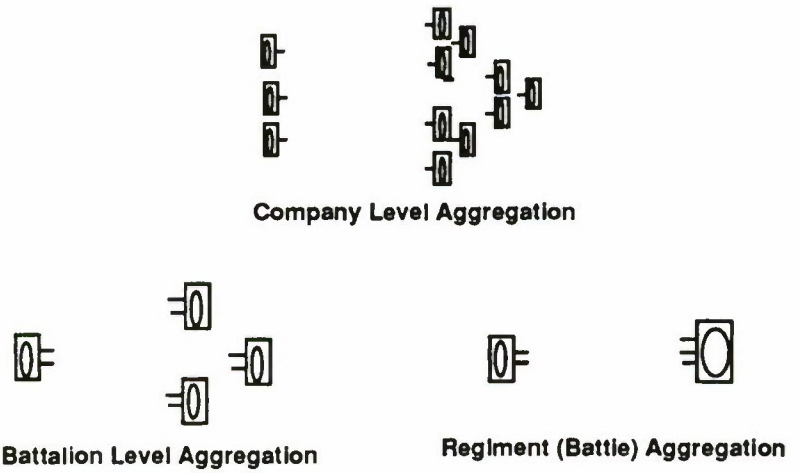


Fig. 13—Levels of Resolution in the Aggregate, Deterministic Model

Figure 14 shows the effects of attacker speed of advance in the deterministic model with company size groups (3 defender groups versus 9 attacker groups, as above). Also shown are the results obtained from the equivalent runs of the stochastic simulation. The label above the columns shows the winner in the deterministic model and the number of replications(out of 30 total) won by the defender/attacker in the stochastic model. The bar heights represent the mean number of survivors in the stochastic model. At low speeds of advance, the models are in good agreement - the defender wins with little or no loss. At higher speeds, the deterministic simulation strongly favors the defender in comparison to the stochastic model. Why is this?

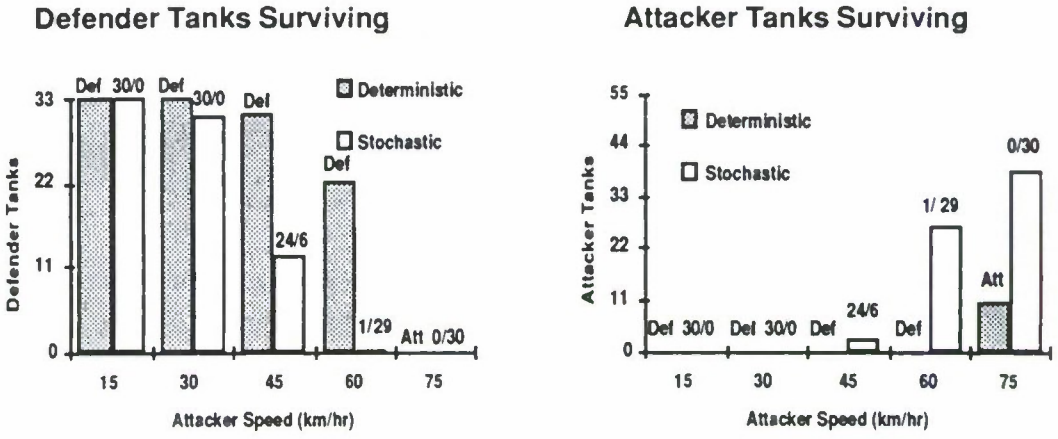


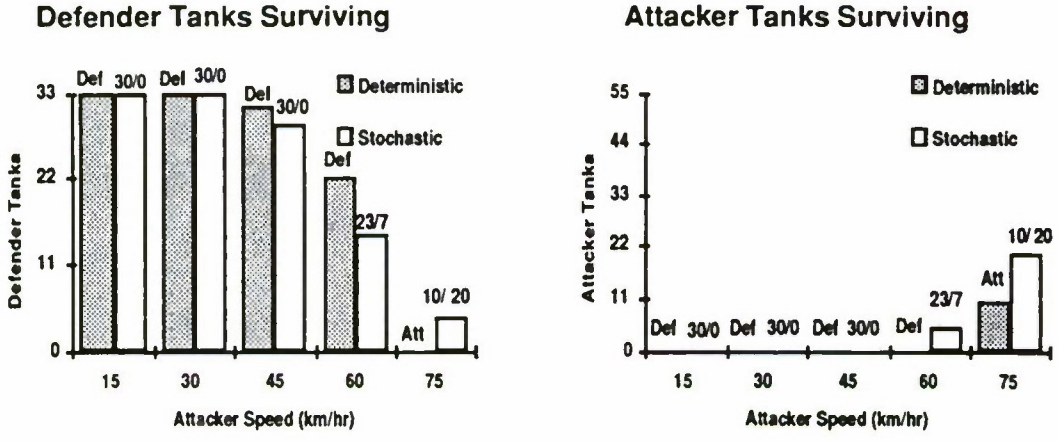
Fig. 14—Stochastic and Deterministic Models: Effect of Attack Speed

In describing the deterministic model, we noted that it did not allow for projectile time of flight. In the stochastic model, time of flight is dependent on range and projectile velocity but at the ranges where the attacker suffers most losses it is typically about three seconds. During this time, even with perfect perception of tank death, shots may be wasted firing at targets which are about to be killed by shells already in flight. The deterministic model assumes that there are no multiple hits on the same target within a time step.

To test whether this difference in representation accounted for the difference in results, runs of the stochastic model were carried out in which the speed of projectiles was increased to a point such that time of flight was essentially zero.

Figure 15 shows the deterministic model results as before, but stochastic model results with zero time of flight of shells.

These are far closer to the results of the deterministic model. This one difference in modeling led to most of the divergence in results between these two models in the cases examined. In constructing the aggregated model, an assumption was made that the time of flight could be left out of consideration without a significant impact on results (at least in the perfect perception cases) - a presumption which turned out to be incorrect. This illustrates the need to consider carefully and to test all the simplifications carried out in aggregation.



Labels above columns shows Winner in Deterministic Model, and replications won by Defender/Attacker in Stochastic Model Mean survivors are shown for Stochastic Model All Stochastic Model results are for runs of 30 replications

Fig. 15—Stochastic and Deterministic Model Results: Zero Projectile Time of Flight

Setting the time of flight of shells to zero has brought the more detailed model's results closer to those of the more aggregated one. However, this is the "wrong way round." The question is, how to take time of flight, and the consequent effect on allocation of fire, into account in the aggregated model. One approach is to determine an "effective rate of fire" for the deterministic model which matches that of the stochastic model.

Figure 16 shows the results of varying the rate of fire in the deterministic, company level model; the stochastic model mean result is shown as the rightmost bar. Note that an effective rate of fire of 4 shots per minute in the deterministic model most closely matches the results of the stochastic model in this case. This is an empirical result; there is no guarantee that it applies at other attack speeds, let alone more widely. We have not yet attempted to predict the result by side calculations.

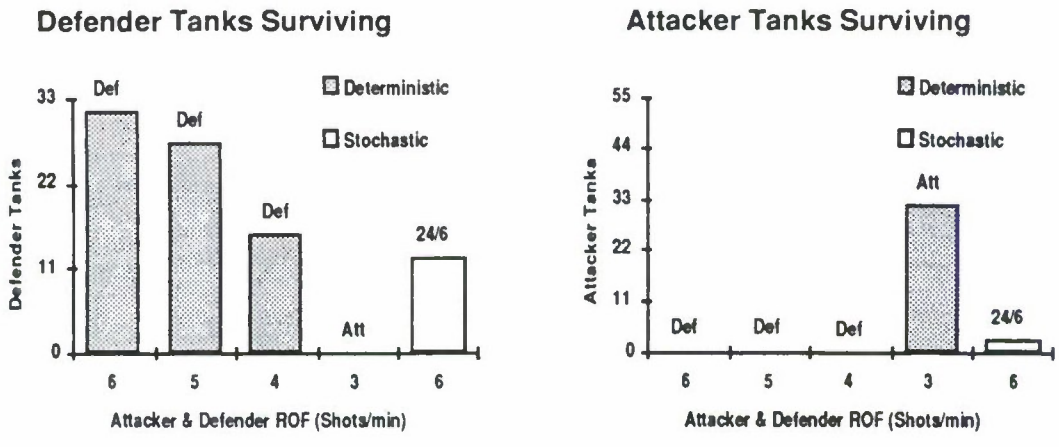


Fig. 16—Comparison of Models: Varying the Effective Rate of Fire in the Deterministic Model

The time step in a deterministic simulation is a form of aggregation in that the results of all processes that go on during a time step are computed at a single point in time using assumptions about the constancy of rates of movement, firing rate, etc. If the time step is too large, the state of resources may not change as fast as they should. That is, if we assume a certain number of weapons at the beginning of the step and compute the loss rate to the other side from that number of weapons, the

loss may be exaggerated because some of the killing systems would have been destroyed themselves during the time step. Figure 17 plots the number of defender tanks surviving in the original scenario at the end of runs with different speeds of attacker advance. All results are for a representation with company sized groups, hence nine defender groups against three attacker groups.

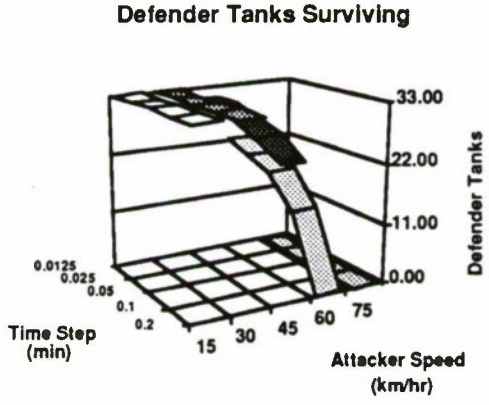


Fig. 17—Deterministic Simulation: Effect of Time Step and Attack Speed

The third axis in Fig. 17 shows the length of time step used in the calculations; there are 25 cases shown here, all the combinations of five different time step durations and five different speeds of advance. The results shown in the previous figures which compared the deterministic and stochastic model results are those with the shortest time step value.

Increasing speed favors the attacker, as already noted. With the extreme speed values, the length of time step has no effect on the number of survivors. At the intermediate speeds, however,

changing the time step changes the battle outcome dramatically. Long time steps favor the attacker. Pks in a time step are calculated using the ranges at the end of the time step. The shorter the range, the less the defender's advantage in lethality. With long time steps, the amount of time for which the attacker is unable to fire is reduced.

This simply illustrates the importance of time step length in this type of model. If this model were being used as the direct fire attrition calculator in a corps or theater model, it might be convenient to have long time steps in order to keep run time down; this could be dangerous. All subsequent results shown of this model will be those obtained with the shortest time step value used here.

Consider next the size of the group represented in the deterministic model. What differences arise from using battalion or regiment sized groups as we make the model have even less (lower) resolution?

Figure 18 shows the number of defender and attacker tanks surviving at the end of the battle. Results for different speeds of advance are shown. The third axis varies the level of aggregation used in the representation. The first set of results use company size groups, as in previous figures. The second set pits 3 battalion sized attacker groups against a single battalion group of defenders. The third set has a single regiment or brigade attacking group advancing on a battalion group.

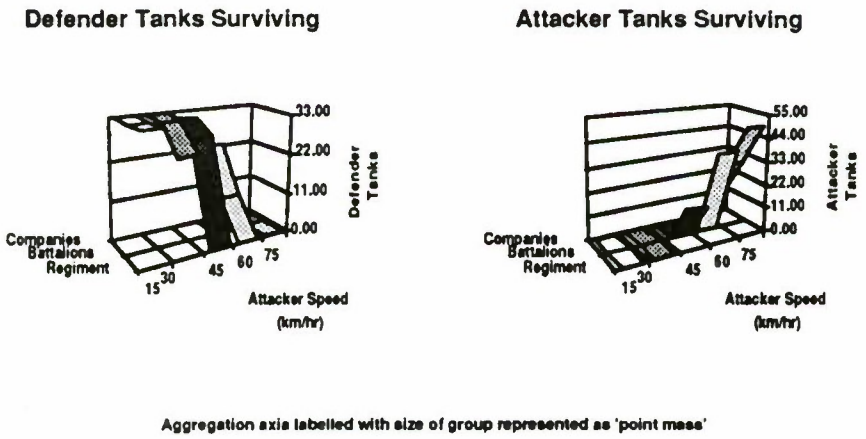


Fig. 18—Aggregated, Deterministic Model: Effects of Level of Aggregation

The level of aggregation has no effect at low speeds - the attacker is unable to get into range with enough survivors to fire effectively, no matter what the level of aggregation. At higher speeds, the larger group sizes favor the attacker.

With company groups, the defender first engages the forward companies of the forward battalions - 4 in all or 4/9 of the attacker's strength, then the rear companies of these battalions - 2 companies, then the 2 forward companies of the third battalion, and finally the third company of this battalion. It is also in this order that the attackers, if they survive, come into range to fire back.

With battalion groups, the forward 2 battalions, 2/3 of the attacker strength, come into range at the same time. Where there is a single regimental group, the whole of the attacker's force comes into range at once. The effect of the greater aggregation is similar to that of putting more attacker companies into the

front line. As shown in the stochastic model, the attacker does better with more units forward because he does not feed the units into the battle piecemeal and give the defender the advantage of shooting at only a few forward units at a time. However, the aggregation to large units implies that the entire larger unit comes into range at once and this is an artifact of the aggregation, not an explicit assumption of the aggregate model. Suppose it is desirable to represent a different configuration of the sub units in an aggregation. Is there a way to represent a different forward posture? Figure 19 shows the effects of attempting to represent the attacker's formation within a battalion size group by restricting the proportion of the group allowed to fire. Results are compared with those for company groups when the company groups are placed in the original 2 up, 1 back configuration. The graphs show defender and attacker survivors, at different speeds of advance. Results with company groups and battalion groups are shown as in the previous figure. The third and fourth sets of results also use battalion groups, but an extra factor was added to the calculation of attrition rates; a multiplying factor on the number of firers. This factor was left at 100% for the defender, but was set to 75% and 67% for the attacker's battalions, in the two sets of cases.

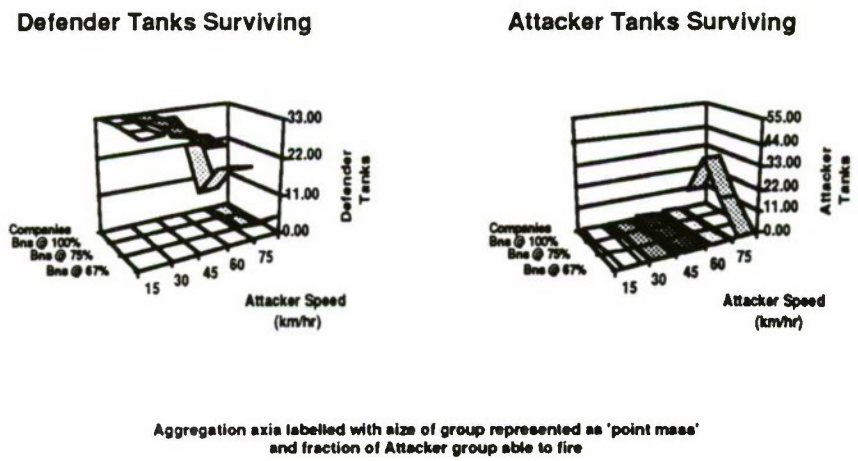


Fig. 19—Aggregated Deterministic Simulation: Effect of Fraction Participating

This factor has no effect at low speeds, where the attacker gets little or no chance to fire. At higher speeds, an effect is apparent. Reducing the proportion of attacker allowed to fire favors the defenders, bringing the results back towards those with company groups. Indeed, if only 2/3 of the attackers can fire, the battle becomes more favorable to the defender than with company groups. It can be seen that with this level of attacker participation, at the highest speed of advance used, the defender wins in that he has a few survivors at the end of the battle, and the attacker does not. A 75% level of participation seems to give a better approximation to the company level results in the cases shown.

Thus, configuration can be represented when greater aggregation is used, but the modeling has to be adjusted to do so. In this case an additional factor had to be added to the low resolution, aggregate model. The value of this factor needs to be

determined somehow and it is probably quite situation dependent. In our case it was done empirically but it is possible that it might be done predictively by careful consideration of the rates of advance, firing rates, battle configuration, etc. We have not attempted to do this but we note that others have not had a considerable degree of success.[12]

5. A CONSTANT COEFFICIENT LANCHESTER MODEL[13]

We also considered how well the battle described by the original scenario might be represented by a constant coefficient (no time variation) square law[14], Lanchester model. For this basic case the familiar equations are

$$\frac{dx(t)}{dt} = -Ay(t)$$
$$\frac{dy(t)}{dt} = -Bx(t).$$

where x(t) and y(t) are the strengths of the two sides at time t, and A and B are the (constant) rates at which one unit of strength on one side causes attrition of the other side's strength. The well known, closed form solution for these equations is given in terms of hyperbolic functions as

$$x(t) = x_0 \cosh\left(\sqrt{AB}t\right) - y_0\sqrt{A/B}\sinh\left(\sqrt{AB}t\right)$$
$$y(t) = y_0 \cosh\left(\sqrt{AB}t\right) - x_0\sqrt{B/A}\sinh\left(\sqrt{AB}t\right)$$

where

$$x_0 = x(0)$$
$$y_0 = y(0).$$

---

[12]For example, see Thomas Schaub, *Zur Aggregation heterogener Abnutzungsprozesse in Gefechtssimulationsmodellen*, Institute for Systems Analysis and Operations Research, Der Universitat Der Bundeswehr Munchen, 1991.

[13]For other discussion on the use of this type of model in aggregation, see Hillestad and Juncosa, op. cit.

[14]In this simple, one weapon system scenario, many weapons can fire at a given weapon of the other side at once--the situation for which the square law was formulated.

The only problem is - what are the values to use for A and B, the attrition rate coefficients? If the initial strengths of the opposing forces are known, and also their strengths at a given subsequent time, Lanchester coefficients which will give those same results at the end points can be calculated by the following formulas:

$$A = \frac{\sqrt{x_0^2 - x_f^2}}{t\sqrt{y_0^2 - y_f^2}} \ln(F)$$

$$B = \frac{\sqrt{y_0^2 - y_f^2}}{t\sqrt{x_0^2 - x_f^2}} \ln(F)$$

where

$$F = \frac{x_f\sqrt{y_0^2 - y_f^2} + y_f\sqrt{x_0^2 - x_f^2}}{x_0\sqrt{y_0^2 - y_f^2} + y_0\sqrt{x_0^2 - x_f^2}}$$

and

$$x_f = x(t_f)$$
$$y_f = y(t_f).$$

The time t_f is the end point at which the Lanchester system is to fit exactly to the simulation results. So, given results from another model, such as the deterministic simulation, Lanchester coefficients can be found to give the same final result for the overall battle, starting from the same initial conditions. We'll now describe some results using this third model with the coefficients derived from the deterministic simulation.

The graphs in Fig. 20 show total defender and attacker strength over time in the battle, starting from when the defender is able to open fire. The attacker speed of advance is 45 km/hr, and the deterministic simulation result shown used company groups. The second line on each graph shows strength using a constant coefficient Lanchester square law model.

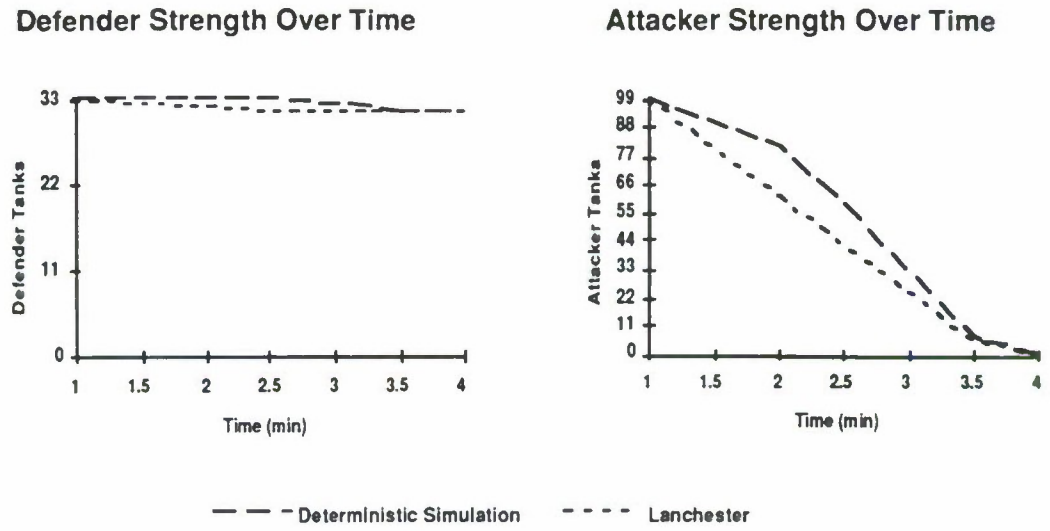


Fig. 20—Constant Coefficient Lanchester Results Fitted to the Company Level Deterministic Simulation

For both attacker and defender, the constant coefficient Lanchester model gives higher casualty rates in the early stages of the battle; in the simulation, lethalities start low and increase as the range closes. Both representations end up at the same point, because the constant coefficients were calculated to achieve this. Halfway through the battle, however, the difference in attacker losses is about 20 tanks, or 20% of initial strength. This might not matter if only the end result was to be used elsewhere, but if intermediate results are needed, or the battle was interrupted by the arrival of reinforcements, the differences might be important.

The differences become even more pronounced as one attempts to fit to the more aggregate deterministic simulation at the battalion and regiment level. Figure 21 shows the same plots, but based on a run of the deterministic simulation with battalion sized groups. The same effects are seen; they are more significant for the defender than in Fig. 20 because the deterministic

simulation at this level of aggregation gives higher defender losses.

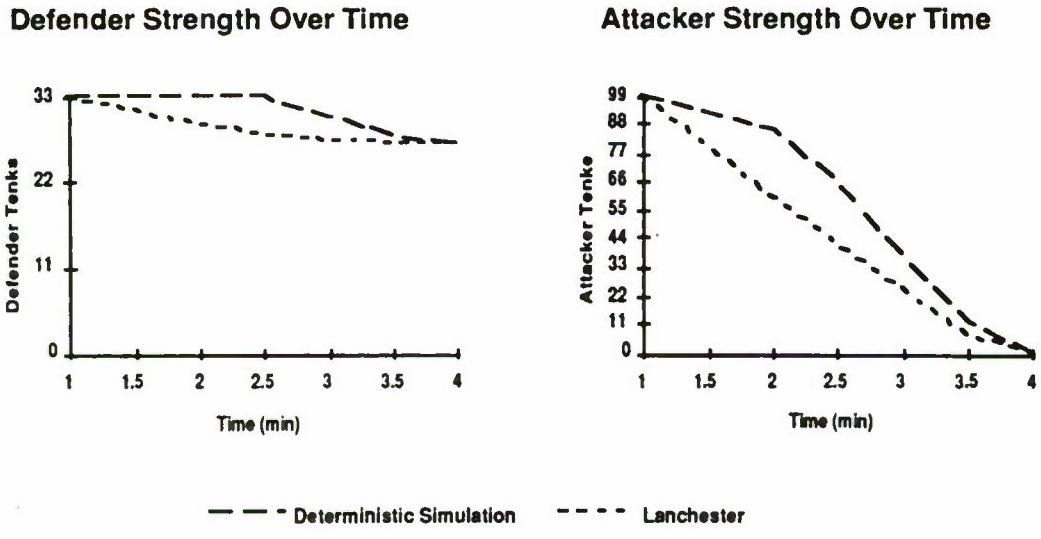


Fig. 21—Constant Coefficient Lanchester Results Fitted to the Battalion Level Deterministic Simulation

Figure 22 shows the equivalent plots where the attacker is represented in the deterministic simulation by a single regiment size group. With this level of aggregation, the deterministic simulation gives an attacker victory. The same differences between the simulation and the constant coefficient Lanchester at intermediate times can be seen but they are even more pronounced.

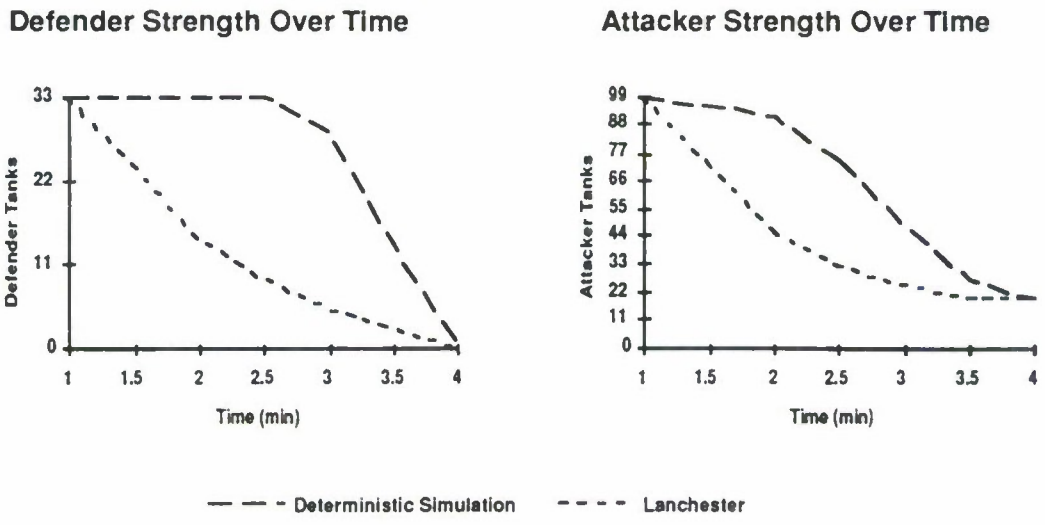


Fig. 22—Constant Coefficient Lanchester Results Fitted to the Regiment (Battle) Level Deterministic Simulation

The constant coefficient model does not represent the two stages of the battle—the stage where the defender uses his range advantage and the attacker cannot fire, and the second stage where both sides can fire. One option, given that it is easy in this scenario to define two distinct stages of the battle, is to approximate it by two constant coefficient Lanchester battles rather than one.

Figure 23 shows the results for the company level aggregation again, but with an extra line on each graph, showing the effect of using a two-stage constant fit. The first stage represents that part of the battle in which only the defender can fire; the defender's attrition coefficient is simply the number of attackers killed per defender per unit time, found by taking the attacker losses in the deterministic simulation at the end of this stage of the battle. The second stage is a constant coefficient Lanchester battle, taking the position in the deterministic simulation at the end of the first stage as the initial strengths, the position at the end of the battle as the final strengths, and calculating

Lanchester coefficients to fit. As can be seen, this produces a much better fit to the deterministic simulation results.

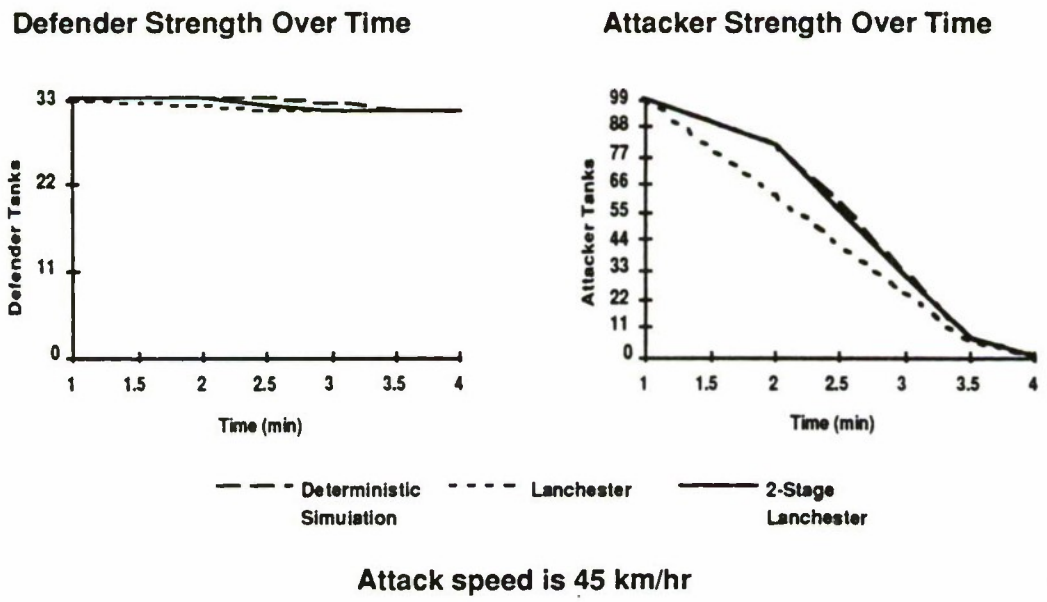


Fig. 23—Fitting Two-Stage Constants to the Company Level Deterministic Simulation

Figures 24 and 25 show that the two stage Lanchester approximation produces much better intermediate results for the battalion and regimental aggregations as well. A common feature of the battles at the three levels of aggregation is that there are two distinct stages to the conflict, each with different characteristics. A constant coefficient representation must recognize this if it is to produce a good approximation. Whether the deterministic simulation results are themselves valid at the higher aggregations does not affect this argument. Given an understanding of the nature of the battle, a simpler model fit can be found. This is encouraging in that these much simpler approximations could be used in place of more computationally intensive models to give essentially the same results.

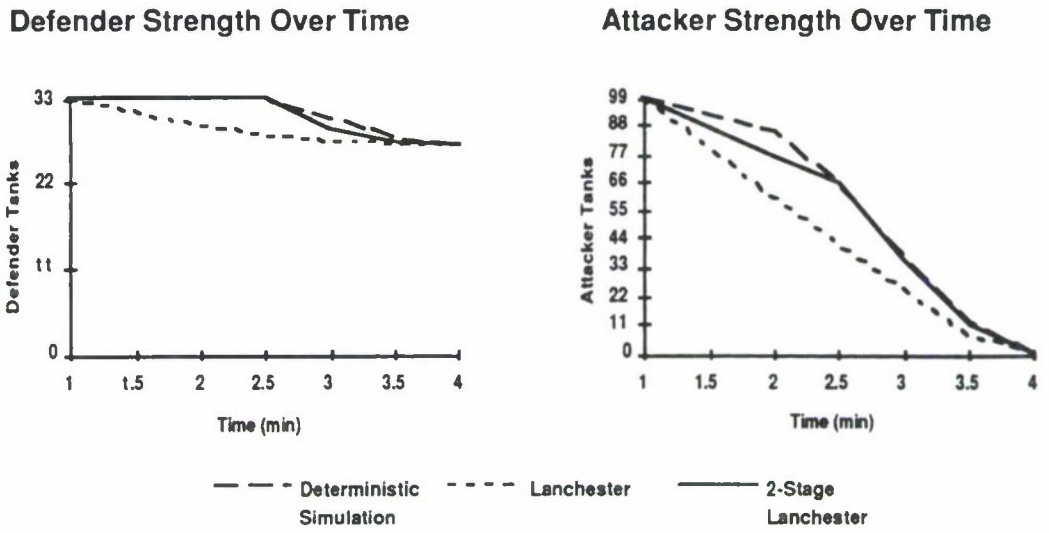


Fig. 24—Fitting Two-Stage Constants to the Battalion Level Deterministic Simulation

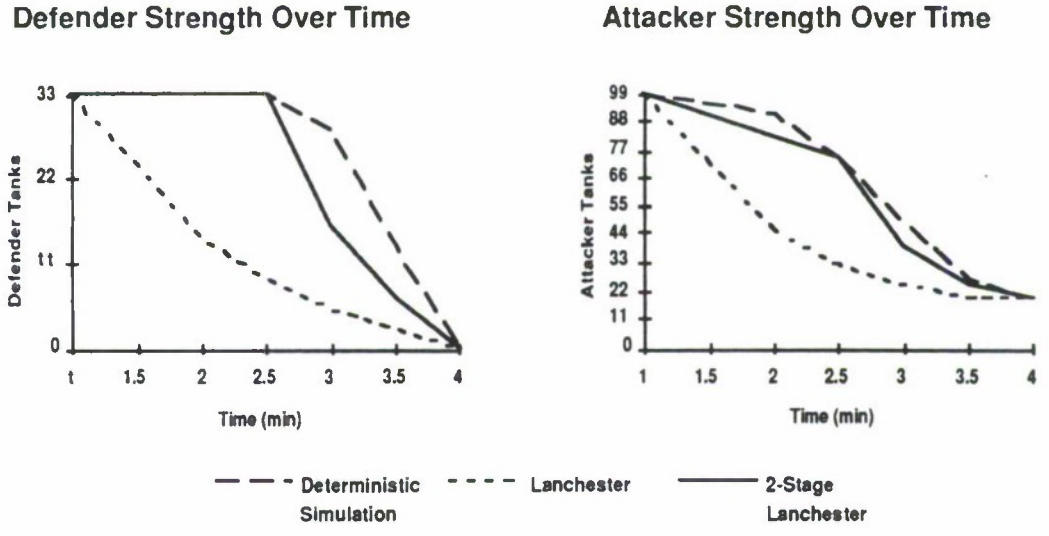


Fig. 25—Fitting Two-Stage Constants to the Regimental Level Deterministic Simulation

The problem, of course, is that of determining the coefficients without having to run the more detailed model. We

have not yet attempted to determine how and whether the Lanchester coefficients could be obtained from knowledge of the battle parameters and configurations without using the detailed simulation.

6. SOME CONCLUSIONS

We've presented a mass of experimental results. What more general points can be drawn? First, it should have been observed in the examples from the detailed stochastic simulation that common intuition about outcomes, causes, and effects is frequently wrong. Since the battle configuration was 3 to 1 in favor of the attacker, one might have concluded, based on the 3:1 rule, that this should lead to a fair fight given some defender advantage. (In our simulation this was given by the Pk versus range advantage.) However, the outcome was highly dependent on the attacker speed and deployment. Also, the effect of projectile time of flight caused a significant reduction in the effective rate of fire of the defender so that when this was not taken into account in a deterministic simulation, the outcomes differed significantly.

The "fair fight" conditions in which both defender and attacker won a significant number of battles also created results with the largest statistical variance. Furthermore, the battle outcomes were distributed bimodally with few cases ending up as a draw even when the average indicates an even battle.[15] This variance was not apparent in the deterministic model. Unfortunately, considerable defense analysis is done in this regime and much of that analysis is done with deterministic models. This can mean that decisionmakers are not really provided analysis which uncovers the considerable uncertainties surrounding that analysis and their possible policy decisions. It should be possible to do better. For example, if one can define the probability of winning or losing with a deterministic model, the

[15] This was also observed in Hofmann, op. cit.

numbers of survivors in these two cases could be approximated by assuming distributions of the types found using stochastic models.

We showed that it is possible to scale results for different levels of resolution but that these scale factors are not obvious. One might suppose that the appropriate scale factor was 67% in the example given because the deployment was 2 up and 1 back. However, this ended up favoring the defender unfairly relative to the more detailed results. In fact, a scale factor of 75% was more appropriate for the specific example given. We suspect that this is situation dependent but not have not tested that fact. The conclusion we reached is that we do not yet know how to scale results.

Even simpler approximations such as the constant coefficient, square law Lanchester one used by us are possible. However, these are also situation dependent and some knowledge of how the battle is likely to progress is needed. In fact, it is probably necessary to divide the battles into stages and have some time varying component of the attrition equations to obtain an accurate representation. We do not know how easy it is to predict these stages or to predict the coefficients themselves from basic knowledge of the scenario, force capabilities, and deployments.

We observed another benefit of this exercise. One learns considerably more about both the problem and the models and modeling assumptions within them when more than one model is used on the same problem. For example, we observed the importance of the time of flight of the projectile on the results by comparing the stochastic model with the deterministic one. The effect of time of flight would have been inherent in the detailed stochastic results but not apparent unless a set of cases were run in which it was varied. If only the deterministic model had been run, the importance of time of flight would have been overlooked. In some of the few defense analysis efforts where different models have been used in parallel, the authors have seen the increase in understanding and insight into the problem. Despite possible additional costs of such a seemingly redundant approach to

analysis, the potentially large differences in results as demonstrated herein due to model aggregations, approaches and assumptions would argue for some parallelism in approach.

Finally, it seems desirable to develop a broader effort of this nature across the defense analysis and simulation community. We have only scratched the surface. One should investigate the effects of simulations with various aggregations of weapon types.[16] How does one represent the different possibilities for fire allocation in a more aggregate model for example? Recently, using our detailed stochastic simulation with two weapon types with distinctly different characteristics we showed that dramatically different results could be obtained with two "reasonable" but different fire allocation policies. How does terrain and line of sight affect the outcomes? It was shown by one of the authors that terrain which reduced the line of sight and firing opportunities shifted the balance toward the attacker because the attacker could get through the zone in which the defender had an advantage with fewer losses. This is contrary to the expectations of some that rough terrain would favor the defender. We were also able to show that when the defender advantage in range was removed and replaced by a "first shot" advantage, the results shifted dramatically toward the attacker. Thus, further investigation should consider what the defender advantage really is - our scenario was only one hypothetical case - and how this could be represented in more aggregate models.

An organized effort like this but on a larger scale is both necessary and possible. Increasingly, simulation is being used and proposed within the defense community for training, extensions of testing, and other forms of analysis. The possibilities for bad training, wrong lessons, and bad analysis as a result of arbitrary aggregations and cross coupling of models of differing resolution will increase as well. At the same time, the proliferation of simulations distributes the problem of doing some

[16] See Hillestad and Juncosa, op. cit., for a theoretical discussion of this problem.

organized military science with those simulations. Hopefully, the results described in this paper will spark some additional concern and interest within the community.

Resolution Changes and Renormalization for Partial Differential Equation Combat Models

Bruce W. Fowler
Advanced Systems Concepts Office
Research, Development and Engineering Center
U.S. Army Missile Command
Redstone Arsenal, AL 35898-5242
(AMSMI-RD-AC)

Recently, considerable research effort has been expended on partial differential equation (PDE) models of combat. These PDE models enjoy the natural characteristic of combining movement with attrition unlike earlier models based on ordinary differential equations such as Lanchester's equations. An additional benefit is that these PDE models also admit of some degree of flexibility in changing the resolution of the model on the fly. We present an approach to PDE model resolution change using invertible transformations from finite difference to finite element representations. Renormalization of the force distributions, highly desirable when proceeding to higher resolutions, is developed using terrain, tactical, and control implications as a basis.

I. Introduction

The question of resolution is as old as computer Combat Simulations, if not Combat Simulations in general. Clearly, resolution in War Games would tend to be fixed by the scale of available maps, and the amount of work that could be performed by the support staff. With the advent of computers and high level programming languages, however, settable, if not adjustable, resolution could become the norm. In general, settable meant that the space scale[a] of the simulation could be set initially, but would be universally constant throughout the range of the simulation and during its execution. Thus, one simulation could represent an amount of terrain that could vary by about as much as two orders of magnitude and the organizational level of the simulation by as much as one order of magnitude. For example, one simulation might represent battalion-brigade level combats through its resolution settability.

This settability did not, in general, extend to internal resolution changes within the scope of the simulation either temporally or spatially. Various *ad hoc* experiments were performed to allow approximations to resolution scalability. Computer limitations restricted many of these experiments. Only recently have attempts been made to develop computer combat simulations that have some inherent form of resolution variability. These simulations, however, seem to be oriented towards having the capability to define different regions with different spatial resolution scales (e.g. UCCATS).

While geographic variation is one clear need for different space resolutions, there are others. These include process variation, event variation, and aggregation variation. Clearly, the spatial scale for urban, forrest, or highly broken terrain needs to be different from that for

---

[a] For time stepped simulations, this also implied the ability to set the time scale as well.

open terrain, on Line of Sight considerations if nothing else. Other variations offer other examples: units travelling to contact should require a different scale of resolution than engaged units; the resolution of search for unit position (intelligence) should be different from that of search for targets (acquisition); and the operation of networks calls for different scales. Certain networks, such as those for C^3I and supply may have several scales.

Clearly then, as our vision of how to model and simulate combat expands, the need to change resolution scales during the course of a simulation's execution becomes evident. Thus, models which allow simulations to change resolution "on the fly" become desirable. This desire is not always compatible with model and computer limitations. We should like these changes to be totally elastic and reversible in both directions, although this may be more than a model can deliver. One class of model that does offer considerable inherent potential for resolution scale change is that of partial differential equations(PDE).

II. Multi-Component, Multi-Phase PDE Models

In PDE models, force strengths are represented as densities, the number of elements per area (or volume for some air elements,) rather than as simple numbers of elements. The concept of component aggregation carries over directly, so we are able to write differential equations of the general Boltzman transport type[1]

$$\frac{\partial \rho_i}{\partial t} + \vec{v}\bullet\vec{\nabla}\rho_i + \vec{a}\bullet\vec{\nabla}_v\rho_i = -\left(\frac{\partial \rho_i}{\partial t}\right)_{collisions}, \qquad (1)$$

where the force strength component densities ρ_i are functions of time, parameters, position, and velocity. Because these are fundamentally transport equations (the left hand side is just a directional derivative,) they inherently admit representation of combat movement and forces (accelerations.) Modifications of these terms are opportunely possible to account for natural dispersive or frictional effects From a mathematical standpoint, the force strengths are non-Newtonian fluids - they may compress or expand. Work is underway to develop insight into the mathematical form of combat processes in this model. It has been speculated, for example, that C^2/C^3 has the form of a colligative force which constrains diffusion (dispersion.)

One elaboration offered naturally by the PDE model is a route to the modeling of more diverse combat processes and the re-aggregation/de-aggregation of components under combat. This may be important in light of efforts to address changes of combat posture. A recently advanced model for such changes addresses military units of essentially divisional organizational level.[2] Since aggregation at divisional level would compromise the benefits of component aggregation, consideration of re-aggregation becomes important. Further, re-/de-aggregation are also important in terms of bootstrapping to, or finding commonality with, strategic and operational level models.

While this elaboration does not technically require the PDE formalism, the generality presented by that formalism contributed to its inception. The concept of **phase** aggregation accounts for the fact that the elements of a force strength component may be simultaneously occupied in performing several combat processes at once, or they may only perform process in a serial manner. In particular, they may be searching for enemy units, acquiring enemy elements

as targets,[b] engaging those targets, moving, communicating, resupplying, or even suppressed. The elements performing each of these processes are called a **phase** of a **component**. Obviously, some processes are serial while other are parallel. Thus, elements may occupy more than one phase at a time. Conservation of elements across phases now becomes a mathematical consideration.

Consider a component that is in combat, and in the interests of simplicity, is moving,[c] acquiring targets, and engaging the targets. We designate the total force strength density by ρ_i, the phase acquiring targets by ρ_i^s, and the phases engaging targets of type j by ρ_{ij}^e. In Bonder-Farrell attrition rate theory, these two phases might be aggregated. The acquisition process, however, is Linear-like and the engagement process is Quadratic-like, or they are both Lanchester-Poisson-like with different forms.[3] This suggests another rationale for phase aggregation.

If both phases move as a parallel process, and acquire and engage as serial processes, then the attrition differential equations may be written as

$$\frac{\partial \rho_i^s}{\partial t} + \vec{v}\bullet\vec{\nabla}\rho_i^s + \vec{a}\bullet\vec{\nabla}_v\rho_i^s = -\sum_j \beta_{ij}\rho_j\rho_i^s + \sum_j \gamma_{ij}\rho_{ij}^e - \sum_j \gamma_{ji}\rho_{ji}^e f_i^s \,, \tag{2}$$

for the search phase, where the first right hand side term is the transport of search phase elements into engagement phase elements, the second is the transport of engagement phase elements into search phase elements, and the third is attrition of the search phase elements by enemy engagement phase elements, and

$$\frac{\partial \rho_{ij}^e}{\partial t} + \vec{v}\bullet\vec{\nabla}\rho_{ij}^e + \vec{a}\bullet\vec{\nabla}_v\rho_{ij}^e = \beta_{ij}\rho_j\rho_i^s - \gamma_{ij}\rho_{ij}^e - \sum_j \gamma_{ji}\rho_{ji}^e f_{ij}^e \,, \tag{3}$$

for the engagement phases. The β matrix represents the acquisition process (assumed here to be simply Linear,) the γ matrix represents the engagement process, and f_i^s, f_{ij}^e represent fire allocation in the attrition term. Component conservation has the form,

$$\rho_i = \rho_i^s + \sum_j \rho_{ij}^e \,. \tag{4}$$

We have drawn no distinction for any acquisition differences between the acquisition and engagement phase elements. Modification to account for this would be straightforward.

If the engagement phase cannot shoot on the move, then equation (3) becomes

$$\frac{\partial \rho_{ij}^e}{\partial t} = \beta_{ij}\rho_j\rho_i^s - \gamma_{ij}\rho_{ij}^e - \sum_j \gamma_{ji}\rho_{ji}^e f_{ij}^e \,, \tag{5}$$

---

[b] In the process, they are acquiring (and hopefully rejecting) friendly elements. The extension to fratricide consideration is natural.

[c] If it is not moving, the problem degenerates into ordinary differential equations.

which has more of the appearance of a standard (ordinary) attrition differential equation since it has no motion terms. This formalism also admits parallel processes, including simultaneous acquisition and engagement, although the resulting differential equations are quite complicated.

One of the motivations for investigating phase aggregation was a paper on armored force field trials that showed the number of units engaging enemy units to be a convex function of time.[4] As the engagement progressed, the number of engaging units increased, peaked, and then decreased. While there were some change of combat posture considerations in the trials, the presented behavior could be emulated by a phase aggregated attrition formalism only with difficulty. To exhibit this situation, we present an illustrative example in Figures 1 and 2. We performed calculations without movement (i.e., using ordinary differential equations,) for two components, each with two phases. Acquisition was represented by a Linear-like term, and engagement by a Quadratic-like term. Parallel calculations for combined Lanchester attrition differential equations of the form,

$$\frac{dA}{dt} = -\frac{\beta_{12} A \gamma_{12}}{\beta_{12} A + \gamma_{12}} B \,, \tag{6}$$

were performed as a comparison. This combined equation is essentially an expression of Bonder-Farrell attrition rate theory. Figure 1 presents a comparison of the force strength components calculated using the combined Lanchester equations, and using the phase model. Clear differences in the curves are obvious. Figure 2 presents the detailed phase trajectories. The convex behavior of the engagement phases due to attrition is clearly shown, and could have been intensified if one of the components had begun to withdraw (changed posture.)

The phase aggregation model also allows consideration of other combat processes and weapon systems. While direct fire weapon systems operate very rapidly (in general) compared to movement, this is not the case for Air Defense, or long range missiles such as FOG-M or ATACMS. Normally, their effects would be approximated in a multi-phase, space aggregated model by time delayed attrition rates. For phase aggregation, this operation can be approximated by introducing additional phases.[d]

Additionally, because the PDE model is basically a transport theory model, the emulation of C^3 and C^3I become possible. A recent effort states that there are four modeling approaches to C^3I:[5]

> (i) Engineering approach - technical properties of data collection (sensors) and transmission are important;
>
> (ii) Connectivity approach - C^3I represented as a network that passes messages between nodes;
>
> (iii) Combat Outcomes approach - the effects of information on combat are modeled; and

---

[d] These phases may have considerably simpler differential equations, however.

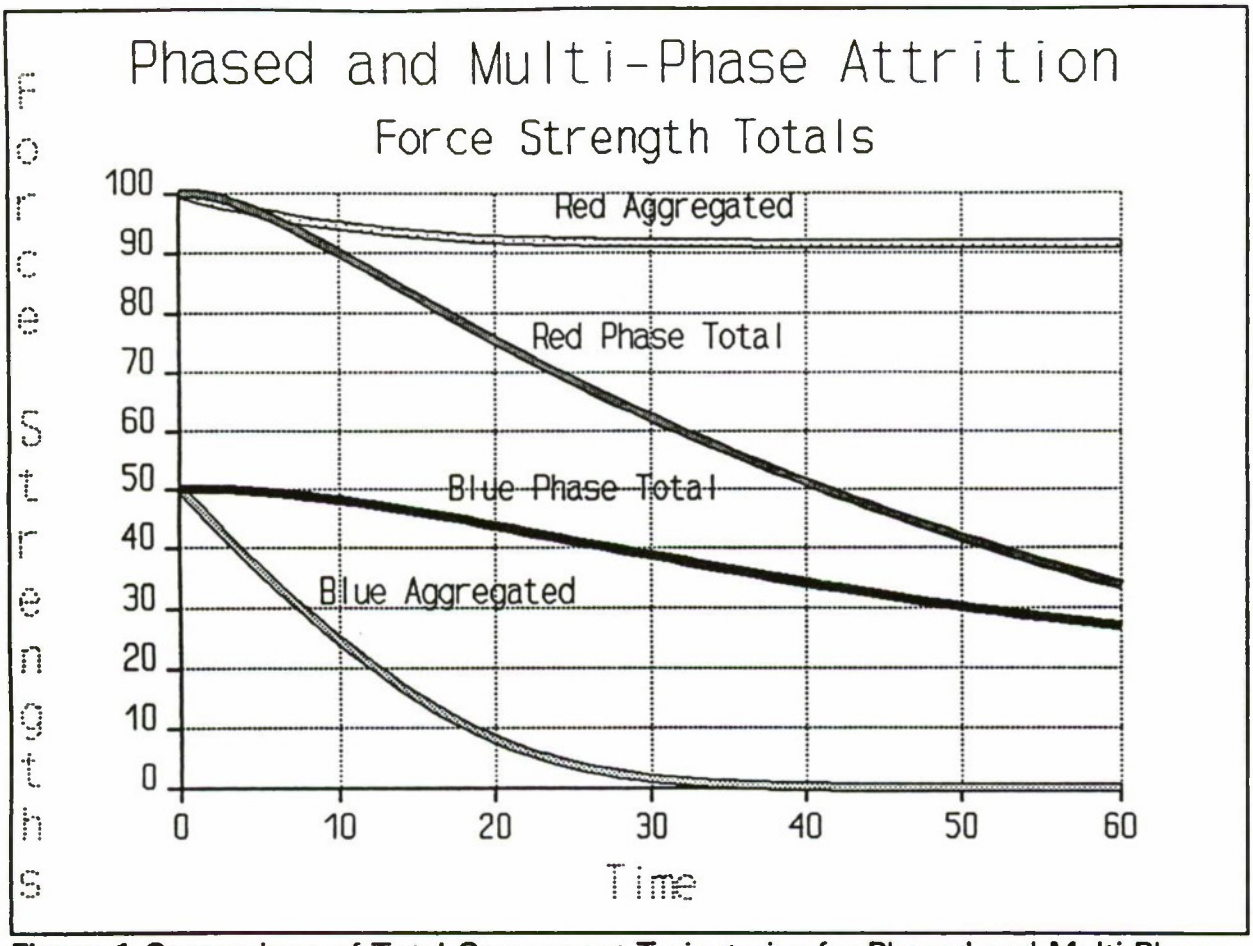


Figure 1 Comparison of Total Component Trajectories for Phased and Multi-Phase Aggregated Attrition.

(iv) Order of Battle approach - integrates (i)-(iii) to depict the enemy order of battle and model command responses to that information.

The PDE-phase aggregation approach captures (i) and (ii) naturally, and can then provide input to an approach (iii) model whose output (of approach (iv) type) are then transported back to units. In principle then, the combination makes for easier C^3/C^3I and logistics (since logistics is described by transport theory) integration in the simulation.

Finally, because the simulation, the PDE solver code, is inherently numerical in nature, it is more amenable to quality control of the code itself. The models of the phases and components can then be configuration controlled separately. Additionally, there is a large body of scientific knowledge and experience which can be carried over into combat studies.

III. Implementations

Two common approaches to solving PDEs of this type are the methods of Finite

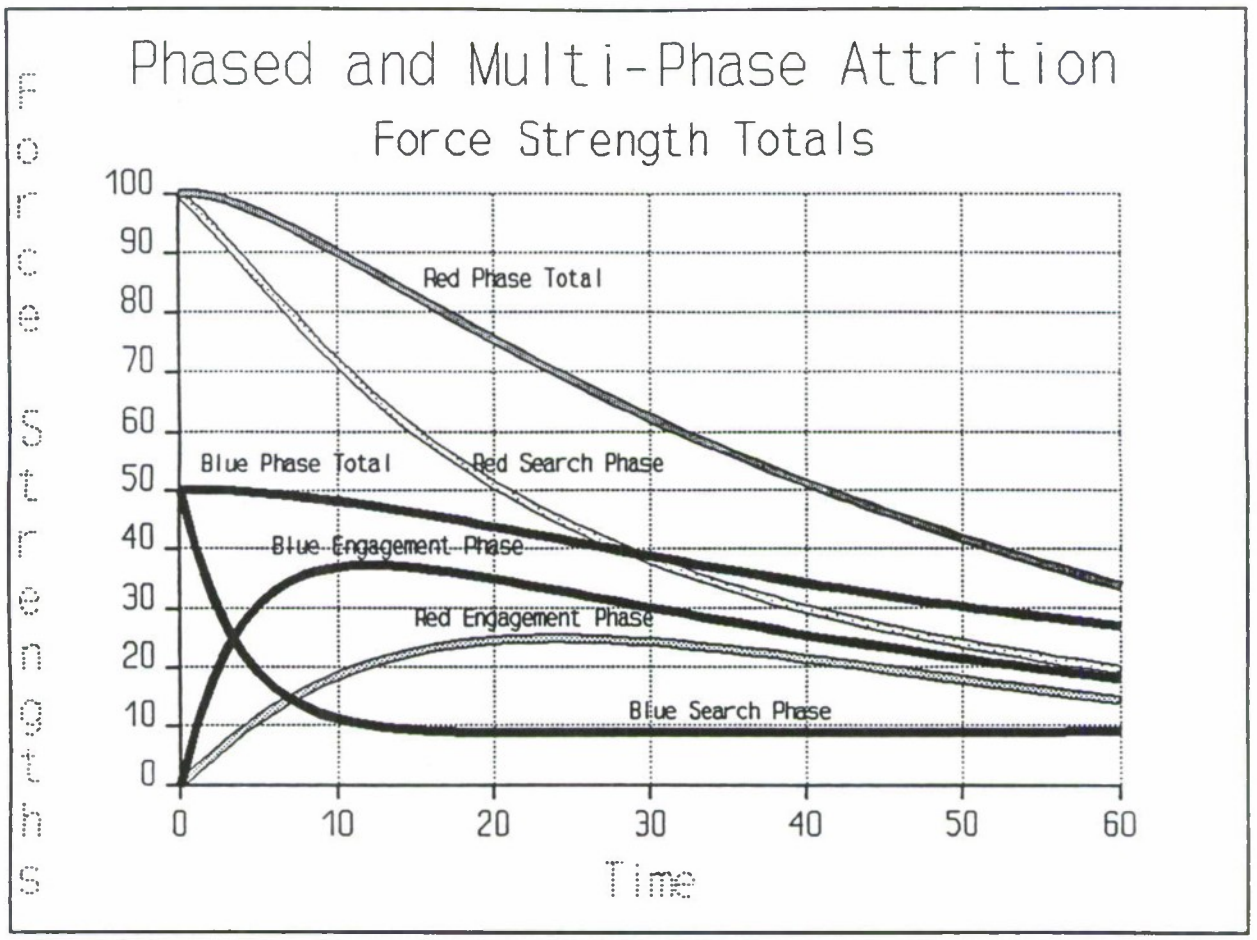


Figure 2 Comparison of Trajectories of Phases.

Differences (FD) and Finite Elements (FE). Both impose a point structure on space and time, and numerical values of the solutions are calculated at the points. In the FD method, derivatives are approximated by forward (or backwards, e.g.,) differences defined from

$$\ln(1 + \Delta) = \Delta x \frac{d}{dx}, \tag{7}$$

where Δx is the distance between grid points in the x direction. The difference operator Δ is defined by

$$\Delta f(x) \equiv f(x + \Delta x) - f(x). \tag{8}$$

Within the context of equation (1), a pair of rectangular grids could be defined for each force strength density ρ_i. Each grid would represent the value of the density at the spatial points (x,y) of interest and one value of time (t). The finite difference approximation to the PDE would be written in the form

$$\rho_i(\underline{r}_{jk}, t + \Delta t) = F[\rho_{i'}(\underline{r}_{j'k'}, t), \zeta], \tag{9}$$

where $\underline{r}_{ij}$ are the grid point (space) coordinates.

The FE method differs from the FD method in several ways. It also numerically approximates the value of the solutions at points, but these points are arranged in a closed network which is not necessarily regular. The solution is approximated by extremization of the differential equation (and its boundary and constraint conditions) on closed subsets of the network, (usually) of three or four points. The solutions are approximated using interpolating functions which are often simple polynomials. In general, there is one interpolating function per point, although the functions on the boundaries of the network may be trivial. Coefficients of series of the interpolating functions are then developed and solved using matrix equations. The need to invert large matrices can be a limitation of this method, just as the need for many grid points may be a limitation of the FD method.

Two key common points in the two methods are the nature of time solution representation and the form of the network. A common FE technique for representing time solution is the Crank-Nicholson method which is essentially a finite difference.[6] In principal, the FE method is quite good for solving the spatial part of the PDE, but the time solution may be FD in form. A second commonality occurs when the network is a regular grid, and the values of the coefficients of the interpolating functions can be directly related to the finite difference grid point values. Under these conditions, the solutions by the two methods can become equivalent.

While it is possible in principle to cast the entire solution process in the FE method, and thereby allow a variable resolution in space inherently, this approach has potential limitations. Additionally, assumption of Lanchester-Poisson form search/engagement terms and non-locality complicate the extremization process. Visualization within the framework of a rectangular space grid is easier. Further, since the PDEs are usually well behaved (within the bounds of some constraints such as rigid nonnegativity of force strength densities,) retention of the rectangular grid does not necessarily compromise accuracy.[e]

Given that it is desirable to retain the FD method of solution on a rectangular space grid for the PDEs, we may now recognize that the interpolating polynomials of the FE method offer a means for dynamic, "on the fly" resolution changes.

IV. Resolution Shifts

In the simplest resolution shifts, such as one associated with terrain, the form of the PDE approximation, equation (9), will not change, although the values of Δx, Δy, will. The simplest of these is a shift within a part of the grid to greater resolution. This adds grid points. Consider that we will only shift resolution within part of the original grid.[f] If we represent the spatial coordinate within the original grid by $\underline{r}_{jk}$, which is defined as

---

[e] It is widely believed that only a rectangular grid can be used for FD calculations. This is not true. For example, a hexagonal grid is feasible if the difference operators are redefined on the grid form.

[f] We will assume that the new section of higher resolution grid lies completely within the boundaries of the original grid - we add no new area to the representation.

$$r_{ij} = x\, j\, \Delta x + y\, k\, \Delta y, \tag{10}$$

where x and y are unit vectors in the x and y directions respectively, then the portion of the grid to be resolution shifted is defined by r_{ij} (lower left corner), and $r_{i'j'}$ (upper right corner). This assumes a rectangular area to be shifted - clearly this is not necessary - we make the assumption only for simplicity of presentation.

Within this area, we assume a new set of spatial differences $\Delta x'$, $\Delta y'$ ($< \Delta x, \Delta y$) and indices l,m such that

$$r'_{lm} = r_{jk} + x\, l\, \Delta x' + y\, m\, \Delta y' \leq r_{j'k'}, \tag{11}$$

where the equality holds only if $\Delta x'$ and $\Delta y'$ are integer multiples of Δx and Δy. Within the rectangular area formed by r_{jk}, r_{j+1k}, r_{jk+1}, r_{j+1k+1} (which we will designate A_{jk}) will be at least one point r_{lm}' for some value of l,m. If we designate the interpolating function $\Phi_{i,jk}$ using the values of ρ_i at the four points comprising A_{jk}, we may calculate the values of ρ_i' at the points r_{lm}'. A similar set of interpolating functions $\Psi_{i,lm}$ (of greater functional complexity,) allow the calculation of the values of ρ_i at points r_{jk} along the inner boundary of A_{jk}. This constitutes a reversible transformation between the two resolutions. The forms of the PDEs are the same inside and outside the area, interpolation is required to match transport inside and outside, however.

Having defined a new combination of resolution areas, the solution may now be calculated using the same technique as before if the time differential does not change. If it does, and the shifted time differential is an integer multiple of the unshifted time differential, then the solution form for the shifted area may be calculated independently if there are no process interactions across the area boundary. If there are, it is probably necessary to accept a global shift of time differential.[ε]

As a word of warning, these resolution shifts must preserve stability. It may be necessary to perform a series of shifts in successive time differential steps to reach the desired level of shift.

The shift from lower resolution to higher resolution tends to neglect higher order spatial derivatives in the density distribution. The distribution of force strength density resulting from the resolution shift may be unrealistic, especially if the spatial shift is accompanied by an organizational (aggregation) resolution shift. This leads to recognition of a need to renormalize the relative structure of the force strength densities in the shifted area. One particular example of this is terrain effected. If the resolution shift increases the resolution of the terrain, it is common that the force strength density be adjusted to compensate for trafficability. A good indicator function for weighing terrain effected renormalization seems to be the directional spacing of contour lines. The road network is also a factor here. A further consideration may be tactical posture of units (i.e. defensive, offensive).

These renormalizations are currently in a state of development. They tend to be rather

---

[ε] This seems to result regardless of whether we use the FD or FE method.

untidy, but seem to be necessary if the merits of resolution shifts are to be realized.

V. Conclusions

The basis for a means of resolution shift for PDE models of combat has been advanced here. Admittedly, we have not explicitly addressed shifts from higher to lower resolution, but these shifts seem to present problems primarily in terms of density conservation. To date, we have not found any good means to accommodate time differential shifts except globally, although special cases may exist. We present no example results of calculations with this method with only the excuse that we are still examining the stability of the method itself. Nonetheless, the inherent ability of these PDE models to accommodate resolution shifts is a strong point in their favor, and may help to increase the use of these models in practical studies.

REFERENCES

1. Mohling, Franz, "Kinetic Theory of Gases", Chapter III in Statistical Mechanics, John Wiley and Sons, Inc., New York, 1982, pp. 93-146.

2. Dupuy, T. N., Understanding Defeat, Paragon House, New York, 1990.

3. Fowler, Bruce W., "A *Perestroika* Program for Modeling and Simulation.", Proceedings of the 1991 Calloway Gardens Workshop on Modeling, Simulation and Gaming of Warfare, December 1991.

4. McNaught, K. R., Paper TA-7, ORSA-TIMS National Convention, Nashville, Tennessee, May 1991.

5. Camm, Frank, *et al.*, "Evaluating Intelligence Systems That Support Deep Fires", RAND Arroyo Center Report R-3795-A, October 1989.

6. Reddy, J. N., An Introduction to the Finite Element Method, McGraw-Hill Book Company, New York, 1984, pp. 51-52.



A MATHEMATICAL TECHNIQUE FOR ANALYSIS AND DERIVATION OF MODELS OF VARYING RESOLUTION

by

DARRELL V. FOWLER
INFORMATION SYSTEMS ENGINEERING CENTER

5 May 1992

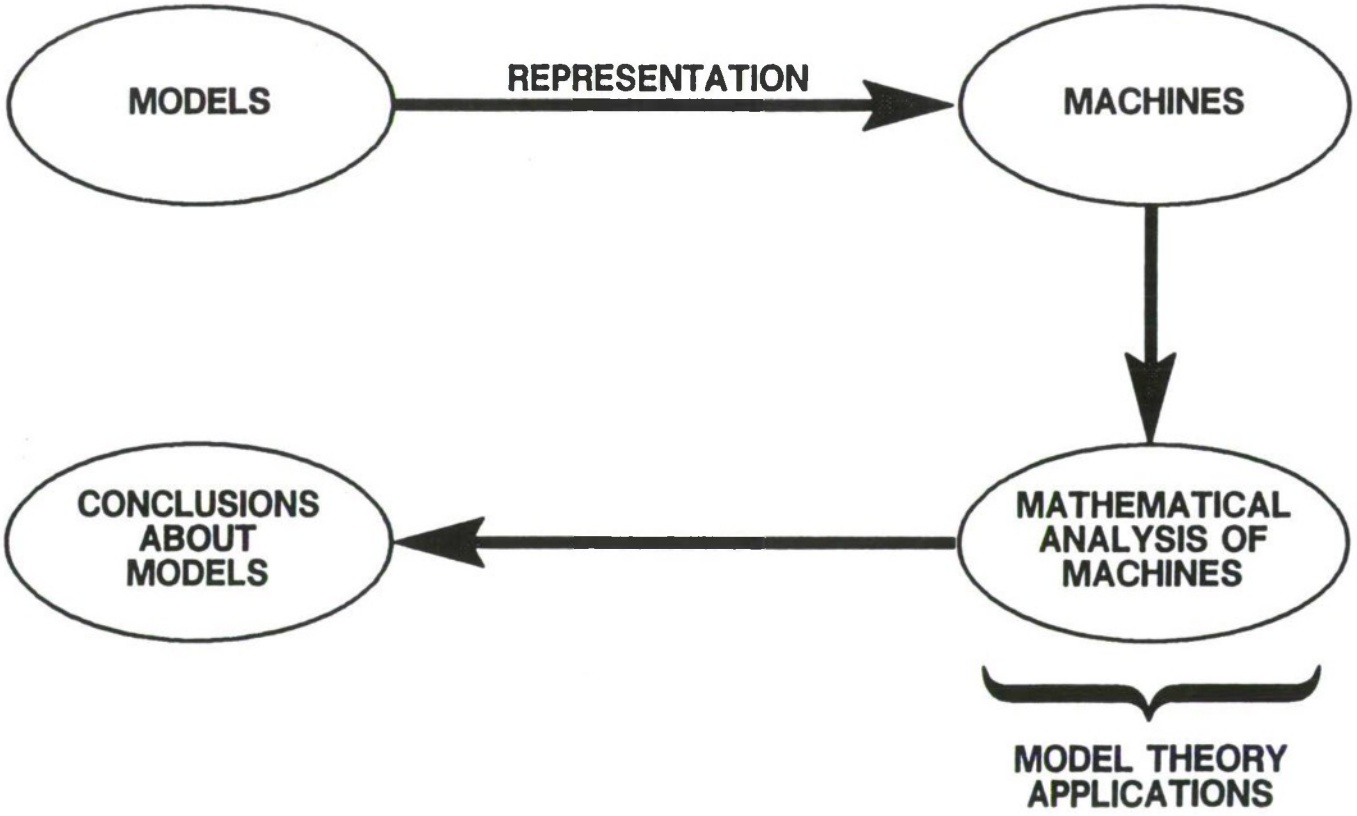
341

TOPICS



- ANALYTIC APPROACH – MODEL THEORY
- REPRESENTING A MODEL AS A COMPOSITE MACHINE
- EXAMPLE – THE MARCHING SOLDIERS MODEL -- MSM (3,4)
- HOMOMORPHIC IMAGES AND PRESERVED PARTITIONS
- DERIVING CONSISTENT AGGREGATED MODELS
- EXAMPLE THE MARCHING SQUAD MODEL -- MSQM (3,4)
- CONCLUSIONS

THE ANALYSIS APPROACH



MODEL THEORY



SYNTHESIZE FROM;

- **MATHEMATICAL SYSTEMS THEORY**
  - AUTOMATA THEORY
  - CYBERNETICS
  - INFORMATION THEORY
  - CONTROL THEORY
  - MEASURE THEORY
- **ALGEBRA**
- **SET THEORY**

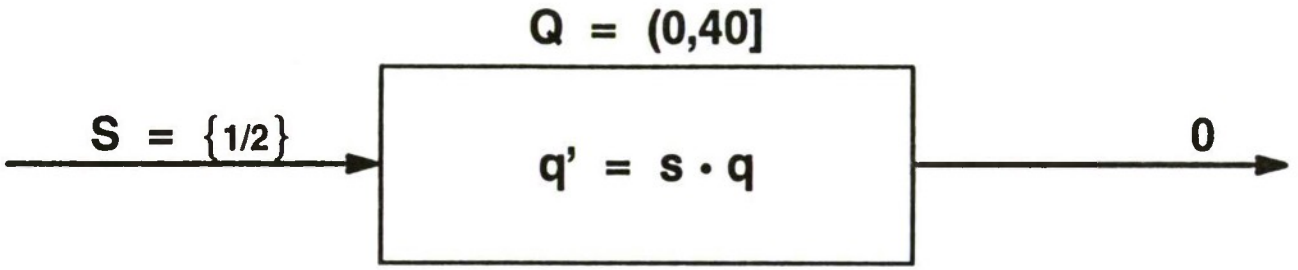
MACHINE DEFINITION

A MACHINE IS:

- A SET S = {s} CALLED INPUTS
- A SET Q = {q} CALLED STATES
- A TRANSFORMATION M: S x Q $\longrightarrow$ O

- A SET O = {o} CALLED OUTPUTS
- A TRANSFORMATION N: S x Q $\longrightarrow$ O

MACHINE EXAMPLE



BEHAVIOR

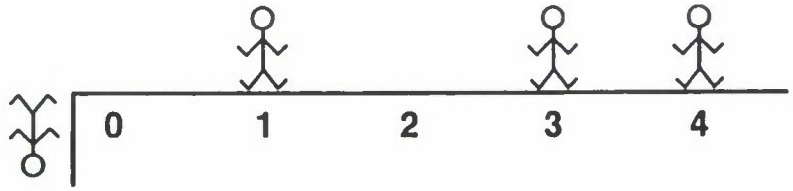
if $q_0 = 32$

STATE SEQUENCE IS 32, 16, 8, . . .

if N = M

OUTPUT SEQUENCE IS 32, 16, 8, . . .

THE MARCHING SOLDIERS MODEL – MSM (n,D)



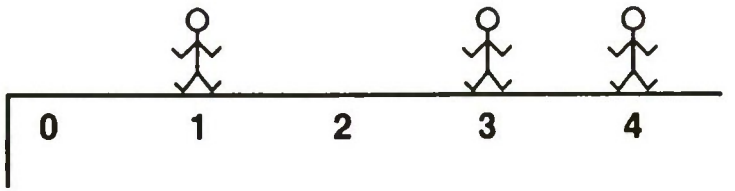
MSM (nD)

- n SOLDIERS IN SINGLE FILE
- MARCH 1 UNIT LEFT ON ORDER
- SEPARATED BY 1 OR MORE DISTANCE UNITS
- START WITH LAST MAN DISTANCE D FROM A CLIFF
- ANY SOLDIER WHO REACHES THE CLIFF AT 0 FALLS OFF AND DIES

THE PROBLEM

- IN A HIGH RESOLUTION MODEL SHOW POSITION AND STATUS (DEAD OR ALIVE) OF EACH SOLDIER
- DERIVE ALL POSSIBLE CONSISTENT LOWER RESOLUTION MODELS

MSM (3,4)



MACHINE

$$Q = \{(x_1, x_2, x_3) \mid \text{integer} \leq (4,3,2)\}$$

$$S = \{\text{“march”}\} \qquad 0 = Q$$

$$M(\text{“march”}, (x_1, x_2, x_3)) = \begin{cases} x_i - 1 \text{ for all } x_i > 0 \\ 0 \text{ for all } x_i = 0 \end{cases}$$

$$N = M$$

MSM (3,4) (CONCLUDED)



BEHAVIOR

		INPUT S	STATE Q	OUTPUT 0
			q = (4,3,2)	
CYCLE	1	"march"	(3,2,1)	(3,2,1)
	2	"march"	(2,1,0)	(2,1,0)
	3	"march"	(1,0,0)	(1,0,0)
	4	"march"	(0,0,0)	(0,0,0)

COMPOSITE MACHINES



- **SIMPLE MACHINES CAN BE COMBINED ACCORDING TO CERTAIN RULES TO FORM MORE COMPLEX MACHINES**

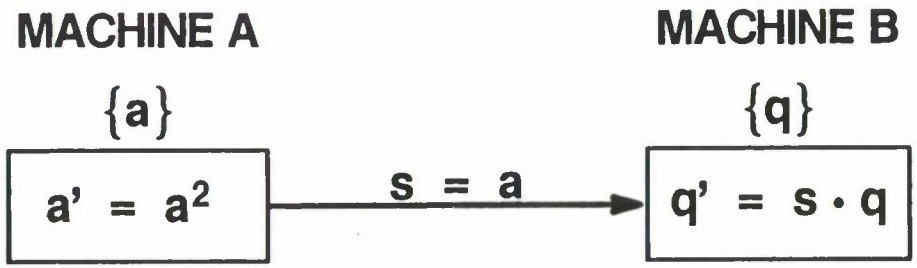
  $\oplus$ **DIRECT SUM**

  $\otimes$ **DIRECT PRODUCT**

  $\longrightarrow$ **COUPLING**

- **COMPLEX MACHINES CAN BE DECOMPOSED INTO SIMPLE COUPLED COMPONENTS**
  - MATHEMATICAL TESTS (NECESSARY AND SUFFICIENT)
  - FINITE DECOMPOSITION ALGORITHMS

COUPLED MACHINES



COUPLING

C = A ⟶ B

STATES OF THE COUPLED MACHINE ARE PAIRS (a,q)

IF $a_0 = 3$ AND $q_0 = 2$

TRAJECTORY OF THE COUPLED MACHINE IS

(a,q) = (3,2), (9,6), (81,54), . . .

HOMOMORPHISM

TWO MACHINES T AND T' ARE HOMOMORPHIC

IF THERE EXISTS A TRANSFORMATION h: Q⟶Q'

BETWEEN THEIR STATES SUCH THAT:

$$M'(s, h(q)) = h(M(s,q))$$

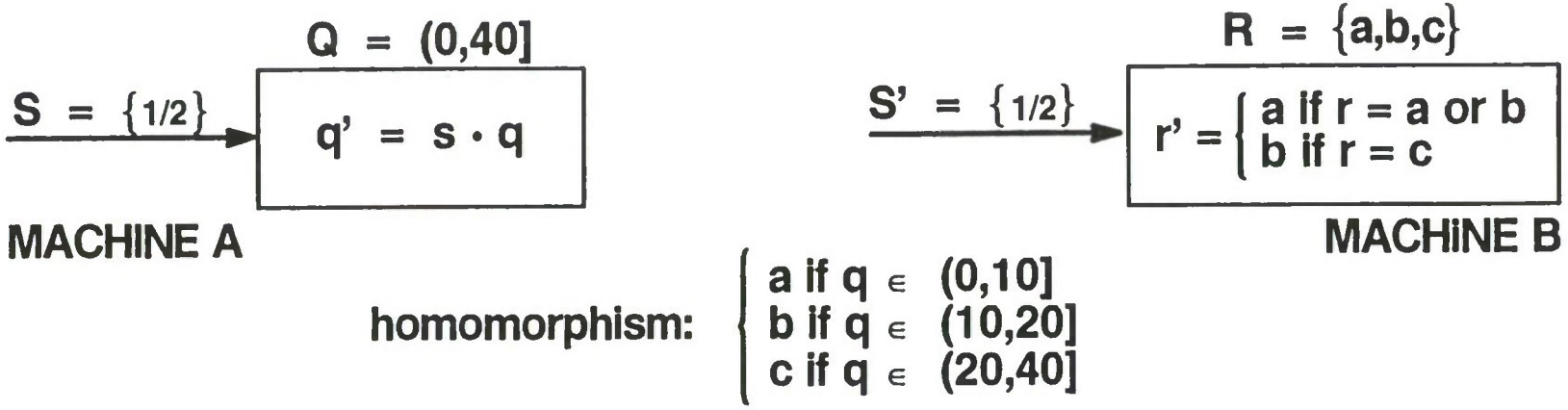
FOR ALL s and q.

ISOMORPHISM

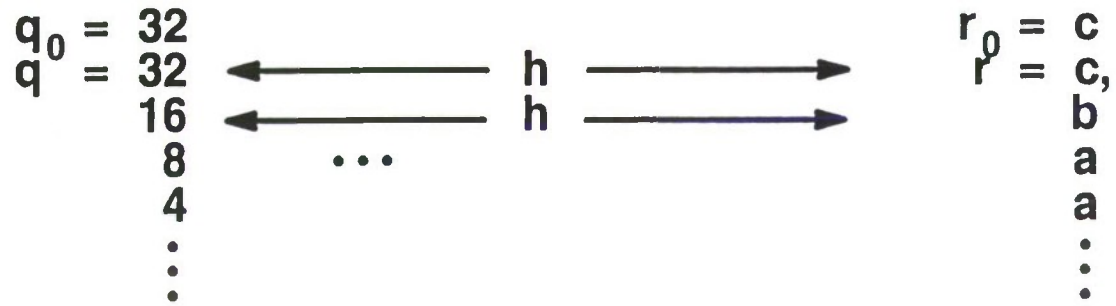
$T' \equiv T$ if h is 1–1 mapping

ISOMORPHIC MACHINES ARE THE SAME IN THEIR BEHAVIOR -- JUST DIFFERENT NAMES OF THINGS

HOMOMORPHISM EXAMPLE



BEHAVIOR



5 May 1992
Fowler

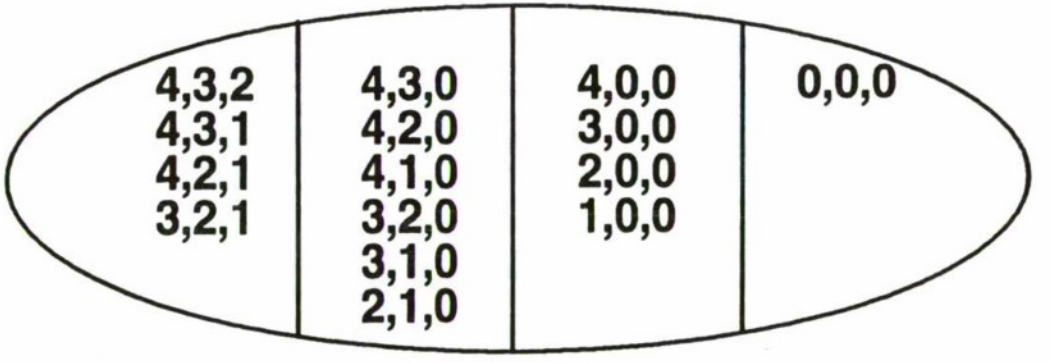
PARTITIONS

A PARTITION OF THE STATE SET DIVIDES THE SET INTO NON-OVERLAPPING SUBSETS. A PARTITION IS DEFINED BY AN EQUIVALENCE RELATION ON THE STATE SET

$q_a \equiv q_b$ if and only if rule

EXAMPLE: IN MSM (3,4) , THE RULE

$(x_1, x_2, x_3) \equiv (x'_1, x'_2, x'_3)$ IFF SAME NUMBER OF ELEMENTS ARE 0



THERE ARE 15 STATES IN MSM (3,4). THERE ARE 4 SUBSETS IN THE PARTITION. EACH SUBSET CAN BE INTERPRETED AS AN OBJECT ITSELF.

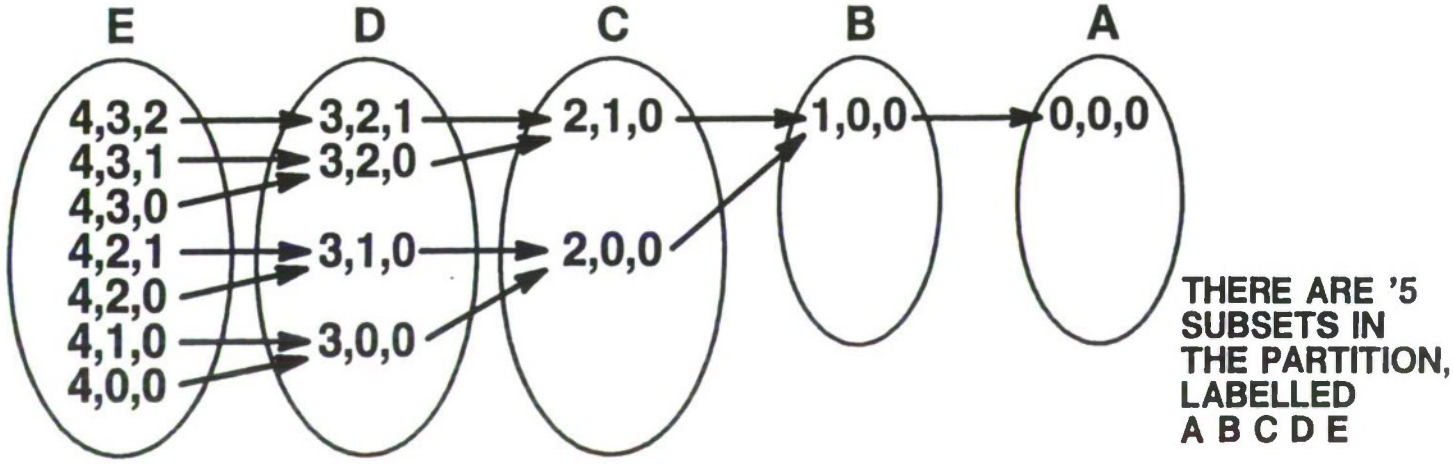
PRESERVED PARTITION

A PRESERVED PARTITION IS ONE WHOSE SUBSETS ARE MAPPED WHOLLY INTO SUBSETS OF THE PARTITION BY THE TRANSITION FUNCTION M. THAT IS

$$q_a \equiv q_b \longrightarrow M(s,q_a) \equiv M(s,q_b)$$

EXAMPLE: $I^{N}_{\wedge}MSM\ 3,4$ THE PARTITION

$$(x_1,x_2,x_3) \equiv (x'_1,x'_2,x'_3) \text{ iff } x_1 = x'_1$$



USE OF PRESERVED PARTITIONS AND HOMOMORPHISMS



- **THERE IS A 1–TO–1 CORRESPONDENCE BETWEEN PRESERVED PARTITIONS OF A MACHINE AND HOMOMORPHIC IMAGES OF THE MACHINE; I.E.,**
  - EACH PRESERVED PARTITION DIRECTLY DEFINES A HOMOMORPHIC IMAGE MACHINE
  - EVERY HOMOMORPHIC IMAGE MACHINE IS ASSOCIATED WITH A PRESERVED PARTITION
- **PRESERVED PARTITIONS ARE COMPUTABLE APPLYING A KNOWN ALGORITHM**
- **THE IMPLICATIONS FOR EXAMINING MODELS OF VARYING RESOLUTION ARE THAT:**
  - ALL LOWER RESOLUTION MODELS THAT BEHAVE CONSISTENTLY WITH A HIGH RESOLUTION MODEL CAN BE DIRECTLY DERIVED FROM THAT MODEL
  - INCONSISTENCY CAN BE READILY PROVEN (AND ISOLATED FOR CONSIDERATION)

DERIVING A CONSISTENT MARCHING SQUAD MODEL

THE PRESERVED PARTITION IN MSM (3,4) DEFINES A RELATED HOMOMORPHIC IMAGE MACHINE (CALL IT MSQM (3,4))

NO CHANGE TO INPUT SET: S = {"march"}

STATES ARE THE FIVE LABELLED SUBSETS OF THE PARTITION.

$$Q = \{A,B,C,D,E\}$$

TRANSITION FUNCTION IS INDUCED BY THE MAPPING OF ELEMENTS WITHIN THE SUBSETS

$$M \quad (\text{"march"},q) = \begin{cases} D \text{ if } q = E \\ C \text{ if } q = D \\ B \text{ if } q = C \\ A \text{ if } q = B \\ A \text{ if } q = A \end{cases}$$

THUS TRANSITIONS ARE

$$E \longrightarrow D \longrightarrow C \longrightarrow B \longrightarrow E$$

DERIVING A CONSISTENT MARCHING SQUAD MODEL (CONCLUDED)



POSSIBLE INTERPRETATIONS ARE

A – STRENGTH IS 0

B – SQUAD TAIL AT 1, STRENGTH 1/3

C – SQUAD TAIL AT 2, STRENGTH 1/3 OR 2/3 (UNDETERMINABLE)

D – SQUAD TAIL AT 3, STRENGTH 1/3 OR 2/3 OR 1 (UNDETERMINABLE)

E – SQUAD TAIL AT 4, STRENGTH 1/3 OR 2/3 OR 1 (UNDETERMINABLE)

INCONSISTENT AGGREGATIONS

QUESTION: CAN WE CONSTRUCT AN AGGREGATED MODEL (LOWER RESOLUTION) THAT BEHAVES CONSISTENTLY WITH MSM (3,4) AND THAT USES CENTER OF MASS AND SURVIVING STRENGTH AS ITS STATES? I.E.,

$Q = (\bar{x}, n)$ WHERE $\bar{x}$ = CENTER OF MASS OF LIVE "SQUAD"
n = SURVIVING INDIVIDUALS / 3

ANSWER: NO!

REASON: THE PARTITION WHICH AGGREGATES STATES WITH THE SAME $\bar{x}$ AND n IS NOT PRESERVED. PROOF BY COUNTER EXAMPLE, I. P.

$(3,2,0) \equiv (4,1,0)$ SAME SUBSET -- $\bar{x} = 2.5$, $n = 2/3$

BUT $M(S,(3,2,0)) = (2,1,0)$ WHICH IS IN SUBSET $\bar{x} = 1.5$ $n = 2/3$

$M(S,(4,1,0)) = (3,0,0)$ WHICH IS IN SUBSET $\bar{x} = 3$ $n = 1/3$

SO $M(S,(3,2,0)) \neq M(S(4,1,0))$

CONCLUSIONS



- MODELS CAN BE THEORETICALLY REPRESENTED AS COMPOSITE MACHINES
- MODEL THEORY PROVIDES EFFECTIVE MATHEMATICAL TOOLS TO DEAL WITH ISSUES OF CONSISTENCY IN MODELS OF VARYING RESOLUTION.

REFERENCES

1. Ashby, W. Ross, An Introduction to Cybernetics. New York, NY, John Wiley & Sons, 1966.

2. Beer, Stafford, Decision and Control. New York, NY, John Wiley & Sons, 1966.

3. Booth, Taylor, Sequential Machines and Automata. New York, NY, John Wiley & Sons, 1967.

4. Ginsburg, Seymour, An Introduction to Mathematical Machine Theory. Reading, MA, Addison Wesley, 1962.

5. Halmos, Paul, Naive Set Theory. Princeton, NJ, Van Nostrand, 1965.

6. Hammer, Preston C. (Ed), Advances in Mathematical Systems Theory. University Park, PA, PA State University Press, 1969.

7. Lang, Serge, Algebra. Reading, MA, Addison Wesley, 1965.

8. Nelson, R. J., Introduction to Automata. New York, NY, John Wiley & Sons, 1968.

9. Shannon, Claude E. and Weaver, Warren, The Mathematical Theory of Communication. Urbana, IL, University of IL Press, 1949.

10. Wiener, Norbert, Cybernetics. Cambridge, MA, MIT Press, 1961.

11. Wilder, Raymond L., Introduction to the Foundations of Mathematics. New York, NY, John Wiley & Sons, 1965.

12. Fowler, Darrell V., Systems Theory as a Metalanguage for Analysis of Models, USACAA SP74-8, Bethesda, MD, USA Concepts Analysis Agency, 1974.

VARIABLE RESOLUTION MODELING IN MATHEMATICS

DR. JULIAN PALMORE†

University of Illinois
Department of Mathematics
1409 W. Green Street
Urbana Il 61801-2975

ABSTRACT

Several examples of variable resolution modeling (VRM), aggregation, consistency and conjugacy from the fields of arithmetic, chaotic dynamics, and graphics are given. The purpose in using three different mathematical settings is to define with respect to each field the concepts of variable resolution modeling and to illustrate their applicability. I intend to demonstrate a concept of measure in several dimensions of variable resolution modeling.

INTRODUCTION

Resolution is about analyzing details. It specifies the scale(s) by which detail is defined. Variable resolution modeling requires several sets of scales in which to identify details of interest. Consistency has to do with dynamics and the representations of a system by two or more models. Let P be a process that compares the states of two models for a given time interval [0,T]. Two models are said to be *consistent in the process P* if for all t, $0 \leq t < T$, the states of the two models lie within a prescribed distance with respect to the process P. Aggregation and disaggregation are mappings of data from one set of variables to another. To aggregate a dynamical system means to represent the system by another that uses fewer variables. To disaggregate a system means to represent the dynamical system in greater detail with more variables.

Here are several examples: a fractal (one scale) or a multifractal (two or more scales); a planetary system with one set of scales (for example, the International System of Units, MKSA, for planetary motion) or two sets of scales (MKSA, for planetary motion and CGSA, for planetary surface effects). If there is no coupling between the sets of scales, then one can view the model at different resolutions. An interesting example is given by the system of astronomical constants. This system has a theoretical basis for dependencies of secondary units on primary units. Secondary units are derived from the primary units by theoretical relations. One cannot change a scale within the system without regard to changing all other scales that depend upon it.

There are several terms that arise in variable resolution modeling that can be defined and utilized in mathematical settings. These seven terms are RESOLUTION, VARIABLE RESOLUTION,

---

† Professor of Mathematics, UIUC and Principal Investigator, USACERL.

AGGREGATION, DISAGGREGATION, CONJUGACY, CONSISTENCY IN THE AGGREGATE, CONSISTENCY IN THE DISAGGREGATE.

Examples are taken from ARITHMETIC, CHAOTIC DYNAMICS, AND GRAPHICS. We consider ARITHMETIC over the field of real numbers. This allows scaling to be identified with the base of arithmetic. Scaling is a form of resolution. CHAOTIC DYNAMICS is found in unstable dynamical systems. Dynamical systems models have different resolutions with regard to numbers of variables. We consider aggregating a dynamical system, with one resolution, to a dynamical system, with a coarser[†] resolution. This is difficult to implement because the dynamics of the aggregated system will be inconsistent. As an example of a chaotic dynamical system we take the GAUSS MAP. This map yields continued fraction expansions of real numbers. COMPUTATIONAL CHAOS exists in computational models of chaotic dynamics at different resolutions. Information loss is an essential feature of chaotic dynamics. The arithmetic operation of division is the implementation of a Bernoulli shift, an elementary chaotic operation. Information is not lost in the division process for rationals if the period of the expansion is identified and the transient block and a period are given. Finally, GRAPHICS is a field in which the mathematical information displayed by graphics can be compared to the mathematical information sought. The disparity between graphics resolution and resolving mathematical detail is illustrated effectively in Section III.

GENERIC DEFINITIONS

1. RESOLUTION: A set of scales. The smallest scale(s) at which information is recognized.

2. VARIABLE RESOLUTION: Two or more sets of scales. The smallest scale(s) in each set of scales at which information is recognized.

3. AGGREGATE[††]: To assemble, to combine into a mass. We use this term to denote a mapping from particulars to a mass.

4. DISAGGREGATE: To disassemble, to separate into particulars. We use this term to denote a mapping from a mass to particulars.

5. CONJUGACY: Invertibility of aggregation and disaggregation. Conjugacy allows reversing the action of aggregation and disaggregation.

6. CONSISTENCY IN THE AGGREGATE: Commutativity of dynamics and aggregation.

7. CONSISTENCY IN THE DISAGGREGATE: Commutativity of dynamics and disaggregation.

[†] We use the adjectives "fine, coarse," and "high, low," to refer to the degree of resolution. The words "fine, high" refer to increased resolution; "coarse, low" refer to decreased resolution.
[††] For clarity, we will use the phrases "*aggregate X from Y*" and "*disaggregate X to Y.*"

I. ARITHMETIC

ARITHMETIC gives us examples of variable resolution. Division of integers (or reals) requires an algorithmic solution that has variable resolutions of the quotient: lower resolution (left most bits, or "←") and higher resolution (right most bits, or "→"). In FIG. 1, the binary representation of 1/19 by division, we observe that higher resolution bits are constructed after those of lower resolution. Rounding is an example of aggregation. Rounding *aggregates a single digit from a digit string*. Another example of aggregation is to *aggregate a rational number from a digit string*. *Disaggregate a rational number to a digit string* is the reverse process. Thus, division is an example of disaggregating a rational to a binary bit string.

The domain of real arithmetic is the complete ordered field of real numbers. The binary field operations are addition (+) and multiplication (×). These operations induce inverses: subtraction (−) and division (/), respectively. For each integer base of arithmetic, $b \geq 2$, a real number x corresponds to a base b expansion. For x in the range $0 \leq x < 1$, we designate this correspondence *formally* by writing $x = {}_{(b)}0.a_1a_2\ldots$, with $a_k \in \{0,1,\ldots,b-1\}$. One cannot *write down* all expansions; computability considerations restrict to a countable set those real numbers for which a correspondence can be written down.

Let $b = 2$ so that $a_k \in \{0,1\}$. A *normalized* binary number x, $0 \leq x < 1$, is written with a significand S and an exponent E in the form $x = 2^{-E} \times S$, where S lies in the range $1 \leq S < 2$. The significand is written $S = 1.a_1\ldots a_K$. A significand S has *precision* N if $K = N\text{-}1$, that is, $S = 1.a_1\ldots a_{N-1}$. Thus, there are N bits in each significand with precision N and each begins with a leading bit "1". There are a total of 2^{N-1} numbers in the set of significands with precision N. If the exponent E is unrestricted, then one has "finite precision *real* arithmetic." If the exponent E has a finite range, then one has "finite precision *computer* arithmetic." We identify *resolution* with *precision*. The smallest scale in the significand is given by the least significant bit a_{N-1}. We identify *disaggregation of a number* with *binary expansions*. Consequently, we identify the process of *aggregation of a number* as determining a rational number that is within a given tolerance of a real number.

To make aggregation yield a unique answer, we introduce the concept of *best rational approximation*. We use the Euclidean distance on the real numbers, $d(x,y) = |x - y|$, $x, y \in \mathbf{R}$, to measure tolerance. A rational number p/q is a best rational approximation of a real number α to within a distance T, $|\alpha - p/q| \leq T$, if every other distinct rational number, $p'/q' \neq p/q$, with denominator, q', $q' \leq q$, differs from α by a distance greater than T.† By using the uniqueness of aggregation to rational numbers from digit strings by the concept of best approximation we are able to compare different aggregations.

The binary expansion of 1/19 is $0.\underline{000011010111100101}_{18}$. The expansion has period 18. This means that the block of 18 bits repeats indefinitely. All of the information content of the rational number is stored within the block above. In normalized form the number is written $2^{-5} \times \underline{1.10101111001010000}_{18}$. Let us consider now the precisions found in computer arithmetic: N = 24 (single), 53 (double), 64 (extended). The binary bit expansions appear in FIG. 2. Important information is the period of a rational number. A fact of arithmetic is that every

rational number is eventually periodic. By knowing the transient bits of a bit string and all bits in a period block, one can reconstruct the rational number. Let r = p/q and s=u/v be rationals. Consider two operations: *aggregate:* reals → rationals and *disaggregate:* rationals → reals. Each operation requires a *resolution*.

Consistency in the aggregate requires that within the given *tolerance* we have:

$$\begin{array}{ccc} (r,s) & \rightarrow & \times(r,s) \\ \uparrow & & \uparrow \\ (x,y) & \rightarrow & \otimes(x,y) \end{array}$$

where ↑ represents *aggregation from reals to rationals*. "⊗" denotes bit string multiplication.

A specific example of CONSISTENCY IN THE AGGREGATE is instructive. Compare FIG. 3. Let X and Y be numbers that are expressed in binary as X = 0.$\underline{0101}$ = 0.01010101... and Y = 0.$\underline{0011}$ = 0.00110011.... We can aggregate 1/3 from X and 1/5 from Y because X and Y are the binary bit strings of these rational numbers. For finite precision X and Y aggregation is accomplished to a best approximation. Multiplying 1/3 and 1/5 yields 1/15. Multiplying X and Y b yields a bit string Z = 0.00010001...= 0.$\underline{0001}$. Aggregating Z yields rational 1/15.

CONSISTENCY IN THE DISAGGREGATE requires that within the given *precision* we have:

$$\begin{array}{ccc} (r,s) & \rightarrow & \times(r,s) \\ \downarrow & & \downarrow \\ (x,y) & \rightarrow & \otimes(x,y) \end{array}$$

where ↓ represents *disaggregation from rationals to reals*. Compare FIG. 4.

Finally, we consider CONJUGACY IN THE AGGREGATE. This requires both aggregation H and disaggregation h. For conjugacy, two parameters must be given: PRECISION for disaggregation h and multiplication ⊗ and TOLERANCE for aggregation H. Both precision and tolerance can be chosen independently. This causes problems because the diagram may not commute:

$$\begin{array}{ccc} (r,s) & \rightarrow & \times(r,s) \\ \downarrow h & & \uparrow H \\ (x,y) & \rightarrow & \otimes(x,y) \end{array}$$

Figures 5 and 6 show examples of conjugacy in aggregation and its apparent failure. The solution in FIG. 6 is obtained by comparing continued fraction expansions of 2/315 and 31/4883.

II. CHAOTIC DYNAMICS

DYNAMICAL SYSTEMS provide examples of variable resolution and aggregation. If one considers aggregation as a process of representing a dynamical system in N dimensions by aggregated dynamical system using fewer variables, then one can consider aggregation as a process of reducing dynamics from N dimensional space to a space of K dimensions, $K < N$. A problem arises because dynamical systems on N dimensional space rarely reduce consistently so that the dynamics induced on the lower K dimensional space makes sense. There may be inconsistencies in the dynamics induced by reduction. Simple examples of inconsistent reduction can be seen in projection of orbital motions from a phase space of four dimensions to a plane. If we examine a projection of the orbit to two dimensions to show only the position trajectory, then we find that there are points of intersection of the projected orbit with itself that are inconsistent with dynamics in two dimensions. However, disregarding these inconsistencies in the dynamics, we can gain useful information on the location of the orbit.

Consistency in the aggregate can be defined in the following circumstance. An important example of consistency in the aggregate is given a dynamical system with a fractal attractor. Nonperiodic orbits near the attractor tends toward it as time increases. Thus, as time increases the statistics of orbits near the attractor coincide with those of the attractor. A dynamical system that models attractor dynamics will need fewer variables than the original dynamical system. We can consider the attractor and its properties as an aggregation of the original dynamical system.

An example of consistent aggregation with reduction is for a system with first integrals. Here motion is restricted to integral surfaces of lower dimension than the phase space dimension. Thus, the system can be modeled by a dynamics using fewer variables.

PROBLEMS WITH AGGREGATION IN DYNAMICS

1. *ATTEMPT TO REPRESENT A STATE SPACE AND DYNAMICS WITH FEWER VARIABLES.*
2. *TAKE A PROJECTION OF N-SPACE ONTO K-SPACE, WHERE $K < N$.*
3. *BUT DYNAMICS ON N-SPACE RARELY PROJECTS CONSISTENTLY TO DYNAMICS ON K-SPACE.*

CONTINUED FRACTIONS AND THE GAUSS MAP

The GAUSS MAP is defined by f: $x \to 1/x \bmod 1$ for $x \in (0,1)$ and $f(0) = 0$. To write the continued fraction of a real number $x \in (0,1)$, one defines $[1/x]$ as the integer N such that $N \leq 1/x$ and $N + 1 > x$. For $x = 0$ we write $N = 0$. Consider the orbit of a real number $x \in [0,1)$ by the GAUSS MAP. Let $x_0 = x$ and define $x_N = f(x_{N-1})$, recursively, $N \geq 1$. We write $X_0 = 0$ and define recursively $X_N = [1/x_{N-1}]$. Thus, to each orbit of the GAUSS MAP there is an orbit of integers associated with it. This sequence of integers is the continued fraction of the real number x. We observe that if x is a rational fraction, then the orbit of x by the GAUSS MAP reaches 0. A second observation shows that there is no loss of generality in restricting the real number to lie in the unit interval, $0 \leq x < 1$. Let $x \in [0,1)$ and let K be a positive integer. By the mapping $x \to 1/x$ we have $x + K \to 1/(x + K) < 1$. The orbit of $1/(x + K)$ by the GAUSS MAP is $1/(x + K) \to x + K \bmod 1 \ (= x) \to \ldots$ Thus, the continued fraction of $x + K$ begins with K and is written [K; [0;x]], where [0;x] is the continued fraction for x. Consider two examples of this process. Let $x = 113/355$. The continued fraction is written symbolically as **[0; 3, 7, 16]**. This denotes the expansion: $x = 1/(355/113) = 1/(3 + 16/113) = \mathbf{0} + 1/(\mathbf{3} + 1/[\mathbf{7} + 1/\mathbf{16}])$. The reciprocal 355/113 has a continued fraction expansion that is written [3; 7, 16]. Let $x = \pi$. The continued fraction expansion of π does not terminate at 0 because π is not a rational fraction. It begins [3; 7, 15, 1, 292, 1, 1, 1, 2, 1, ...]. A sequence of rational convergents to π begins with the rational fractions: 22/7, 333/106, 355/113, 103993/33102, ...

CHAOS AND CONTINUED FRACTIONS

The GAUSS MAP is a chaotic map with positive Lyapunov exponent $\lambda = \pi^2/(6 \ln 2)$. Every orbit is bounded. The aperiodic orbits are chaotic.

For a rational number $r = p/q$, where p and q are integers, we know by the Euclidean algorithm that the continued fraction expansion is a finite sequence. Given a rational number $0 < r < 1$, we can write the continued fraction as $[0; a_1, \ldots, a_N]$. Aggregating r from the continued fraction yields,

$$r = 0 + 1/(a_1 + [1/a_2 + \ldots + 1/a_N)]\ldots]).$$

The information content of the continued fraction is the same as that of the rational number. No information is lost in the transformation between the space of relatively prime integer pairs and the space of continued fractions. A 1-1 correspondence has been established. However, there are transformations that lose information. Consider the following. Take a rational number $r = p/q$ and expand in base B. Thus, $r =_{(B)} 0.a_1a_2\ldots$, and the expansion is *periodic*. Thus, for 355/113 we have a decimal expansion with period 112,

3.1415929203539823 0088495575221238938053097345132743362831858407079646017699115044247787610619469026548672566371681415929203539823

In finite precision, information is lost. Compare 112 to the number of digits required to write the continued fraction expansion of 355/113.

III. GRAPHICS

Resolution in GRAPHICS is about the detail that can be seen in a display. We demonstrate below that a display can be mathematically correct from the standpoint of the algorithms used to calculate information but, at the same time, misleading because the picture does not convey the sought after mathematical information. There must be a correspondence between the resolution of the display and the resolution of the information sought.

There are several definitions of resolution in graphics.

1) The *smallest scale at which graphical information can be displayed* is the resolution of the display. The resolution of a graphics device is usually represented by the aspect ratio and the number of lines in the pixel lattice. The number of grid points on a graphics device of very high resolution can exceed 7000 × 7000. The number of grid points on a 35mm negative is about 2000 × 3000.

2) The *minimum angular separation of two symbols* that is detectable to the eye. This depends upon the symbols used: i.e. dots, parallel lines, vernier. This minimum angular separation is on the order of 20" of arc and may be as low as 10" with best "seeing".

3) The mathematical resolution of a graphics display is the *scale at which the global mathematical detail of the system is shown.* The mathematical resolution is always coarser than the graphics resolution.

A demonstration of the need to resolve mathematical structure is shown in Figures 7a-7d. The phase portrait is drawn for the integral curves of the vector field:

$$dz/dt = - z^2(z - 1)\overline{(z^2 - z + 1)}$$

Here, $z = x + iy$ and the velocity field is given in complex notation for compactness. The differential equation above represents a system of two first order equations of the form: $dx/dt = f(x,y)$ and $dy/dt = g(x,y)$. The banding shows changes in threshold values of an integral of motion. The edges of the bands depict solution curves of the system of differential equations.† Successful resolution of the mathematical structure is shown in FIG. 7d, where there is a mathematically balanced view of the solution curves.

---

† There are two references for the mathematical and graphical theory for Figures 7a-7d. Various examples of phase portraits are given in color that shows both integral curves and time.

1) Palmore, J.I., Burns, S., and Benzinger, H.E., Jr., 1988, "Ecology Models and Newton Vector Fields," J. Mathematical Biosciences, 90, 221-232.
2) Burns, S., and Palmore, J.I., 1989, "The Newton Transform: An Operational Method for Constructing Integrals in Dynamical Systems," Physica D, 37, 83-90.

DIVISION IN BINARY 1/19 HAS PERIOD 18

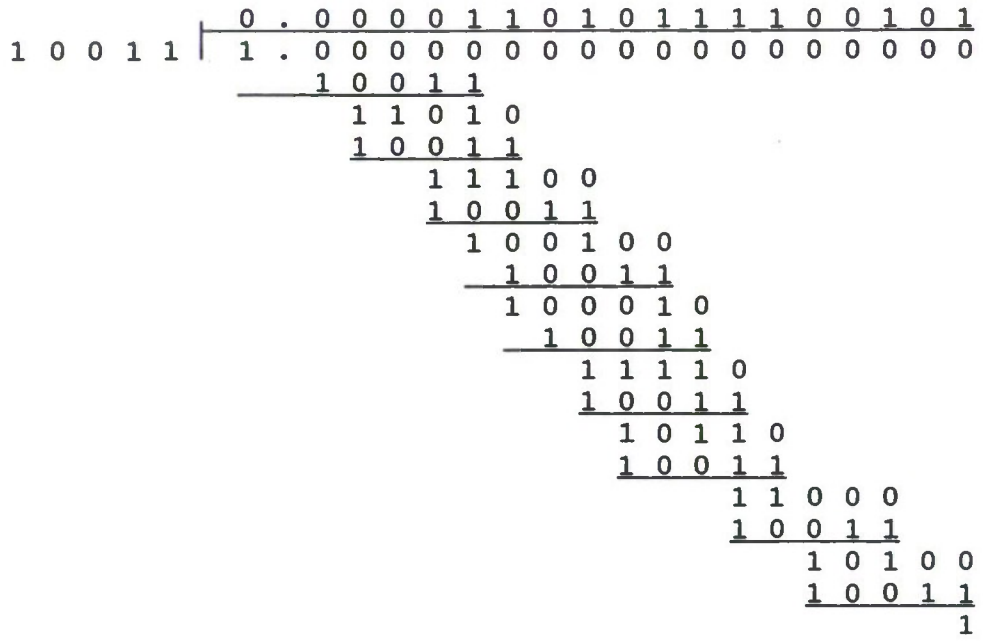


FIG. 1. VARIABLE RESOLUTION IN DIVISION

NORMALIZED UNROUNDED BINARY SIGNIFICANDS OF RATIONAL 1/19

$1/19 = 2^{-5} \times$

$1.10101111001010000110101|_{(24)}111\ldots$

$1.1010111100101000011010111100101000011010111100101000|_{(53)}011\ldots$

$1.101011110010100001101011110010100001101011110010100001101011110|_{(64)}010\ldots$

PRECISIONS OF NORMALIZED ROUND TO NEAREST SIGNIFICANDS OF 1/19

$1/19 = 2^{-5} \times$

$1.10101111001010000110110|(24)$ SINGLE

$1.1010111100101000011010111100101000011010111100101000|(53)$ DOUBLE

$1.101011110010100001101011110010100001101011110010100001101011110|(64)$ EXTENDED

FIG. 2. VARIABLE RESOLUTION IN ARITHMETIC

$$
\begin{array}{lcl}
(1/3,\ 1/5) & \rightarrow & 1/15 \\
 & \times & \\
\uparrow\ (H,H) & & \uparrow\ H \\
(0.a_1a_2..,\ 0.b_1b_2..) & \rightarrow & 0.c_1c_2\ldots \\
 & \otimes &
\end{array}
$$

$$\text{TOLERANCE} = 2^{-N}$$

$$\left\| (1/3,1/5) - (H,H)(0.a_1a_2\ldots,\ 0.b_1b_2\ldots) \right\| \leq 2^{-N}$$

$$\left| 1/15 - H(0.c_1c_2\ldots) \right| \leq 2^{-N}$$

FIG. 3. CONSISTENCY IN THE AGGREGATE

$$
\begin{array}{lcl}
(1/7,\ 1/19) & \rightarrow & 1/133 \\
 & \times & \\
\downarrow\ (h,h) & & \downarrow\ h \\
(0.a_1a_2..,\ 0.b_1b_2..) & \rightarrow & 0.c_1c_2\ldots \\
 & \otimes &
\end{array}
$$

$$\text{TOLERANCE} = 2^{-N}$$

$$\left\| (0.a_1a_2\ldots,\ 0.b_1b_2\ldots) - (h,h)(1/7,1/19) \right\| \leq 2^{-N}$$

$$\left| 0.c_1c_2\ldots - h(1/133) \right| \leq 2^{-N}$$

FIG. 4. CONSISTENCY IN THE DISAGGREGATE

	$\times$	
(1/19, 1/7)	$\rightarrow$	1/133
$\downarrow$ h		$\uparrow$ H
	$\otimes$	
(0.052632, 0.14286)	$\rightarrow$	(0.0075188)

H TOLERANCE $= 10^{-8}$

h precision $= 5$

FIG. 5. CONJUGACY IN AGGREGATION

	$\times$	[[157,1,1],15]
(1/19, 31/257)	$\rightarrow$	31/4883
		2/315
$\downarrow$ h		$\uparrow$ H
	$\otimes$	
(0.052632, 0.12062)	$\rightarrow$	(0.0063486)
		[[157,1,1],16,8,2,1,1,1,6,2]

H TOLERANCE $= 10^{-6}$

h precision $= 5$

2/315 = [157,1,1] = 0.0063492... 31/4883 = 0.0063485...

FIG. 6. APPARENT FAILURE OF CONJUGACY IN AGGREGATION

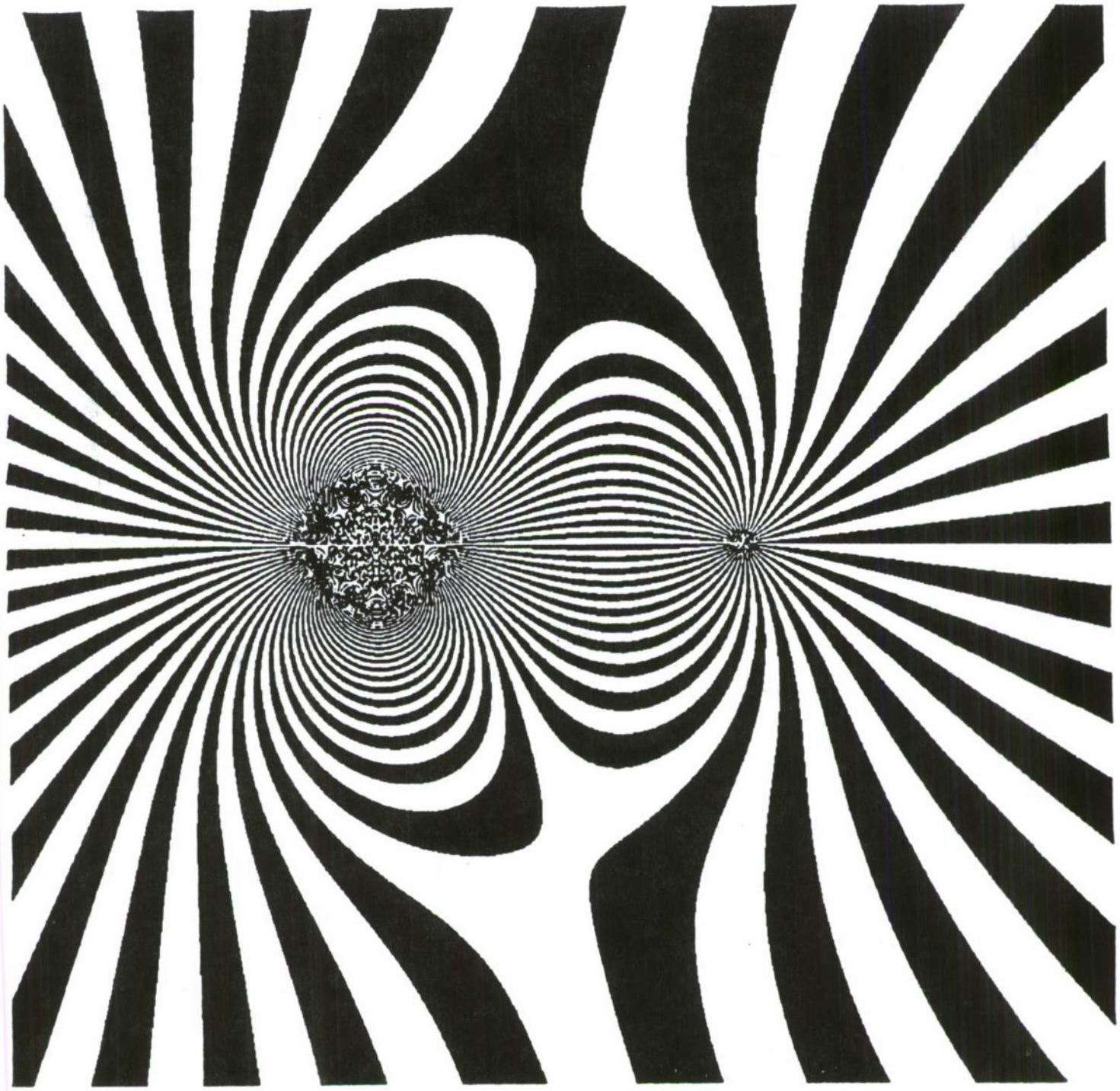


FIG. 7a. PHASE PORTRAIT OF FLOW ON PLANE

INITIAL RESOLUTION

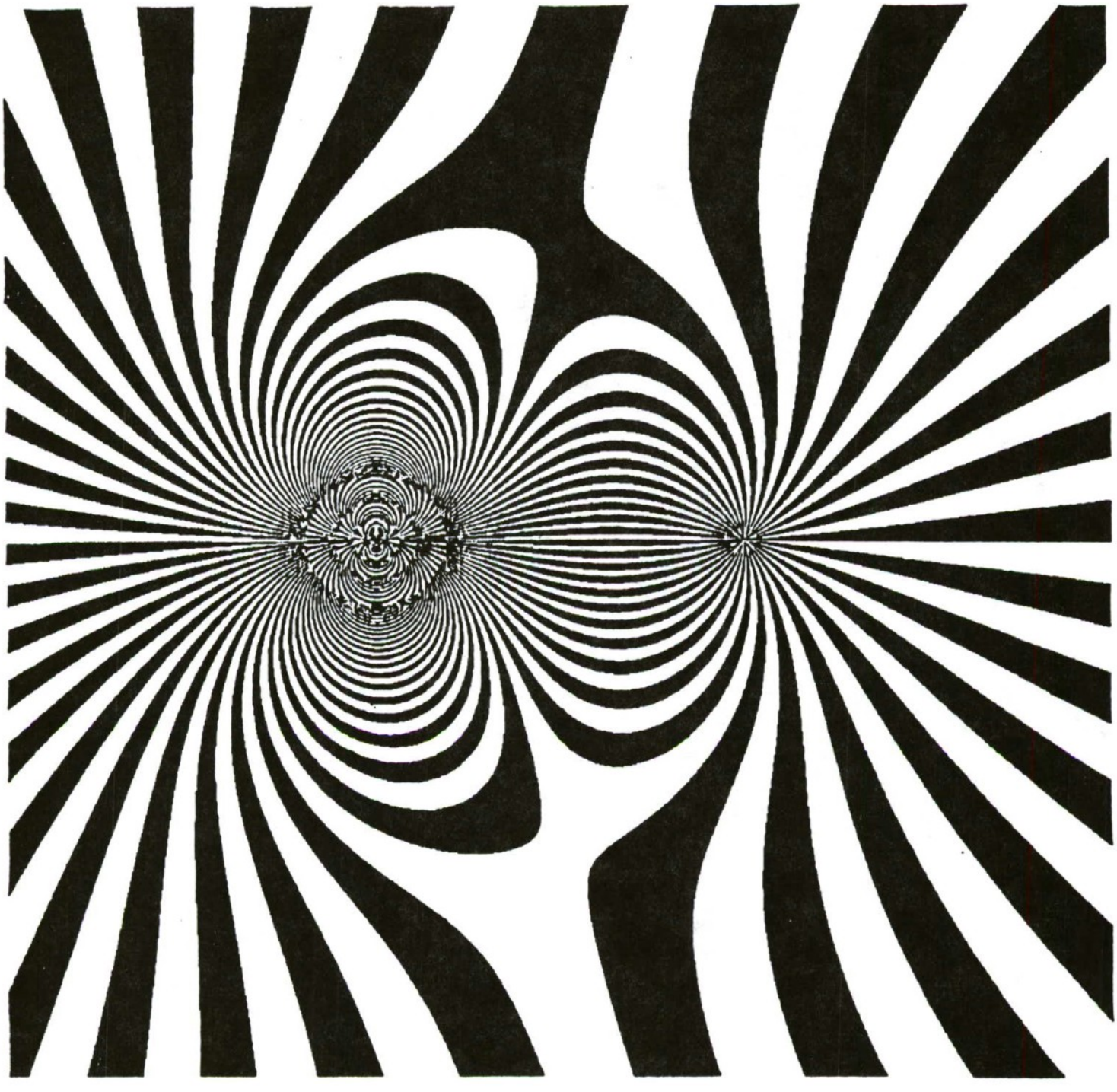


FIG. 7b. PHASE PORTRAIT OF FLOW ON PLANE

INTERMEDIATE RESOLUTION

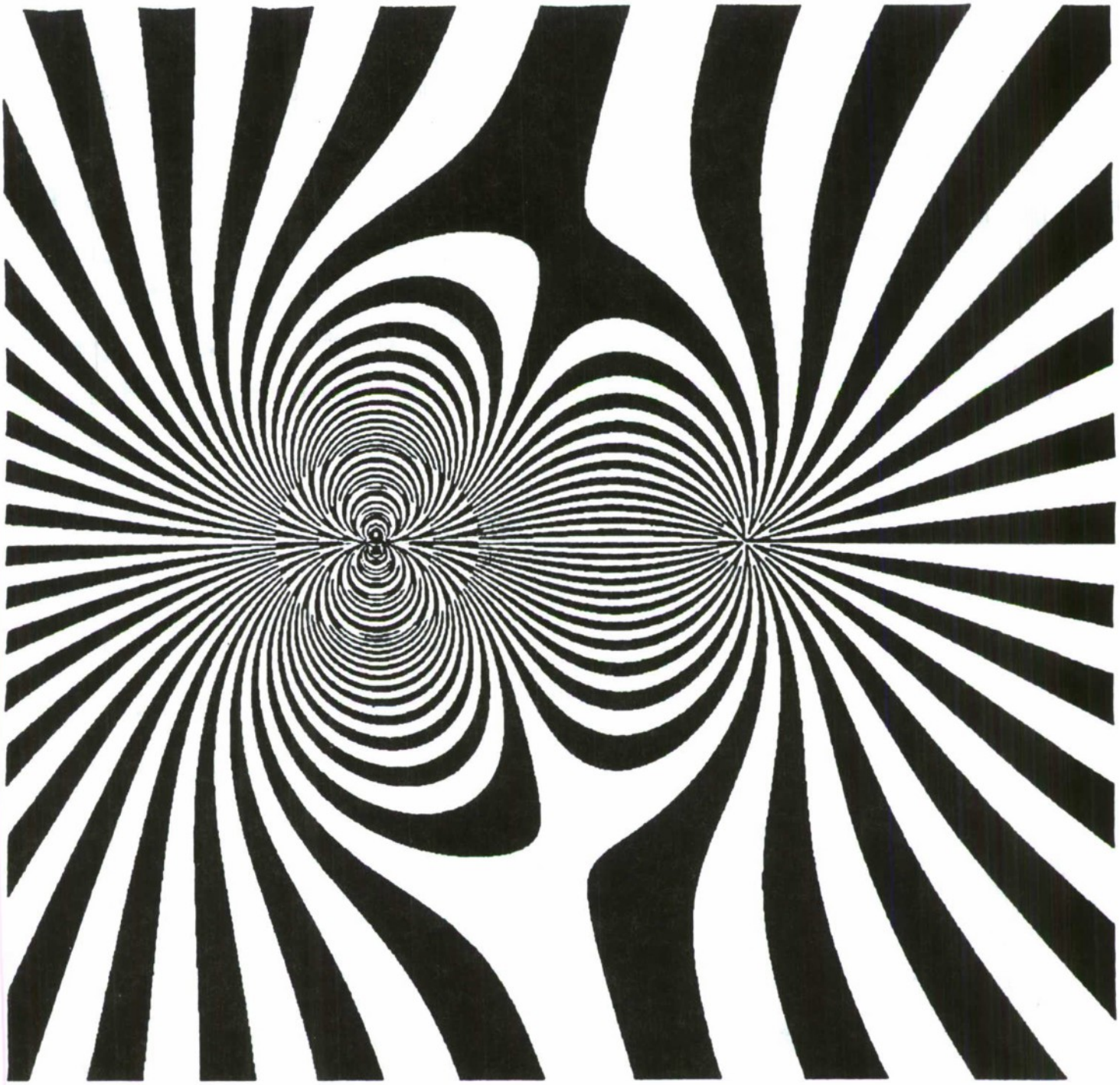


FIG. 7c. PHASE PORTRAIT OF FLOW ON PLANE

INTERMEDIATE RESOLUTION

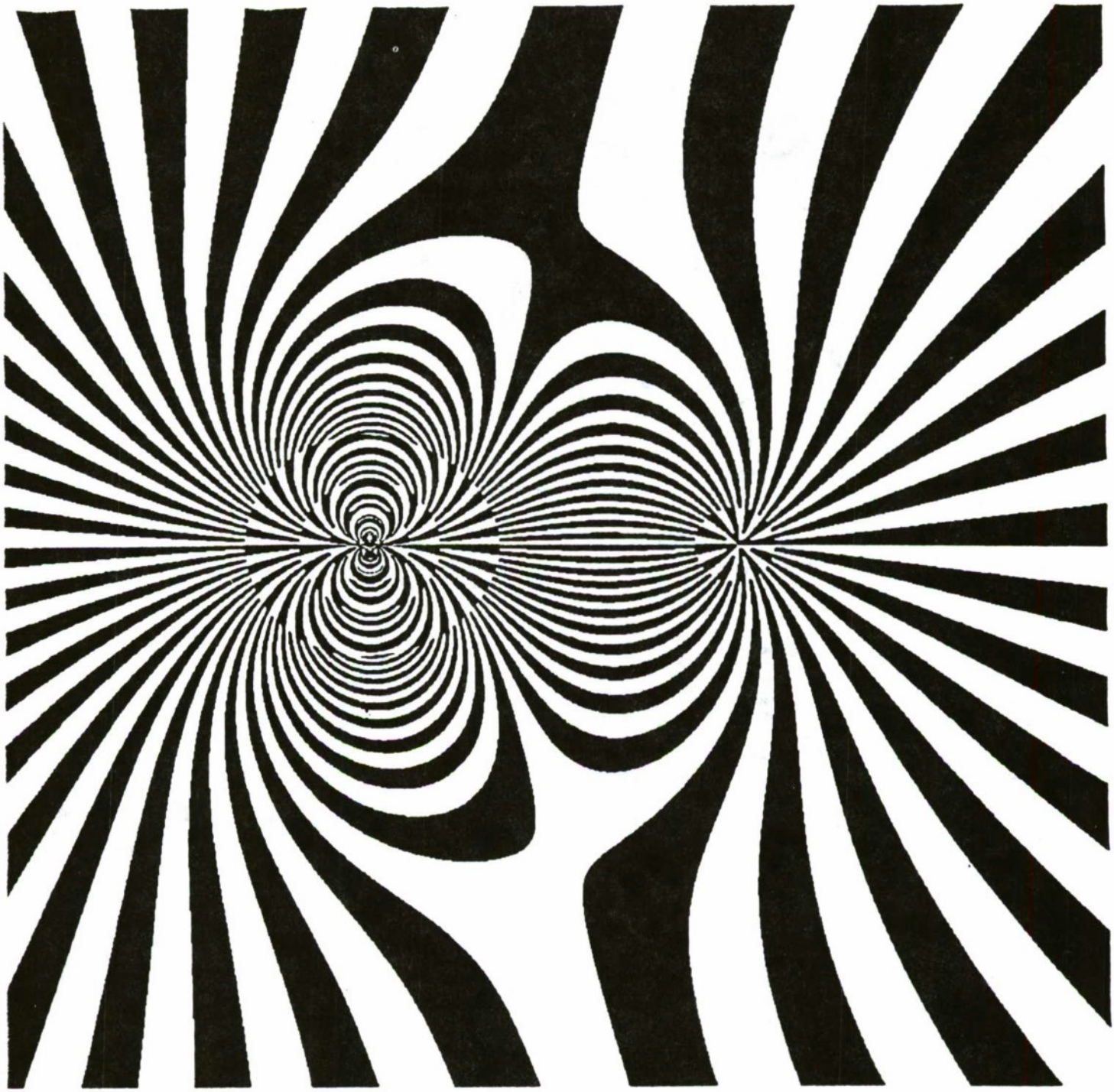


FIG. 7d. PHASE PORTRAIT OF FLOW ON PLANE

FINAL RESOLUTION

A Systems Methodology for Structuring Families of Models at Multiple Levels of Resolution

Bernard P. Zeigler,
AI–Simulation Group
Department of Electrical and Computer Engineering
University of Arizona, Tucson, AZ 85721

1 Introduction

This paper presents a methodology for structuring families of models expressed at multiple levels of resolution and integrating them into a coherent whole. The importance of such structuring for model reusability and interoperability has been well discussed by Davis [5]. This *multiresolution methodology* is based on a research in modelling and simulation methodology that has been in process since the early 70's and which has produced a framework and supporting software environment to manage the complexity of large system modelling and simulation. In what follows, we shall first review the basic framework and concepts underlying formal approaches to multiresolution model construction. This is followed by a discussion of its application to combat modelling (more specifically, to an interpretation of a study let by Hillestad) and conclusions that are drawn from it.

2 Review of the SES/MB Framework

The *Systems Entity Structure/Model Base (SES/MB)* framework has been shown to provide a workable foundation for model base management in advanced simulation environments and workbenches [14]. Such management facilities aim to to provide a sharable repository of models and a means of assisting users to synthesize models to satisfy the objectives of the current study. Advanced simulation environments that employ techniques from artificial intelligence and software engineering are discussed in References [1,2,6,15,18,25,26,27,31]. The system entity structure (SES) is a declarative knowledge representation scheme [3,16,26,34,35,36,37,41] that characterizes the structure of a family of models in terms of decompositions, component taxonomies, and coupling specifications and constraints. The *model base* contains models which are procedural in character, expressed in dynamic and symbolic formalisms. SES-based model base management has been implemented in the DEVS-Scheme simulation environment [15,16,33,37,38]. The environment supports building models in a hierarchical, modular manner, a systems-oriented approach not possible in conventional simulation languages [36,39]. Moreover, the environment supports hierarchical structuring of a family of models, pruning the structure to a reduced version, and transforming the latter to a simulation model by synthesizing component models in the model base [16,42].

One application of the SES/MB framework is to the design of systems [4,26,29]. Here the SES serves as a compact knowledge representation scheme of organizing and generating the possible configurations of a system to be designed. To generate a candidate design pruning reduces the SES to a so-called *pruned entity structure* (PES). Such structures are derived from the governing structure by a process of selecting from alternatives where ever such choices are presented. Not all choices may be selected independently. Once some alternatives are chosen, some options

are closed and others are enabled. Moreover, rules may be associated with the entity structure to further reduce the set of feasible configurations.

As shown in Figure 1, pruned entity structures (PES) are stored along with the SES in files forming the *entity structure base*. Hierarchical simulation models may be constructed by applying the *transform* function to pruned entity structures in working memory. As it traverses the pruned entity structure, *transform* calls upon a retrieval process to search for a model of the current entity. If one is found, it is used and transformation of the entity subtree is aborted. *Retrieve* looks for a model first in working memory. If no model is found in working memory, the *retrieve* procedure searches through model definition files, and finally, provided that the entity is a leaf, in pruned entity structure files. A new incarnation of the *transform* process is spawned to construct the leaf model in the last case. Once this construction is complete, the main *transform* process is resumed. The result of a transformation is a model expressed in an underlying simulation language such as DEVS-Scheme [4,40,41] which is ready to be simulated and evaluated relative to the modeler's objectives.

Reusability of models has been recognized as an essential feature of a model management system [1,14]. The fact that the *transform* process can look for previously developed pruned entity structures, in addition to basic model files, has an important consequence for reusability. For now the pruned structures themselves should be available to be placed into larger structures. That is, the modeler should not have to redo the pruning steps that specified a useful simulation model – these are embodied in the pruned entity structure, and should therefore be reusable simply by accessing the PES.

Recent work [42] has developed a mechanism for "cataloguing" PESs to facilitate their retrieval and thereby achieve a high degree of reusability. Additional mechanisms are also necessary to achieve model base coherence and evolvabil-

ity. These properties are necessary since models may be expressed in a variety of formalisms at a variety of levels of abstraction. *Model base coherence* is the ability to maintain consistency of models with each other when modifications are made or new models are added to the model base. Model base coherence is necessary for *evolvability*, the ability to conveniently extend and modify the model base. Our approach to achieving coherence and evolvability requires that there be a systematic way of recognizing when models of the same underlying entity are being referenced. Organizing models by the entity they model, rather than the context they are used in, facilitates such coherence. Partitioning of a large SES into smaller chuncks supports such entity-based organization. Models of an entity may be related by abstraction relationships so that when one is changed others must be amended to retain consistency. With entity-based partitioning, all models of an entity are readily recognized and interrelated.

3 Combat Modelling Example

Combat modelling provides an example of a Model Base/SES domain application. Our goal is to show how the Model Base/SES methodology can

- organize the models underlying a simulation written in a conventional object-oriented simulation language into a coherent set in which the interconnectivity and commonality of components are explicitly represented
- exploit this organization to generate an unlimited number of meaningful reconfigurations of the identified components which support such important capabilities as bottom-up verification and validation and performance evaluation

The partitioned SES in Figure /refcomb-ses represents the family of models identifiable in the simulation studies of Hillestad et. al. []. There are four com-

ponent SES's for COMBAT, ENGAGEMENT, TANK and EXPERIMENTAL-FRAME, respectively. Graphical conventions used in these diagrams are explicated in Figure 3. To begin, let us consider the TANK SES in Figure 2. Pruning this SES, selecting the T60 entity in the *weapon-type* specialization, results in the pruned entity structure (PES) shown in Figure 4. Also shown in this figure are some of the couplings inherited (and appropriately specialized) from the TANK SES. Such couplings provide the message pathways among the component model objects, T60_WEAPON, MOTION-SYSTEM, and LEADER as well as between the enclosing TANK model object and these components. The TANK PES is tranformed into a simulation model whose structure is illustrated in Figure 5. More of the couplings are shown in this diagram.

Atomic models are expressed in the DEVS formalism and are graphically depicted as in Figure 6. The atomic model components of TANK are shown in Figures 7 a),b) and c). For example, WEAPON has two input ports, FIRE and STOP, and two output ports, SEND-BE-KILLED and ROUND-DONE. An external event on port FIRE carries with it target information. It causes a change in phase from PASSIVE to ACTIVE. An external event on port STOP causes the reverse change in phase. Once in phase ACTIVE, the model rests there for a duration given by *firing-time*. It then places an output on port ROUND-DONE and transits to phase FIRE. Holding in the latter phase for a time *delay*, it emits an output containing the target designation on port SEND-BE-KILLED, before returning to phase PASSIVE. As shown in Figure 5, when coupled in TANK, the ouput on port ROUND-DONE of WEAPON is sent to the same port in LEADER. On the other hand, the port SEND-BE-KILLED appears as on external output port for model TANK.

Note that all model are standalone objects and can be loaded, executed and tested independently of each other. For example, the target object that is to be

the recipient of the SEND-BE-KILLED message need not exist in memory while testing the TANK model. It is only in the more inclusive coupling illustrated in Figures 8 and 9 that an actual output is transmitted from the TANK to the target following the coupling pathway.

In the PES shown in Figure 8 the multiple entities COMPANYS and TANKS have each been replaced by numbers of instances of types COMPANY and TANK as specified by the modeller in the pruning process. In this instance, for example, the RED-SIDE has been given 9 instances of COMPANY, each of which contains 11 instances of TANK. Note that the instances of TANK may be pruned so that they all have the same structure, such as the T60_WEAPON selection, or have different structure. Thus, pruning presents a powerful means by which alternative configurations of a simulation model can readily be specified.

4 Multiresolution Modeling Concepts

Terminological confusion is no stranger to the modelling and simulation community – no more so that in relation to model resolution issues. Although general agreement may be a long way off, it is worthwhile to propose some terminoligical standards– to which we adhere at least in this paper. Figure 10 suggests that models may be ordered along different dimensions: **resolution** and **complexity**, and may be scaled in these dimensions by processes such as **abstraction**, **aggregation**, and **simplification**. Although these dimensions refer to distinct ordering principles, they are often correlated. This correlation leads to confounding one for the other.

Resolution refers to the degree to which objects and processes in the real world are resolvable in the model. Pragmatically, the resolution level is synonymous with the level of detail. Thus, a high resolution model represents its counterpart real object in great detail. The analogy is to magnifying glass – the greater its

mangification or resolution, the greater the detail that can be perceived in the real system. We sometimes the use the terms *base* and *lumped* to refer to a pair of models, related in the homomorphic manner to be discussed, where the base model is at a higher resolution level than the lumped model counterpart.

Complexity refers to time and space resources needed to simulate a model. Thus, a complex model (one that is high on the complexitity scale) requires a long execution time or a large memory, depending on the metric of interest. Although often related in monotonic manner, resolution and complexity are independent dimensions. For example suppose that in the (often tacitly accepted) context of serial computation a high resolution model executes much more slowly than a low resolution counterpart. However, in a massively parallel simulation the former actually may run faster than the latter due to better ability to distribute component computations.

Aggregation refers to replacing a group of variable by an aggregate quantity such as their sum. Disaggregation is the inverse process of breaking out a variable into a group of other variables. Thus aggregation is a model manipulation process that maps a model from one point on the resolution scale to a lower one. An aggregated, lower resolution, lumped model is often also less complex in the sense of time/space resource requirements than the original base model, but, as indicated, not necessarily so.

Abstraction is a general model mapping process that takes a base model into a lumped mode where the models may be expressed in different formalisms. Abstraction subsumes aggregation as a special case.

Simplification refers to the process reducing the the complexity a base model to a lower level. As indicated, simplification often results from abstraction or aggregation process, but not necessarily so.

Multifacetted modeling methodology recognizes the need for a multiplicity

of abstractions to support system objectives[37]. These partial models, being oriented to specific objectives, are more computationally tractable, more understandable, and easier to develop than comprehensive multipurpose models[25,36,37,38]. Also, the multiplicity of abstracted models may provide an evolutionary path for the modeling process if models evolves from one to another as more domain knowledge is gained. Fishwick [7,8,9] has extended process abstraction concepts and implemented a simulation system that is able to switch between levels within a simulation run. Although the need for multiple levels of abstraction has been recognized in mainstream AI, there has been little consideration of the importance the morphism concept to this issue. Much recent research has focused on derivation of abstract knowledge from deep domain knowledge for problem solving and schemes for device or process analysis at multiple levels of resolution (abstraction) [16,22].

However, these approaches lack criteria for valid abstraction. Such criteria are imposed by an explicit statement of the kind of relation to be preserved relative to the objectives at hand. Sevinc[28] developed a means to support automation of simplification of discrete event models. The simplification tools use observation data from simulation runs of the original model to generate the lumped model that preserves validity under experimental conditions of interest. The simplification approach, based on **homomorphism**, is intended to provide faster running lumped models rather than more understandable models. However, abstraction can provide both. Also, this empirical approach does not address the problem of maintaining consistency of related models. Sevinc and Foo[30] proposed a state classification algorithm to group base model states into classes which in turn act as states for the lumped model. Conceptually this algorithm can be used to generate lumped models at different levels of resolution as desired. However, it is critical that the modeler must choose an appropriate measure of proximity between states

to serve as the criterion for state clustering. Sevinc and Foo recognized the need for facilities for discovering such proximity measures, but they did not provide an approach to this issue.

5 Morphism and Model Base Concepts

5.1 Levels of System Specification

The concept of homomorphism provides a well-founded criterion for valid abstraction. To do the homomorphism concept justice it is necessary to sketch the basics of the system theory concepts on which it is bases. Considering a real system as a black box, there is a *hierarchy of levels* at which models may be constructed ranging from the purely behavioral — in which the model claims to represent only the observed input/output behavior of the system, up the strongly structural — in which much is claimed about preserving the structure of the system. Simulation is recognized as the process of moving in the structure-to-behavior direction. Thus, simulation models are usually placed at the higher levels of structure and they embody many postulated mechanisms to generate the behavior of interest. In contrast, behavior descriptions obtained by curve fitting, for example, represent lowest level models.

As illustrated in Figure 11 corresponding to each level of system specification, is a morphism that encodes structure and behavior equivalence of a pair of models at that level. The existence of a morphism at one level implies the existence of a corresponding morphism at the level below. This expresses the general notion that models that have the same structure must also exhibit the same behavior. However, the converse is false:— behavior preservation does **not** imply structure preservation. This is so since many structures may exhibit the same behavior.

The concept of structural homomorphism between models is illustrated in Figure 12. Generally, a state correspondence between base and lumped models

must be preserved under transitions and outputs. Morphisms differ in such details as the nature of the state correspondence and the lengths of micro-state transition sequences corresponding to macro-state transitions [22]. Informally, a *generalized homomorphism* (or *system morphism)* involves a triple (h_i, h_s, h_o) of mappings of the input segment, state, and output sets of the base model to the respective counterparts of the lumped model, which together preserve the transition and output functions of the models. The preservation requirements can be schematized by the so-called commutative diagrams shown in Figure 13. In Figure 13a, we start the base model in one of the states q_0 in a restricted subset of its state set. The mapping h_s yields a corresponding state in the lumped model $q_0' = h_s(q_0)$. We then inject an input segment ω into the base model and its corresponding version $h_i(\omega)$ into the lumped model sending them to the states q_n and q_n', respectively. These states also correspond under h_s. In Figure 13b, when the base model is in state q and the lumped model is in corresponding state $q' = h_s(q)$, the output values observed in these states are y and y', respectively; the value y' is also obtained by decoding y with mapping h_o, that is $y' = h_o(y)$. In applications we are often interested in homomorphisms between systems described over the same observational base, i.e., the same time base, input value set, and output value set. In such cases, h_i and h_o are identity mappings and h_s is said to be a homomorphism. See [19,36] for more details.

An important concept that underlies morphism at all levels is that of *experimental frame.* The experimental frame characterizes the objectives of a model construction process by providing the input/output and state space constraints corresponding to these objectives. See [36,37] for a more complete discussion.

6 Multiresolution Family of Models: Case Study

Having reviewed the general background on multiresolution modelling methodology, we turn to the combat modelling example to illustrate its application. To demonstrate the methodology we

- characterize the questions being addressed in the simulation as experimental frames
- identify some of the primary forms of model abstraction employed in the models and explicitly represented these as morphism objects attached to the SES.

The main experimental frame, shown in Figure /refcomb-ses, contains 1) a GENERATOR to activate the ENGAGEMENT model by placing an output on port START; 2) a TRANSUCER to gather statistics relating to attrition from the model (this is the equivalent of the BATTLE object in the Hillestad simulation); andd 3) an ACCEPTOR, which stops a simulation run when there are no live tanks on either side.

6.1 Model Abstractions and Representation

Some types of morphisms identified in the combat model family are illustrated in Figure 14 as subclasses of a general class *morphisms*. The most elementary is called *identify-morphisms* where variables are identified in two models such that the values assigned to them must be equal. The utility of such a morphism object is that it can be triggered whenever there is a change in value assigned to a variable in one model to assure that its counterpart variable in the second model is updated accordingly. With the identify-morphism in effect, new parameter values assigned

to the corresponding models are automatically propagated to related models. For example, to start in states in which different numbers of tanks can be placed on the FLOT the number assigned to the FLOT state variable of COMMANDER in Figure 8 must correspond to the number of TANKS actually placed on in the forward area to initialize the simulation run.

More generally in a morphism class the mapping between sets of variables need not be a one to one correspondence. For example, consider *statistical-morphisms* which maps from the disaggregated event-generation process to the aggregated one. Here the mean value parameters of the individual firing interarrival processes are summed together to yield the mean value parameter of the aggregated process firing interarrival. If the individual processes are independent and identically distributed Poisson processes then a Poisson process with the summed mean will yield the same distribution as the summed disaggregated processes. This is an example of an error-free morphism. However, for other types of distribution, or when the required independence assumption does not hold, the distribution generated by the aggregated model may deviate from that produced by the original. The morphism object contains slots for explicitly recording the experimental frame in which the high and low resolution models are expected to concord and the assumptions under which this agreement is guaranteed to be exact. Also, if known, a formula can be given for the expected error if the assumptions are violated in an actual experimental frame being employed to exercise the low resolution model.

Yet another example of a morphism classes is *discrete-representation-morphisms.* Here discrete event representation of MOTION-SYSTEM can be used in place of an original continuous formulation. The morphism here requires that the discrete event model respond in the same way as the continuous model to inputs (on ports FIGHT, STOP and CURRENT-POS? shown Figure 7 and generate the same outputs when crossing terrain cell boundaries at the same times.

To verify discrete event representation morphisms properly requires a combined discrete-event/continuous model simulation capability that includes the DEVS formalism as a special case [24,32]

As shown in Figure 14, morphism classes in general have the ability to check whether the relationships expressed in a given morphism hold. However, the class *discrete-representation-morphisms* has an additional method that actually generates a homomorphic low resolution model when given a high resolution model as input.

The class *attrition-morphisms* capture the attrition preserving aggregation process underlying that is the focus of study by Hillestad et. al. and MSRT-MOTION models. In Figure 2, attrition-morphisms can be attached to the COMPANY entity to relate high and low resolution versions COMPANY models. Such a morphism would contain the state space and observer mappings discussed in Hillestad et. al. and would be able to check whether these mappings are preserved throughout a simulated battle. In an error free morphism this would always be the case. However if substantial error is introduced, the mappings might be paraterized so that different parameter values are employed during different "phases" of the battle, as discussed by Hillestad et. al. This actually represents a weaker form of morphism which must have a means of recognizing the on set of such phases. Such information might only be available in the high resolution model. It can be represented in the morphism object by employing the experimental frame slot to characterize the domain of validity of a partial low resolution model.

This situation is well described by the levels of structure and behavior preservation illustrated in Figure 11. In a structure preserving morphism one the high and low resolution models are set into corresponding states this corresponds remains valid as time advances and behavior is preserved everywhere. In a behavior preserving morphism, behaviors of high and low resolution models can be matched

up, but (unless the morphism is also a structural one) this matching can not be uniform across the high resolution state space. Rather different pairings must be associated with different subsets, or "phases" of the space.

7 Organization of the Models into a Coherent Family

Multiresolution structuring of the combat model example contrasts from the original formulation in significant ways. The latter is a monolithic package albeit expressed in object-oriented language. Component models neither can standalone nor is the interconnectivity apparent by inspection. In contrast,the restructuring is characterized by a model base of modular, reusable components. The coupling and specialization possibilities are explicitly represented in a declarative knowledge representation scheme (the SES) that is open to user inspection and can be reasoned with by the software itself. An example of such knowledge-based processing is to check and propagate morphism validity. In the original, calibration data from a high resolution model must be manually fed into the initialization space of a low resolution model In our restructuring, the transfer of such data is automatically executed by the activation of the appropriate morphism instance, e.g., of *attrition-morphisms* linking corresponding configurations of high and low resolution company models.

8 Conclusion

The systems-theory based methodology described here has been shown to provide a feasible approach to organizing, interoperating, and reusing, models of the same underlying system but expressed at multiple levels of resolution. Such a methodology provides a sound basis for future construction of hierarchical model suites. It also suggests the kinds of considerations that must go into re-engineering an

existing hetereogeneous collecion of models so that they can be interoperated. However, such collections most often contain models developed at different times for different purposes with little in the way of the uniformity imposed on model construction by the presented methodology. Nevertheless, because of its fundamental systems theory foundation, we claim that if it is not feasible to re-engineer a set of models following the tenets of this methodology, then it is not possible by any alternative (practical) means to achieve this result.

9 References

[1] O. Balci, "Requirements for Model Development Environments," Com- puters and Operations Research Vol. 13, No. I , pp 53-67, 1986 .

[2]D.W. Balmer and P.J. Paul , "CASM - The Right Environment for Simulation," J. Operational Research Society, No. 37, pp. 443-452,1986.

[3]D. Belogus, "Multifaceted Modelling and Simulation: A Software Engineering Implementation," Ph.D. Dissertation. Weizmann Inst. of Science, Rehovot, Israel, 1985.

[4]S.D. Chi, "Modelling and Simulation for High Autonomy Systems," Ph.D. Dissertation, Dept. of Electrical and Computer Engr., University of Arizona, Tucson, AZ, 1991.

[5]P.K. Davis, "Variable Resolution Modeling and Families of Models", Proc. of Variable Resolution Modeling Symposium, May 1992, Herdon, Va.

[6]M.S. Elzas, "The Applicability of Artificial Intelligence Techniques to Knowledge Representation in Modelling and Simulation," In: Modelling and Simulation Methodology in the Artificial Intelligence Era, (eds: M.S. Elzas, T.I. ren, and B.P. Zeigler) North Holland, Amsterdam, pp. 19–40, 1986.

[7]P. A. Fishwick, "The Role of Process Abstraction in Simulation," *IEEE Trans. Syst. Man and Cybern.*, vol. SMC-18, no. 1, pp. 18–39, 1988 .

[8]P A. Fishwick, "Abstraction Level Traversal in n Hierarchical Modeling," I

Modelling and Simulation : Methodology: Knowledge Systems Paradigms (eds. B. P. , Zeigler, M. Elzas, and T. Ören), Elsevier, North . Holland, 1989 pp. 393–429

[9]P. A. Fishwick, "Process Abstraction in Simulation n Modeling," I *Artificial Intelligence, Simulation, : and Modelling* (eds. L. E. Widman, K. Loparo, and N. Nielsen), John Wiley and Sons, 1989, pp. 93–131 .

[10] J.B. Gilmer and I. Kameny , "SIMTECH 97: Report of the Workbench Working Group," Phalanx, Vol 22. No. 4, 1989.

[11]J.O. Henriksen, "The Integrated Simulation Environment," Operations Research, Vol. 31 , pp. 1053-1073, 1983.

[12]Hillstad, R. "Aggregation and Lanchester's Equations",

[13]Hillstad, R. "Simulation Case Study"

[16]L. Jokowicz, "Simplification and Abstraction of Kinematic Behaviors," *Proceedings IJCAI-89*, Detroit, MI, August 1989, pp. 1337–1342 A.

[16]Kim, T.G., C. Lee, E. R. Christensen, and B.P. Zeigler, System Entity Structuring and Model Base Management, IEEE Trans. Sys. Man. Cyber. 1991

[17]T.G. Kim and B.P. Zeigler , "The DEVS Formalism: Hierarchical, Modular System Specifica- tion in an Object-Oriented Framework," In Proc. of 1987 Winter Simulation Conf., Atlanta, GA, pp. 559-566, 1987.

[18]P. Klahr, "Expressibility in ROSS, an Object-Oriented Simulation Sys- tem", In: Artificial Intelligence in Simulation (eds: G. C. Vansteenkiste, E.J.H. Kerckhoffs , and B.P. Zeigler), SCS Publications, San Diego, CA, 1986.

[19]C. J. Luh, "Abstraction Morphisms for High Autonomy Systems, Ph.D. Dissertation, Dept. of Electrical and Computer Engr., University of Arizona, Tucson, AZ, 1991.

[20]K.J. Marray and S.V. Sheppard , "Automatic Synthesis Using Automatic Programming and Expert Systems Techniques Toward Simulation Modeling," Proc. Winter Sim. Conf., pp. 534-543, 1987 .

[21]S. S. Murthy, "Qualitative Reasoning at Multiple Resolutions," In *Proceedings AAAI-88*, St. Paul, MN, August 1988, pp. 296–300 .

[22]S. S. Murthy and S. Addanki, "PROMPT: An Innovative n Design Tool," I *Proceedings AAAI-87*, Seattle, Washington, July 47–73.

[23] T.I. Oren, "Bases for Advanced Simulation: Paradigms for the Future," In: Modelling and Simulation Methodology: Knowledge Systems Paradigms (eds: M.S. Elzas, T.I. Oren, and B.P. Zeigler), North Holland Pub. Co., Amsterdam, pp. 29–44, 1989.

[24]H. Praehofer, "System Theoretic Formalisms for Combined Discrete-Continuou System h Simulation," Special Issue on Modelling and Simulation for Hig Autonomy Systems, *Int. J. Gen. Sys.*, vol. 19, no. 3, 1991 .

[25]Y.V. Reddy, M.S. Fox, N. Husain, and M. McRoberts , "The Knowledge-Based Simulation System," IEEE Software, March, pp. 26–37, 1986.

[26] Rozenblit, J.W, Hu, J.F, Kim, T.G. and B.P. Zeigler, (1990) "Knowledge-Based Design and Simulation Environment (KBDSE): Foundational Concepts and Implementation," Journal of Operational Research, 1990.

[27]S. Ruiz-Mier and J. Talavage, "A Hybrid Paradigm for Modeling of Complex Systems," In: Artificial Intelligence, Simulation and Modelling (eds. L.A., Widman, K.A. Loparo, and N.Nielsen), J. Wiley, NY. pp. 381– 395, 1989.

[28]S. Sevinc, "Automation of Simplification in Discrete Event Modelling and Simulation," *Int. J. General Systems*, vol. 18, pp. 125–142, 1990

[29] S. Sevinc, and B.P. Zeigler (1988), "Entity Structure Based Design Methodology: A LAN Protocol Example", *IEEE Transactions on Software Engineering*, Vol. 14, No. 3, March, pp. 375–383.

[30]S. Sevinc and N. Y. Foo, "Discrete Event Model e Simplification Via Stat Classification," *AI and Simulation: Theory and Applications*, SCS Simulation Series, vol. 22, no. 3, April 1990, pp. 211–216 .

[31]T. Thomasma and Ulgen, O. M. , "Hierarchical, Modular Simulation Modelling in Icon-based Simulation Program Generators for Manufactur- ing, " Proc.

Winter Simulation Conf., San Diego, pp. 254–262, 1988.

[32]Q. Wang and F.E. Cellier, "Time Windows: An Approach to d Automate Abstraction of Continuous-Time Models into " Discrete Event Models, Special Issue on Modelling and , Simulation for High Autonomy Systems *Int. J. Gen. Sys.*, vol. 19, no. 3, pp.241–262, 1991.

[33]A.W. Wymore, A Mathematical Theory of Systems Engineering: The Elements, Wiley, NY., 1967.

[34] L.A. Zadeh and C.A. Desoer, Linear System Theory, The State Space Approach, McGraw Hill, NY., 1963.

[35] Zhang, G. and B.P. Zeigler (1989), "The System Entity Structure: Knowledge Representation for Simulation Modeling and Design", , In: *Artificial Intelligence, Simulation and Modelling* (eds.: L.A.

[36]B.P. Zeigler, *Theory of Modelling and Simulation*, Wiley, NY. 1976 (Reissued by Krieger Pub. Co., Malabar, FL. 1985).

[37] B.P. Zeigler, Multifacetted Modelling and Discrete Event Simulation. London, UK and Orlando, FL: Academic Press, 1984.

[38] B.P. Zeigler, "The Aggregation Problem," In: *Systems Analysis and Simulation in Ecology*, vol 4, (ed.: B. C. Patten), Academic Press, NY, 1976, pp. 299–311 .

[39]B. P. Zeigler, "System-theoretic Representation of " Simulation Models, *IIE Trans.*, pp. 19–34, Mar. 1984 .

[40]Zeigler, B.P. and T.G. Kim (1990), "The DEVS-Scheme Modelling and Simulation Environment", In: *Knowledge-Based Simulation: Methodology and Applications*, (eds.: P. Fishwick and R. Modjeski), Springer Verlag (to appear).

[41]B.P. Zeigler, *Object-Oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems*, Academic press, NY).

[42]B. P. Zeigler, C. J. Luh, and T. G. Kim, "Model Base Management for Multifacetted Systems," to appear in *ACM Trans. on Modelling and Computer Simulation*, vol. 1, no. 3, 1992.

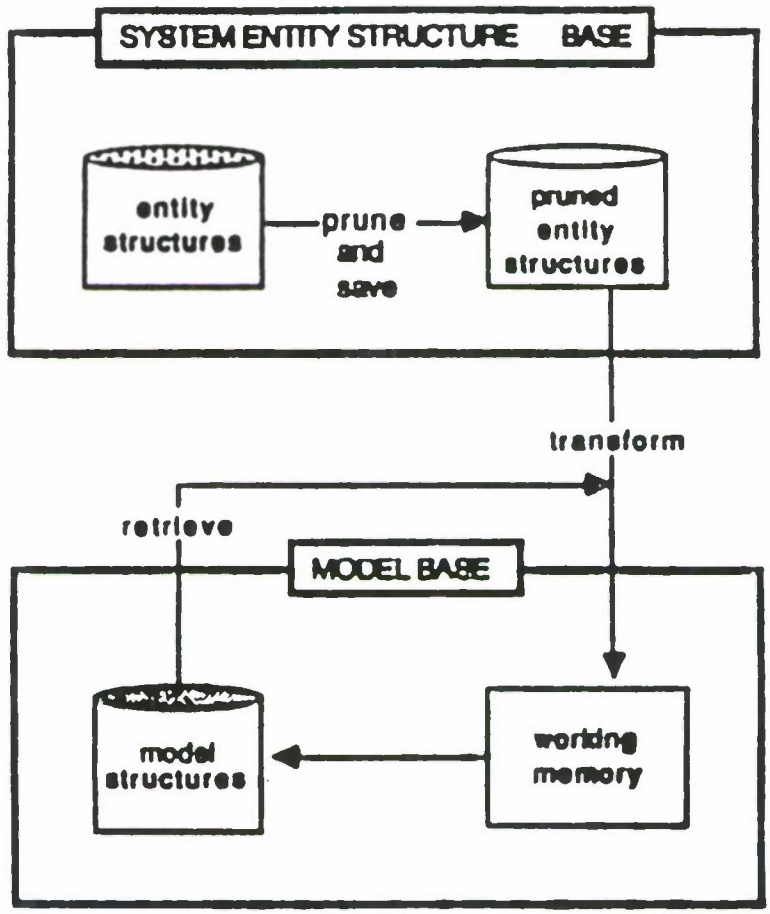
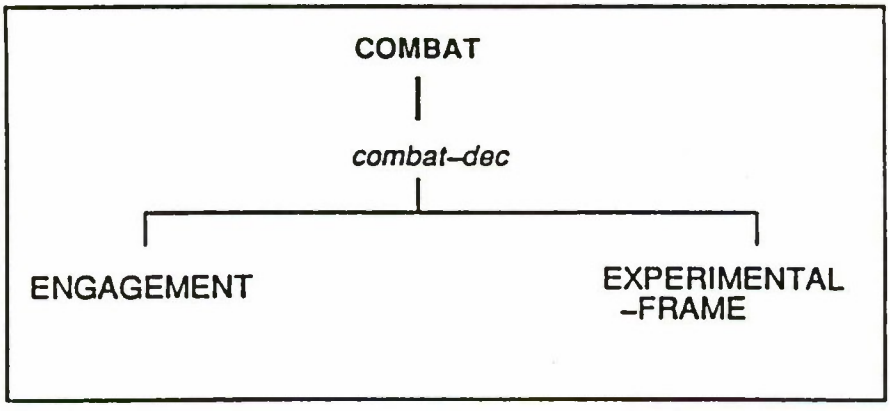
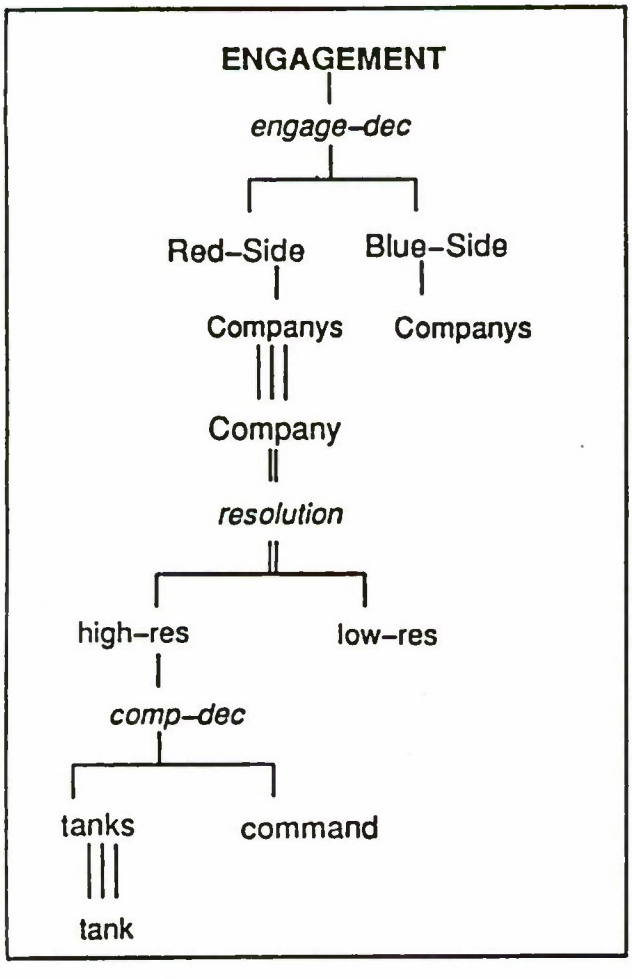
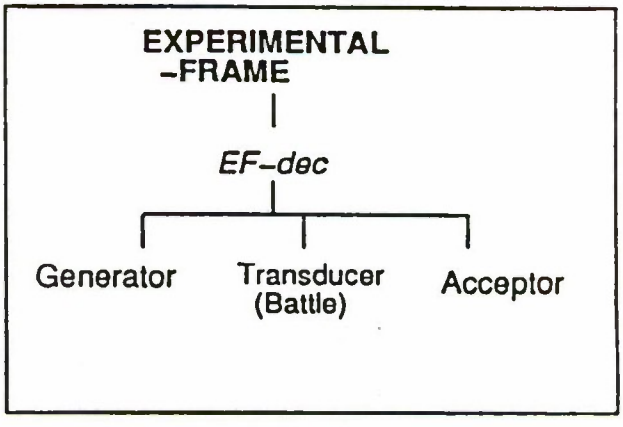
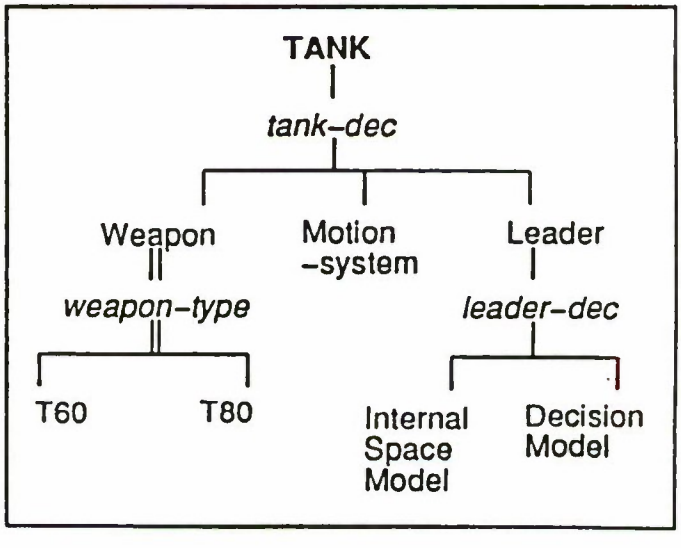


Figure 1: The System Entity Structure/Model Base (SES/MB) Environment.


COMBAT
combat–dec
ENGAGEMENT
EXPERIMENTAL
–FRAME


ENGAGEMENT
engage–dec
Red–Side
Blue–Side
Companys
Companys
Company
resolution
high–res
low–res
comp–dec
tanks
command
tank


EXPERIMENTAL
–FRAME
EF–dec
Generator
Transducer
(Battle)
Acceptor


TANK
tank–dec
Weapon
Motion
–system
Leader
weapon–type
leader–dec
T60
T80
Internal
Space
Model
Decision
Model

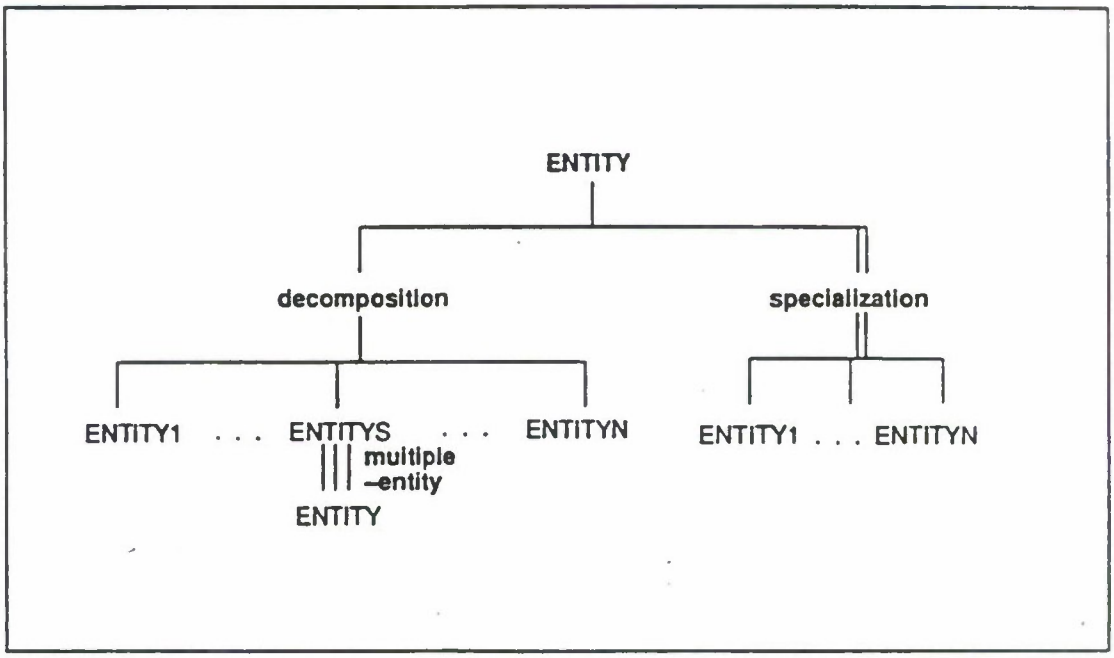


Figure 3: SES Representation Conventions

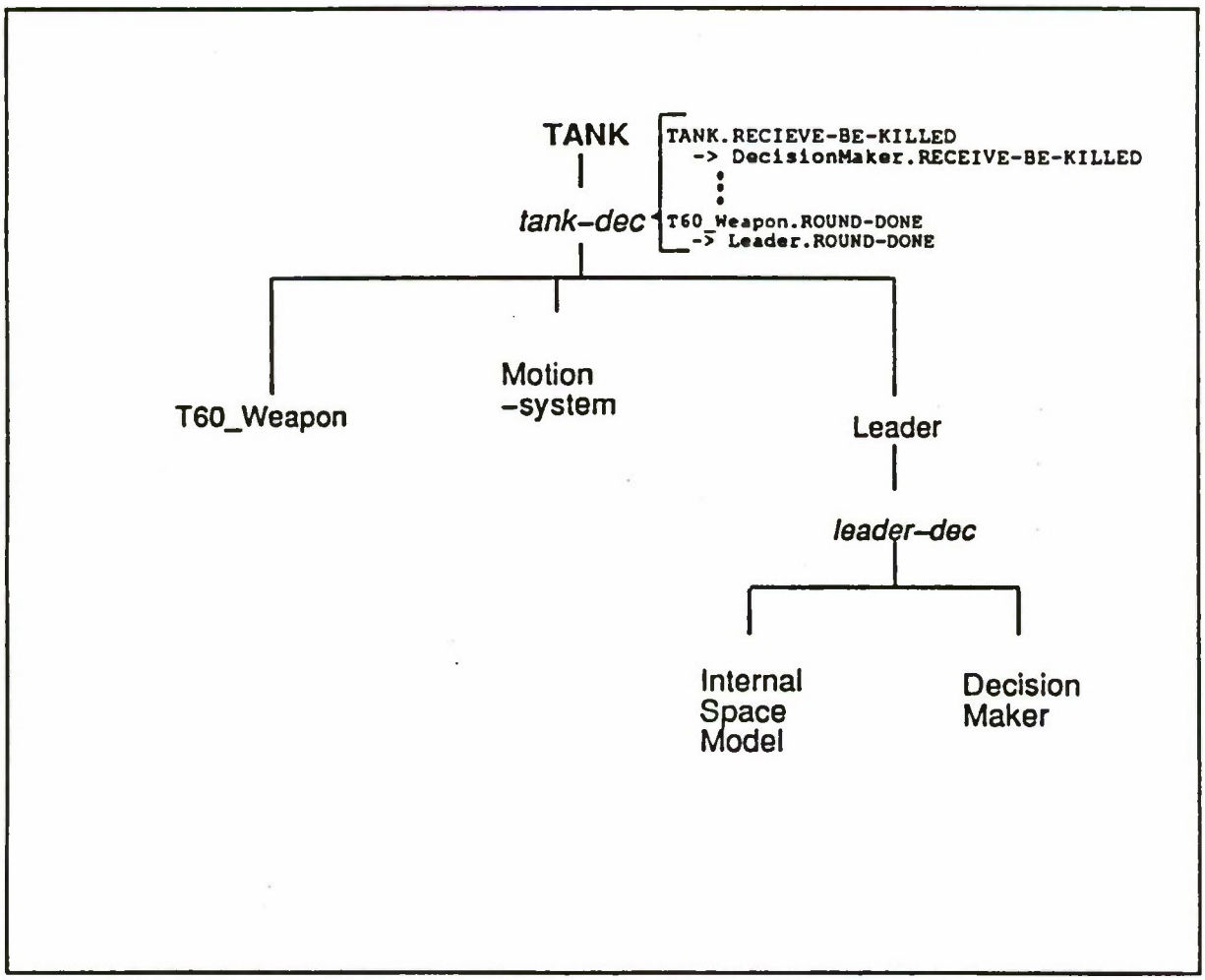


Figure 4: PES of TANK

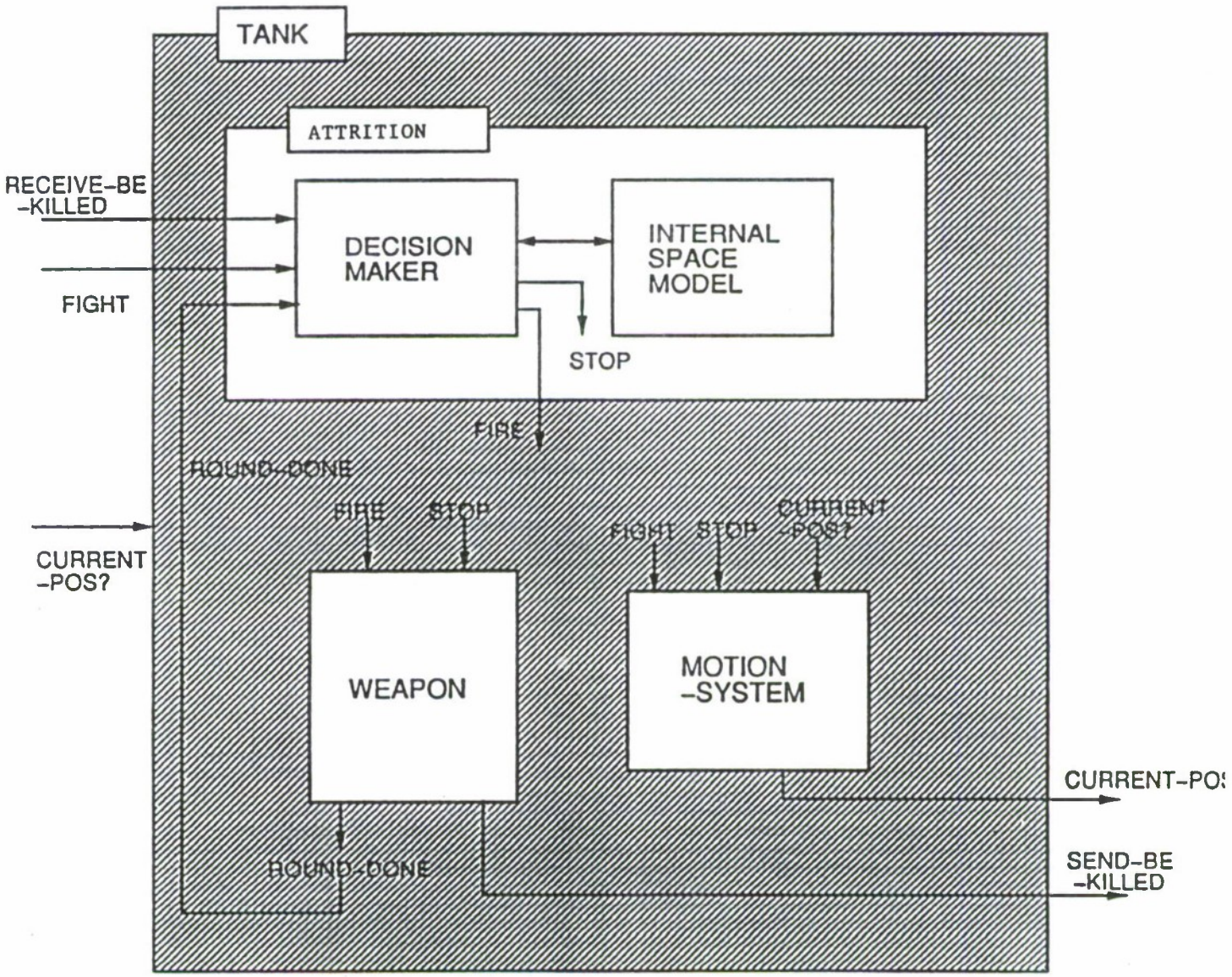


Figure 5: Transformed Hierarchical Model of TANK

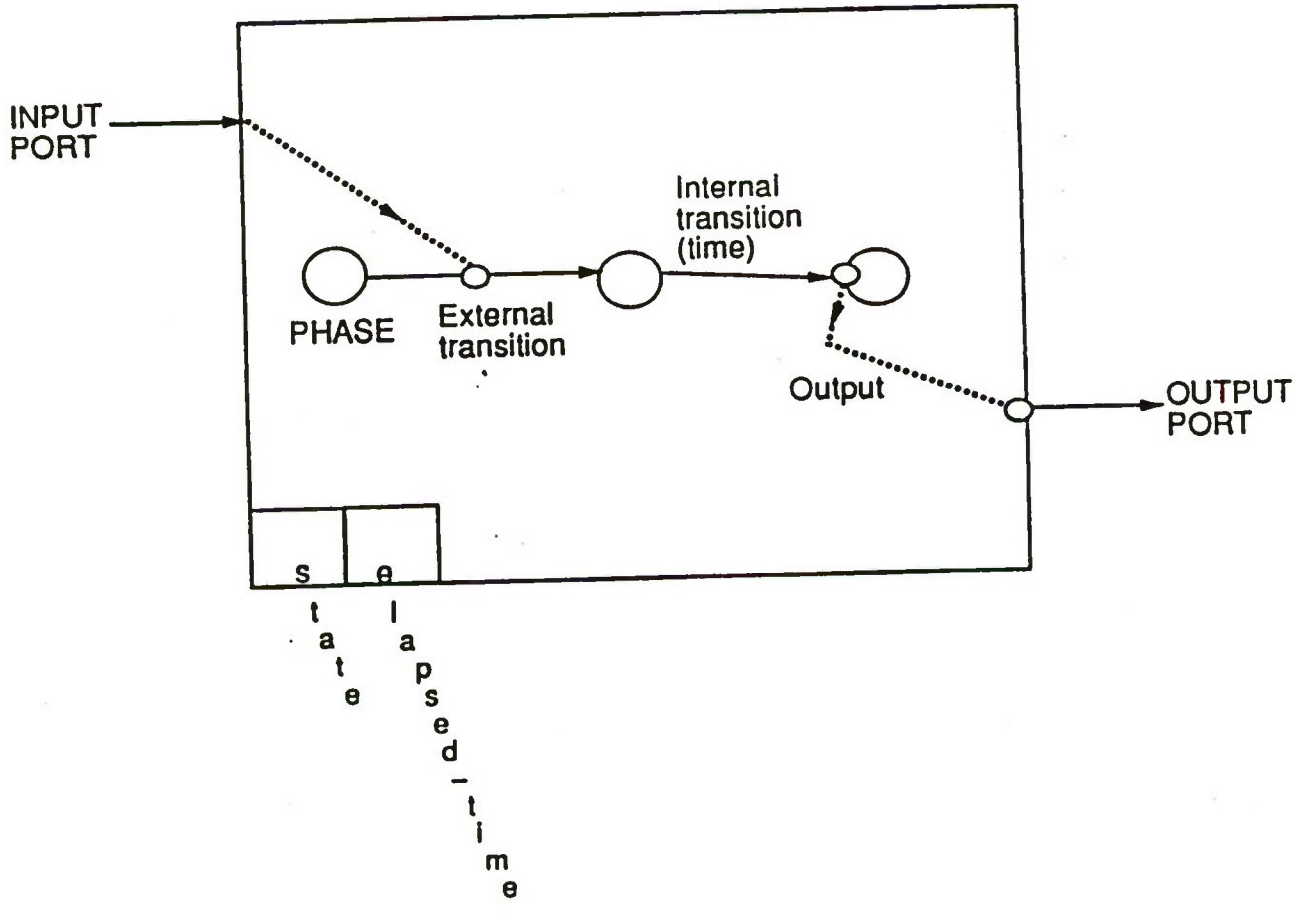


Figure 6: Graphical Conventions for Atomic Models

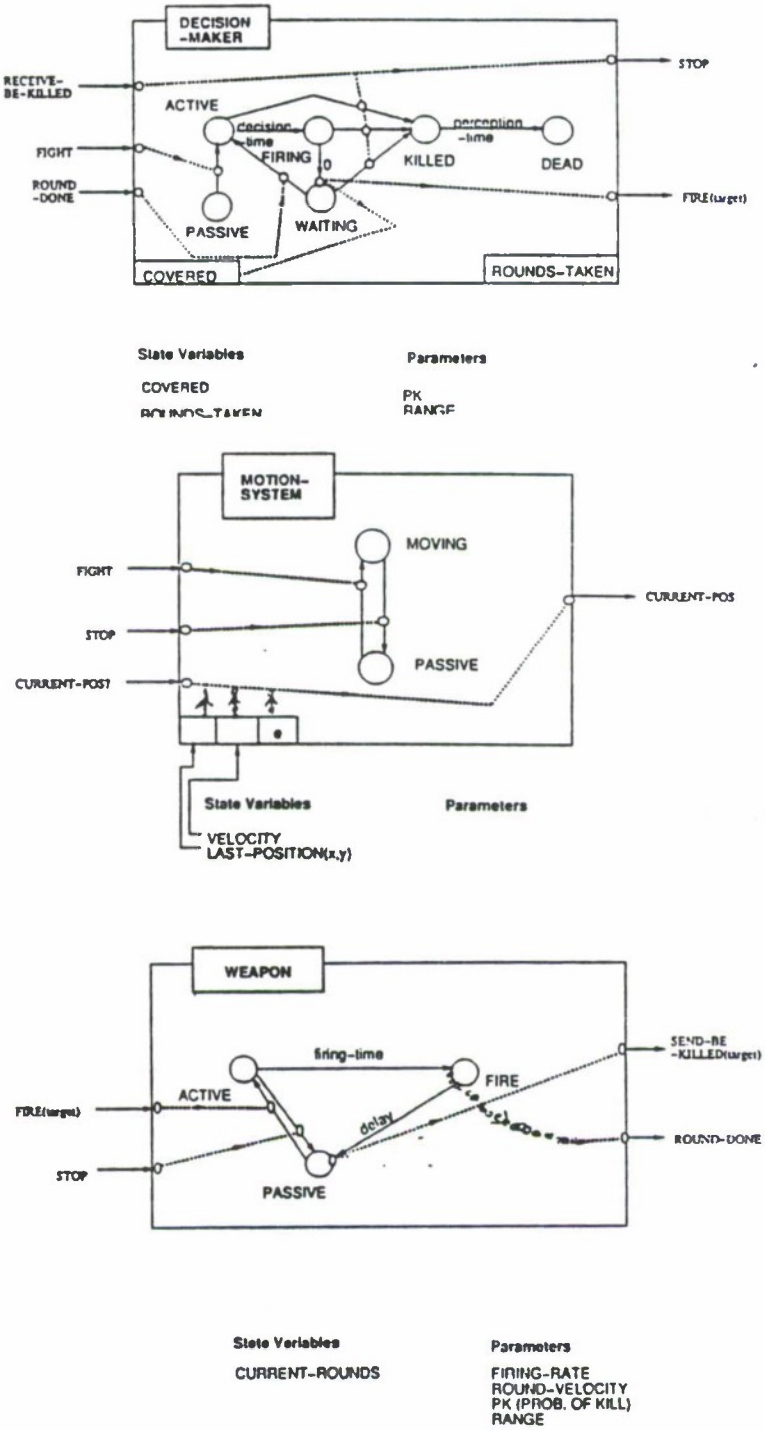


Figure 7: Atomic Model Components of Tank

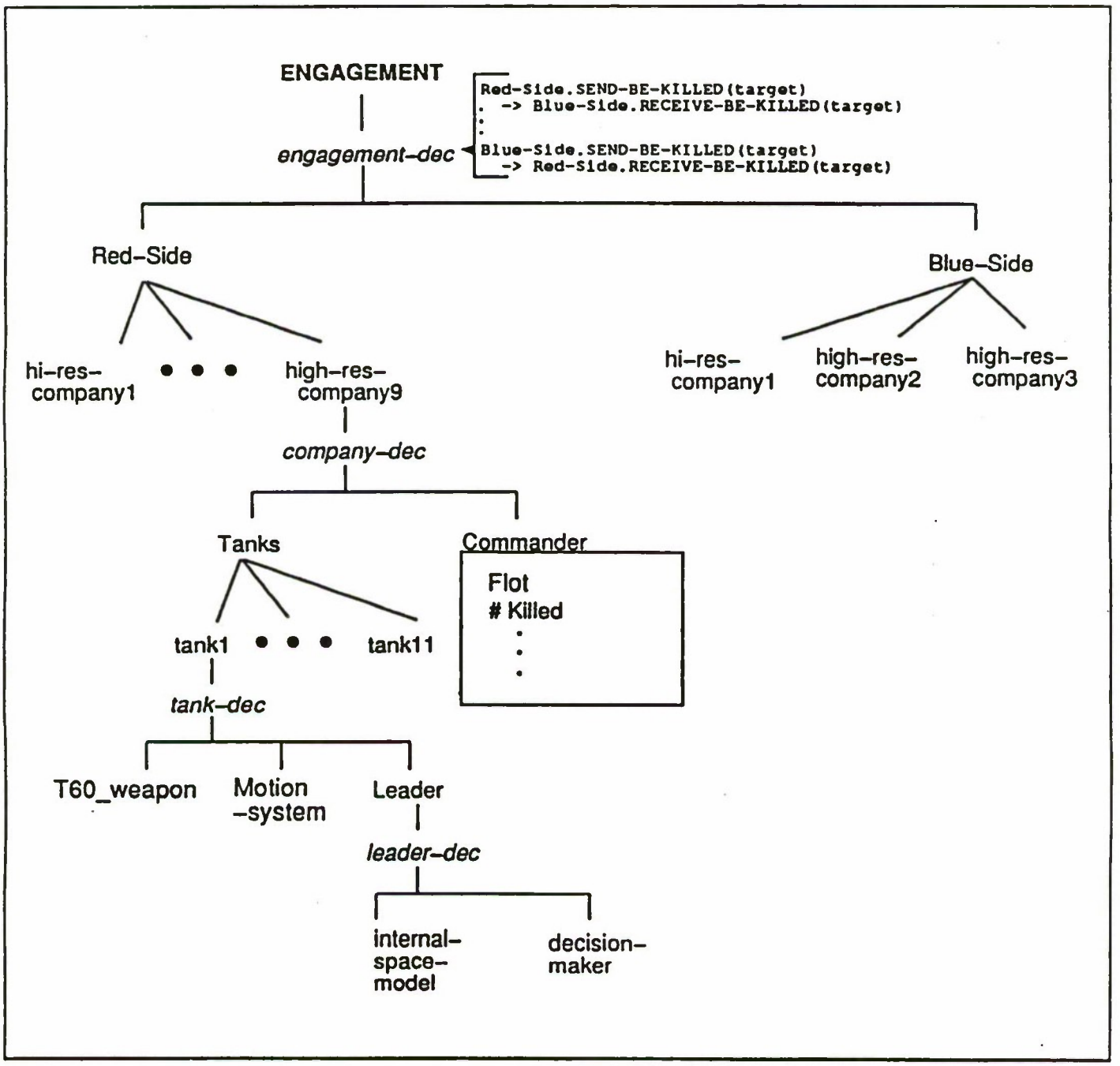


Figure 8: PES ENGAGEMENT for Regiment-on-Battalion Engagement

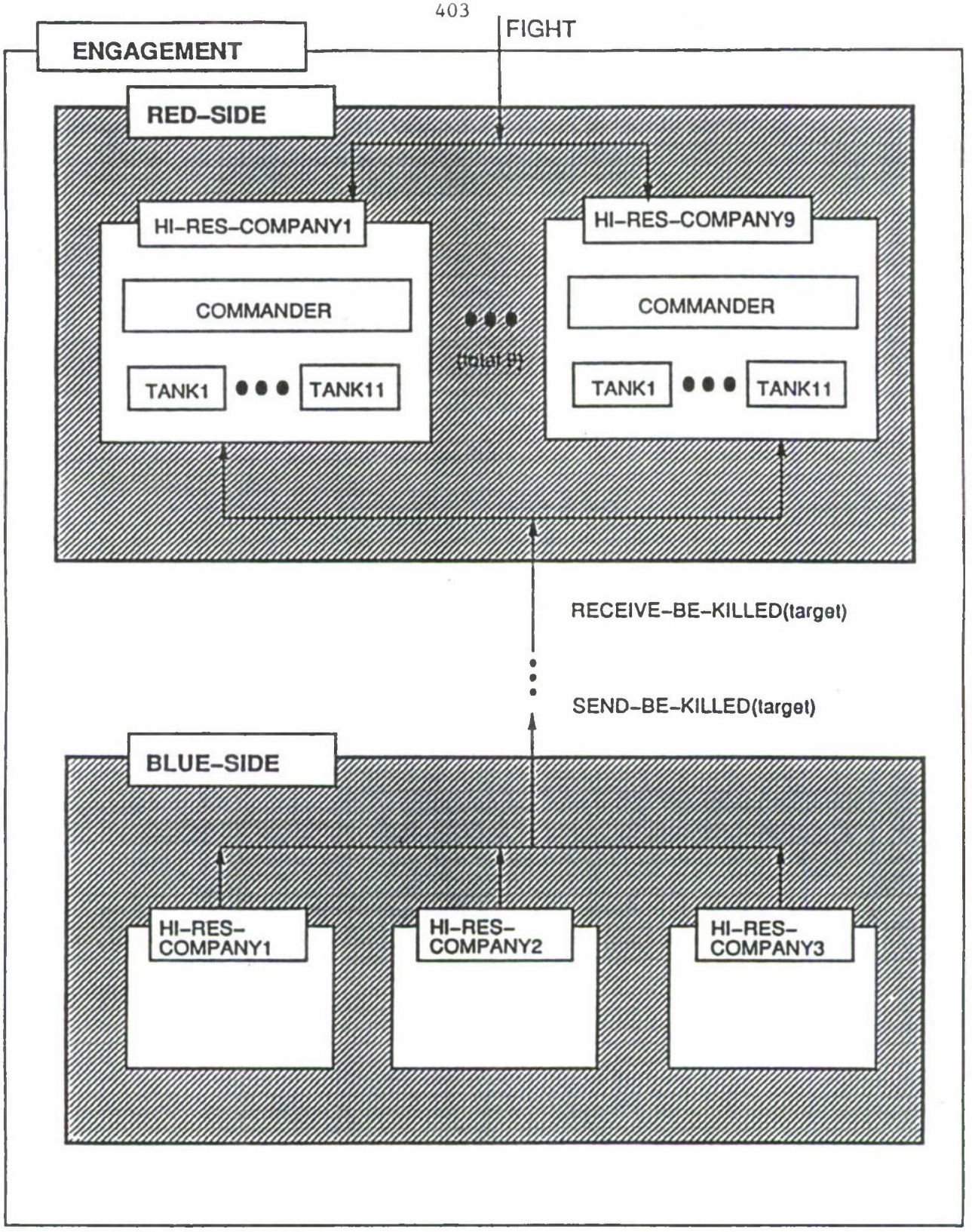


Figure 9: Hierarchical Model of ENGAGEMENT for Regiment-on-Battalion Engagement

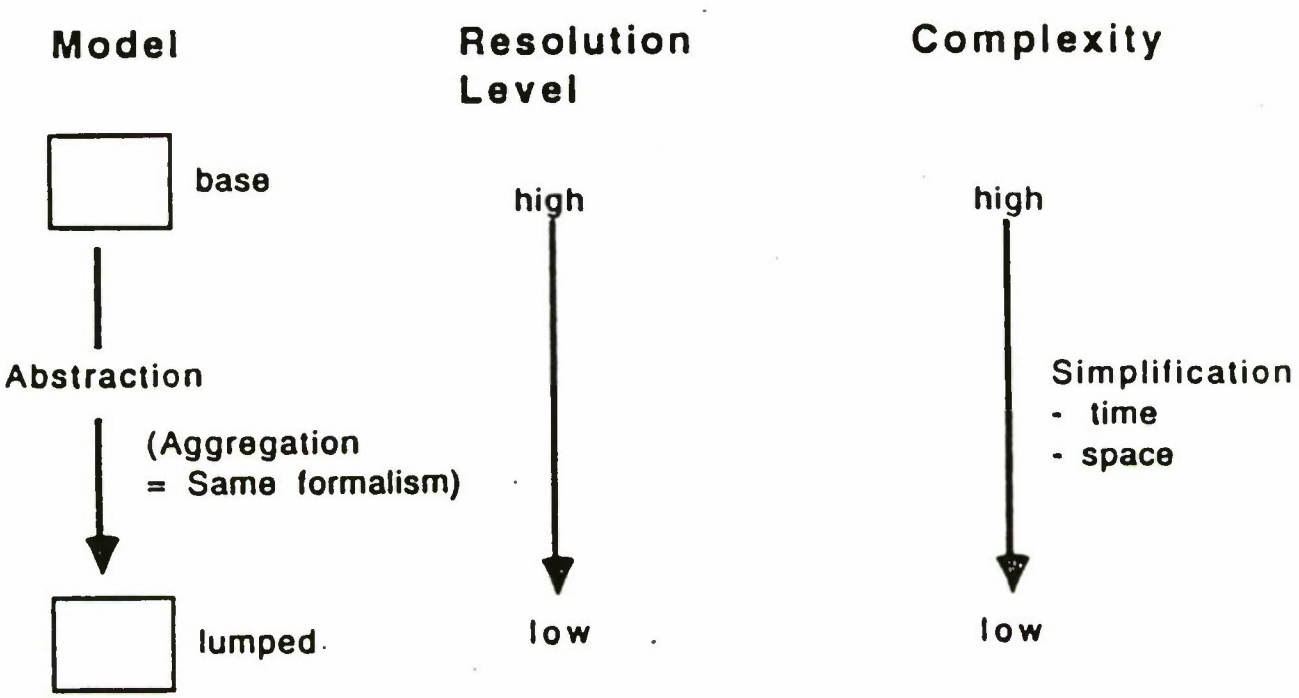


Figure 10: Terminology for Multiresolution Concepts

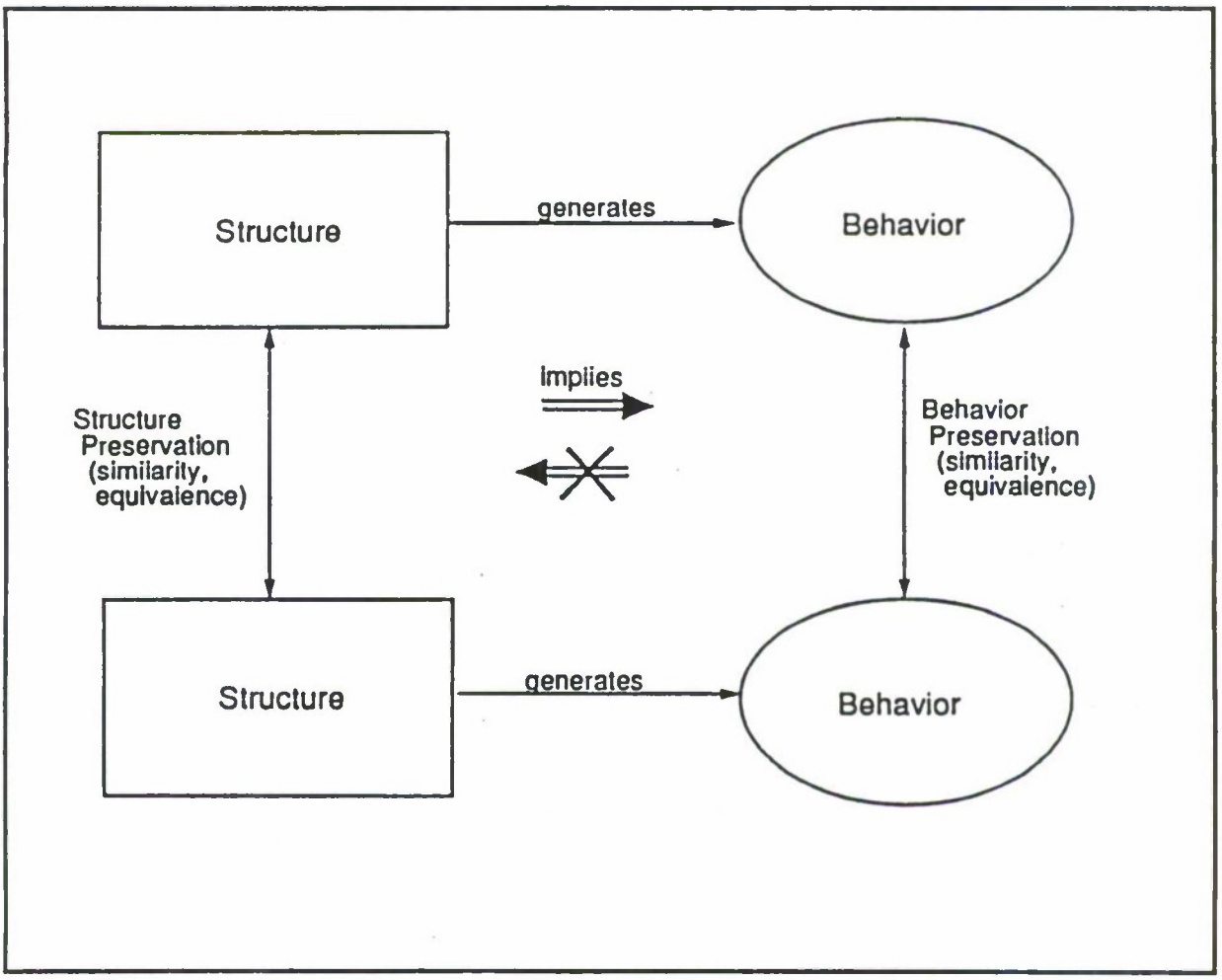


Figure 11: Structure/Behavior Morphisms

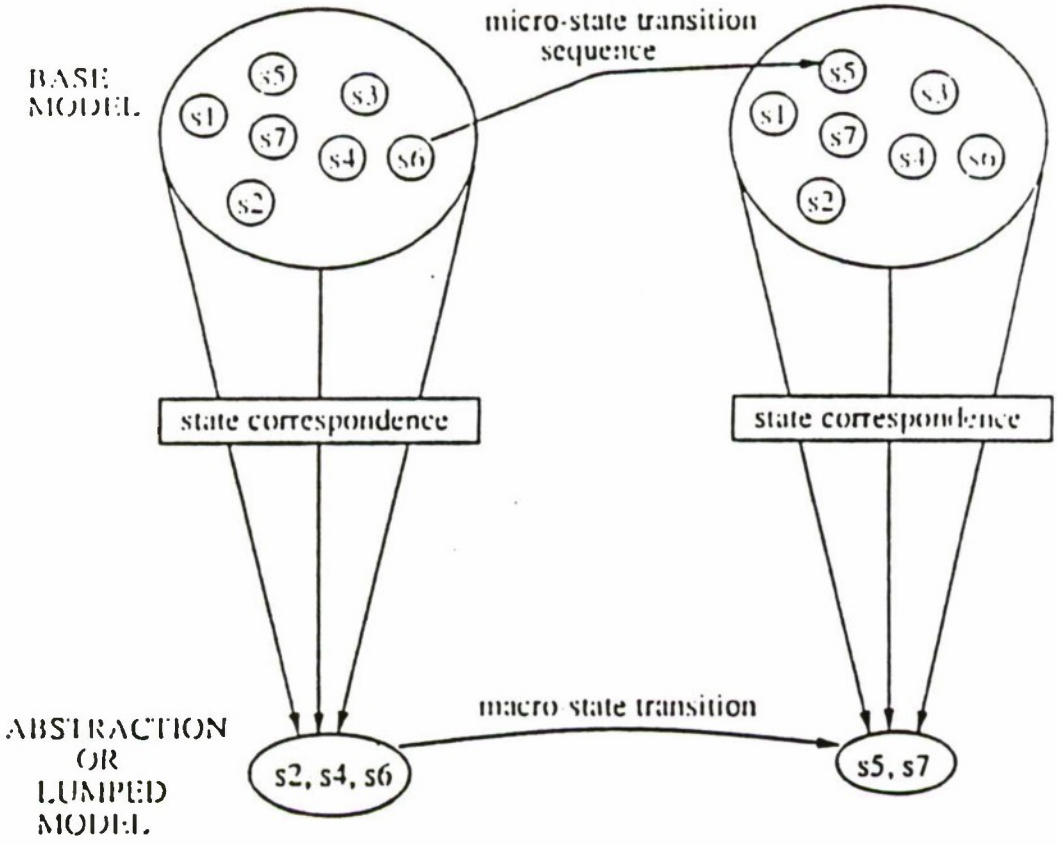
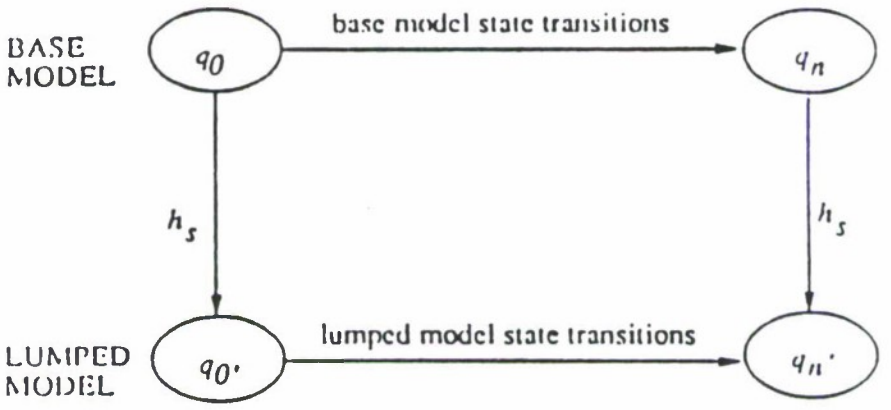
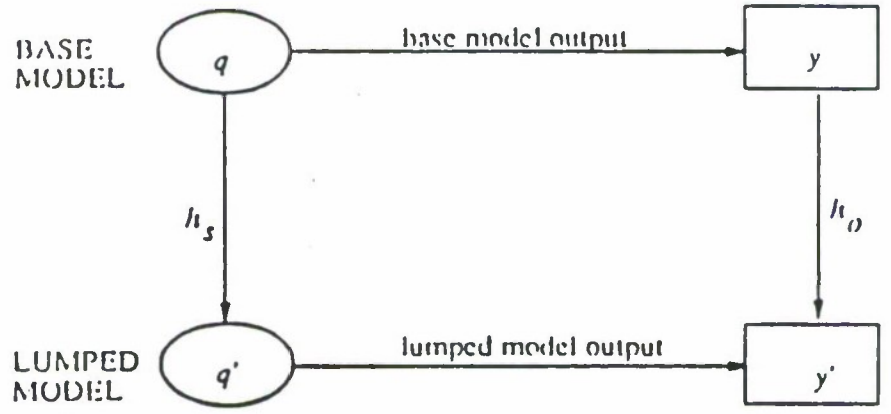


Figure 12: Basic Homomorphism concept.



(a) preservation of transition function



(b) preservation of output function

Figure 13: Commutative Diagrams of Homomorphism.

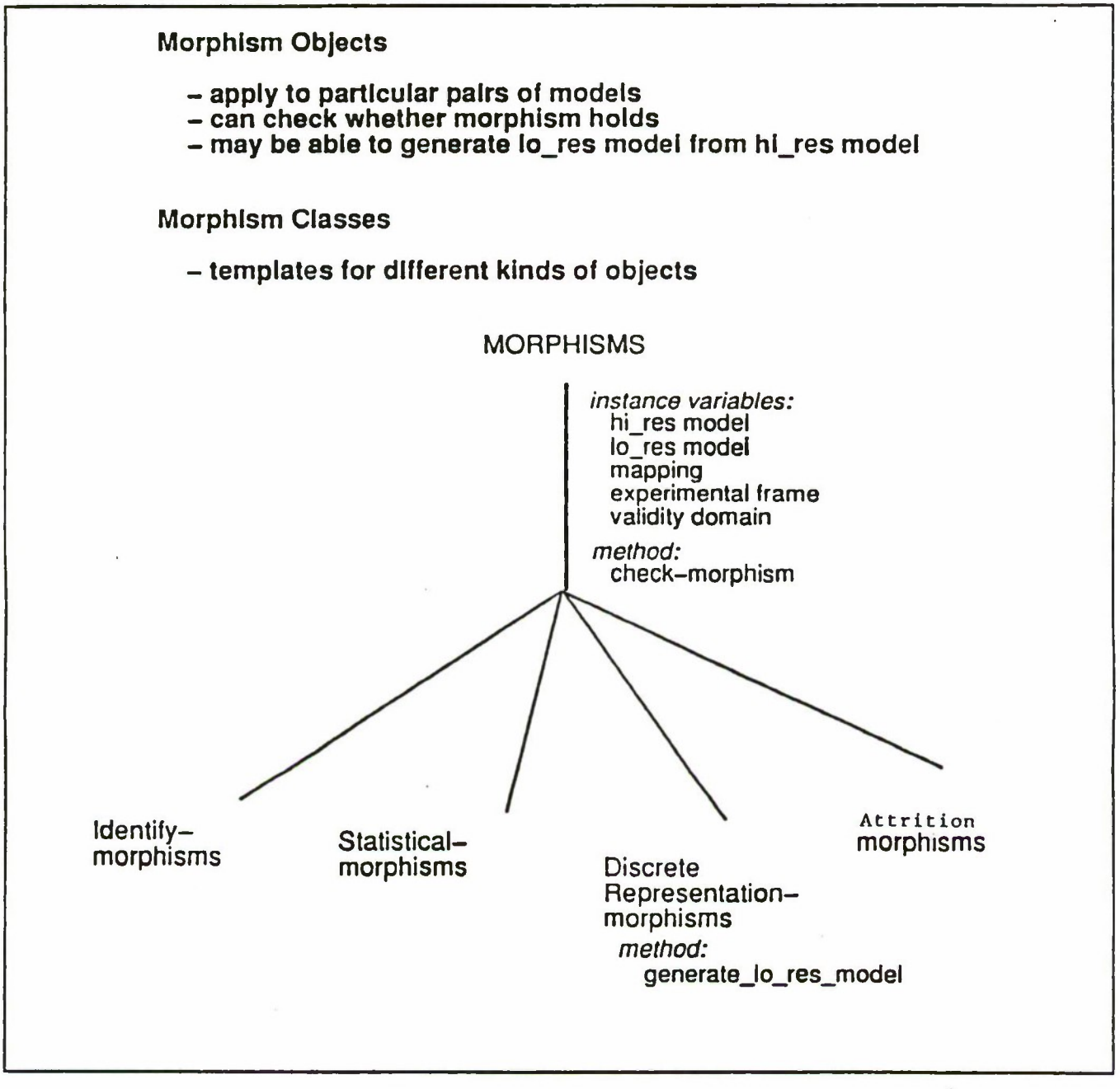


Figure 14: Morphism Classes

Integrated Simulation Environments

Dr. Christopher Landauer, Dr. Kirstie L. Bellman
Computer Systems Research Department
The Aerospace Corporation

Mail Stop M1/102
The Aerospace Corporation
P. O. Box 92957
Los Angeles, California, 90009-2957

Phone: (310) 336-5635 (CL), (310) 336-8604 (KLB)
FAX: (310) 336-6808
Internet: cal@aero.org (CL), bellman@aero.org (KLB)

June 25, 1992

ABSTRACT

This paper describes a modelling environment that can combine mathematical, artificial intelligence, and many other computer-based methods to explore a problem space, helping a user to determine important problems to solve, formulate them in useful ways, and (when possible) solve them. In many problem areas, the individual "solution" is the least important part of a problem. It is much more important to understand the critical features of a problem and characterize the trade-offs among them. All problem analysis in this environment is done within the *problem space*, which is the abstract space in which the problem can be formally stated, and its main goal is to construct a *solution space*, which is a kind of abstract subspace of the problem space that contains all the solutions. Often times, the important aspect of the problem is not "which points satisfy all the constraints", but "what are the important constraints". Having the constraints explicitly described allows them to be studied directly.

Such a modelling environment allows human users to access the powerful computer tools that are available, to use those tools appropriately, and to use those tools together in an integrated fashion. One application focus of this kind of environment is Variable Resolution Modelling, in which simulation program elements exist at different levels of resolution, and must interact or change in a variety of ways.

We use a new approach to constructing this kind of modelling environment. To insulate the users from many of the software integration details, the environment uses explicit knowledge of its own structure to support the user in selecting and adapting the system components. The knowledge is in the form of "wrappings", which are expert interfaces to the programs, tools, and other resources in the environment. This approach is a simple and powerful mechanism for allowing different kinds of resources to work together in an integrated way. We will describe the structure of a program, *vsim*, that implements this approach. The *vsim* program is used to study both the types of wrapping descriptions and the wrapping processes.

The investigations reported here were partially supported by the Aerospace Sponsored Research Program. The authors would like to thank the members of the *Vehicles* Project at The Aerospace Corporation for their continual encouragement and stimulating questions.

Key Phrases: computational reflection, computer support for modelling, integrated software environments, knowledge-based simulation, modelling complex systems, open systems, problem-solving architectures

1 The Vision for Variable Resolution

One application focus of this kind of environment is Variable Resolution Modelling [9], in which simulation program elements exist at different levels of resolution, in time, space, object description, etc.. We use this term inclusively, to apply also to Cross Resolution (in which program elements in the same simulation operate at different levels of resolution), Multiple Resolution (in which program elements may run at different resolutions simultaneously), and the more restricted use of Variable Resolution, in which program elements can change their resolution according to context (this includes reconfiguration at compile time according to problem context).

Our vision for Variable Resolution Modelling support (which to us includes the related issues of Cross Resolution and Multiple Resolution) is a system that contains many coherent models (mini-models) [44], each of which has an explicit description of its limits of applicability. These models may or may not be small, but they will have some kind of consistent or coherent viewpoint, and they will have different kinds of possibly complex interactions among them (e.g., they may be overlapping and partially incompatible).

In a complex system, there is never a single model, tool, or analysis which will adequately represent the system being modelled. Therefore, the system will necessarily be heterogeneous: there will be multiple models, languages, protocols, and even modelling paradigms. Because a diversity of models, at different levels of precision and utilizing diverse sources of information and types of processing, is needed, the system must have both a number of flexibilities (to allow for the use of these diverse resources) and integrative capabilities (to help support the user making use of the different model and software resources in the system). The system needs a flexible response to modelling and analysis problems to construct an integrated application for a problem, to help the user to study the problem on-line (interactively), and to provide support for all stages of model use (model development, analysis, comparison, experiments, and configuration management). Part of the way that flexibility is provided is by having "intelligent user support" functions for the user (whether another machine or a human). That is, services and knowledge that help the user to select and adapt the resources in the system to their problems.

These goals are ambitious. First, we need to consider how to build a large modelling environment. As part of this, we need to understand how to define and exercise models, how to evaluate models, how to compare and contrast models, how to identify parts of models that need improvement, how to retain old models for future reference, and how to use existing models (it is hard to extract them from existing programs). There is no magic and no easy answer. At the moment, our approach involves the processing of explicit knowledge about the models, their limitations, scope, correct context of usage and interactions with other models. Eventually we can conceive of having some processes that will be "smart" enough to discern some characteristics of models without the collection of explicit knowledge (much as theorem provers are slowly gaining the capability to prove the existence of certain important features or relationships in a system). Ideally, one of the results of this research will be a set of guidelines for developing new models, so that they can be integrated usefully into such a modelling environment.

When modelling a complex system, we have found that modelling some aspects of the context of a problem is as important as modelling the original problem. This multiple-layer approach of having some models act as context for others provides information to the processes that select and adapt models to a given problem. These models of the context also serve as a way of organizing some of the qualitative information that is used to evaluate the system or the system's behavior as a whole. For example, the explicit scope of the system, the characterization of the problems handled by the

system, the criteria for attaching priorities to the global characteristics of the system for evaluation, and some of the relationships among different sub-aspects of the system, and across levels of the system under consideration, are all important qualitative aspects of a complex system that we make explicit in higher-level conceptual models.

The basic VR problems in this context are interacting with models at different resolutions, dynamic adjustment of resolutions, and coordination with external sources of resolution (like real-time clocks for human-computer wargames). Each of these problems is a hard technical challenge, which is only partially solved in special cases, but there is the more general challenge of being ready to incorporate whatever solutions are devised by having the software infrastructure to integrate them.

2 Software Integration

Software is expensive; new programs for complex tasks are too late and too expensive, so we want to use old programs. One main difficulty with using old programs is the unspecified domain of applicability of most programs. They work quite well in certain contexts, but there is no adequate record of which contexts are and are not appropriate.

Another difficulty with new software for complex environments is the lack of adequate requirements. To build better software, the requirements must be known before the software is designed. However, we cannot always specify the system requirements before the system (or at least a prototype) is built and fielded. We need instead to design systems in ways that retain flexibility, so that as requirements change, the subsequent implementation changes are not so drastic or difficult. Part of the reason it is so hard to define the requirements for complex systems is that no one model is adequate for a complex system [44], [2], [6]. In particular, there will be multiple models for the same aspect of a system (for comparisons, trade-off studies), different levels of detail of models (for studying different questions), and even different levels of detail in different parts of the same model (for studying some detailed foreground effect against a less detailed background behavior). In addition, if models are serving their purpose and increasing the user's understanding, there will always be the desire to incorporate new understandings and results, leading to a continuing natural pressure for the models to evolve.

Many old programs are "useful but not usable". Collections of these old programs exist, performing useful functions (e.g., numerical methods, astrodynamics, etc.). Part of the problem is that algorithms are not programs. Most software has a small number of basic functions. In order to turn those functions into a program, a large amount of extra code is written: to collect inputs, request selections or other inputs from the user, interact with external data sources, provide displays and other outputs, and move data around from one place to another. None of this code is essential to the algorithm; all of it is essential to the program. Modularity is a necessary step in making such a program comprehensible. However, the component interactions are usually function calls (when they are not references to global data), which are quite rigid in data formats and interaction style. The programs themselves are usually both rigid (there is one way to interact with each program) and brittle (invocations that do not follow the often unstated assumptions cause trouble). They are also less apparent (it is hard to determine what they do) and justifiable (it is hard to evaluate their behavior).

2.1 The *Vehicles* System

Our new wrapping approach originally arose as part of the *Vehicles* System [5], [6], [12], a large software environment supporting the conceptual design of spacecraft, but the approach applies more generally to any software environment containing a heterogeneous collection of software resources (programs, databases, computational tools, interfaces, simulation models).

The Aerospace Corporation's role in defining new space systems and analyzing new space missions emphasizes the need to perform concept exploration studies that use diverse models, including simulations, analytic equations, and other software programs. The key to effective modelling in this context is a software environment that supports flexible use of a wide variety of models to help pose and answer (or at least study) questions. This breadth requirement leads to the incorporation of many heterogeneous methods and software resources, including analysis tools, external programs, datasets, and models. This flexibility requirement leads to the need for rapid integration of models and other resources.

The *Vehicles* System supports the design and analysis of complex systems by providing many tools to analyze system models, including those for comparison of different models, trade-offs between performance variables, parametric study of relationships among variables, and sensitivity analysis of equational models. Because of the large number of analysis tools, and the flexibility and integration support, the *Vehicles* System has recently been used for rapidly prototyping other modelling and analysis environments, beyond conceptual design of spacecraft (its original application), such as environmental tests of satellites (shake, bake, rattle, and roll tests before launch), space debris hazard analysis, and an architectural concept evaluator for the Integrated Satellite Control System.

The capabilities and types of information required in a system that supports conceptual design and planning are numerous, varied, and complex. Designers and planners of space systems take many different approaches and may work at different levels. Mission requirements, technological advances, and constraints on cost or other resources all influence design decisions. Consequently, design tools must be flexible enough to enable one to work at any level, without the need to be immediately aware of all the factors affected by each design decision. By operating in this way, the *Vehicles* system will allow designers to address many different design problems, from the definition of mission requirements to scheduling issues related to the effectiveness of environmental testing. Within the *Vehicles* system are general analysis tools that perform trade studies; those tools may be used at any stage of the design and in any context. When and how a tool is used may vary; however, the basic flexibility of the tools and the environment in which they are used makes it feasible to use them creatively in a variety of situations.

With the *Vehicles* system, one can start at any level. One can start a design with broad mission goals, then work down by interpreting the implications of the mission requirements on subsystem requirements. One could also start with a given subsystem, then work only on sizing that subsystem, analyzing the impact on other subsystems. Furthermore, one could start with a given component, then study the effect of its design on higher-level requirements and performance. As an example of the *Vehicles* system's ability to support design diversity, some projects may begin with a new technology, with the goal of determining what new capabilities are possible. Such projects can start with a known space system and analyze how a new technology would affect not only the performance of the entire spacecraft, but also the requirements on other subsystems should the new technology be adopted. Other design projects may begin with predefined payload and mission requirements; the integration of multiple subsystems or the comparison of several design concepts may be their only goal.

As sets of possible design solutions are generated, the user can compare designs and evaluate their relative merits. The *Vehicles* system has taken a significant step toward speeding up the creation of *families* of viable designs, rather than merely creating single-point solutions. In addition, because designers frequently want to add their own analytic capabilities to the design environment and to access their own databases, we have created links to external programs.

2.2 Software Requirements for Support of Complex System Modelling

Based on our experience with the *Vehicles* system, we briefly list some of the software flexibilities we have found necessary to a modelling environment. In the *Vehicles* environment, and in our other research environment prototypes, we are gradually finding better and better implementations for

the flexibility and integration mechanisms we discuss below.

Each flexibility in the software system is a way of using diverse sources or types of processing in the system. An important principle that has emerged from the work on the *Vehicles* System is that for each type of flexibility, one must provide a corresponding *integrative mechanism* to help the user manage the diversity in the system. For example, multiple models are managed by processing explicit selection criteria, and problem-based configuration is managed by interpreting explicit problem descriptions.

Flexibilities are required at many different levels in an environment such as this. Component flexibility requires different levels of model detail for different studies, so that, for example, less precise (time-consuming etc.) components can be used where possible. Then additional flexibility is required so that simulator configurations can be constructed automatically (or with as much support as possible) according to the problem under consideration.

Part of the ability to reconfigure components rapidly depends on not having rigidly-defined component interfaces. Interface flexibility requires methods of managing component interactions and for example, recording appropriate summary data, so that the monitoring functions can determine what the system was doing. Integration flexibility includes the ability to handle very diverse types of information sources and processing capabilities, and to extend to new sources and processing types. Model development flexibility requires the ability to formulate and compare alternative models (which includes the terms used in the model, the constraints, the context, the numerical or simulation methods used, and so forth), and to study architectural choices explicitly.

We want the system to provide many different kinds of flexibilities at these different levels. There will be different languages available for descriptions and definitions (for resources, knowledge representations, data transfer formats, etc.). There will be different kinds of resources for the same application (multiple tools, with selection criteria, and user screens, including menus and other user interface languages), and there will be different choices for basic processing functions. In the next section, we discuss how even the functions that process and organize the system resources are selected and adapted (e.g. planning and problem solving, knowledge base access and interpretation, and message distribution).

Another important principle is that the appropriate flexibility requires all of the models to be made explicit, and that they be explicitly described in ways that can be processed by the environment. These descriptions are called "wrappings" [4], [25]; they are fundamental to our methods of allowing the environment to assist the user in selecting appropriate tools, models, or other resources, adapting them to the study context, and interpreting their results.

In our view, all components of a system are resources: external programs and data files, user interfaces, computational tools, databases and database systems, simulation support function libraries and complete simulation programs, rule bases and inference engines, symbolic formula manipulation systems, scripts that refer to other resources (e.g., plans), and analysis tools that refer to other resources (e.g., parametric study).

3 The Wrapping Approach

It has become clear that our architecture studies are much more general than the *Vehicles* System for conceptual design. They apply in any software system that has requirements for heterogeneity, whether it is for different languages or processing paradigms, different existing computational programs, or combinations of other software resources to support analysis and modelling of complex systems.

The power of any complex software environment lies in its collection of models and modelling

paradigms (in this context, the programs and databases are also models), and it is clear that harnessing that power effectively is a difficult problem. As noted above, supporting the design and analysis of complex systems requires a diversity of models and tools; the result is often a software environment that becomes itself a complex system. Hence, we feel it is important to provide *Intelligent User Support* functions [3]; that is, some means of supporting the user (be it human or another computer program) in the selection, assembly, integration, adaptation, and explanation of the software resources.

The "wrapping" methodology builds flexible environments by encapsulating both programs and data with the explicit knowledge they embody, and with knowledge of their various styles of use. These wrappings provide standard interfaces to software resources, and provide knowledge about the resource, so that other tools in the environment can interact appropriately with the wrapped resource, either to provide it with information or to use its information effectively. Several overviews or applications of the wrapping methodology have been presented elsewhere recently [3], [4], [24], [25]. The software architecture we have developed has two main components: the wrappings, which are knowledge-based interfaces to the resources, and the Study Manager, which is a program that processes the wrappings.

3.1 Wrappings

A wrapping is an expert interface that describes a resource in a complex software system [6], [21], [22]. It contains an explicit, machine-processable description of the resource, used for management of the system architecture, both short-term (run-time) and long-term (configuration). A wrapping describes a resource, but does not define it (i.e., it describes some characteristics of a resource, but not enough to define the resource behavior completely, so, for example, formal specifications of the behavior are not required).

Wrappings are processed by various programs that are coordinated by the Study Manager. All resource interaction is intercepted, so that resources make service requests instead of function calls. That way the requests for service (e.g., information requests) need not know the information source. The wrappings help make the connections between the request and response. Also, all activity is recorded for monitoring and analysis.

A system that uses wrapping also has an explicit machine-processable model of itself and its processing, a notion called "Computational Reflection" [29]. The self-model supports explicit configuration control from within the same system, and provides a basis for self-monitoring, explanation and failure diagnosis. For example, the system can be used to support an active sort of configuration control. Instead of just keeping track of versions of resources, the system can simulate the effects of changing certain resources or adding others.

The next major notion is to wrap everything (Everything!): All tools, data and other software resources are wrapped (external programs, data files, user screens, plans and scripts, and other resources listed above). A resource need not be an entire program. Certain uses of a complicated program may be considered separately. Different styles of use of a complex resource can be wrapped separately (many wrappings for one resource). Combinations of resources that commonly apply together can be wrapped together (many resources for one wrapping). Even (especially!) the programs that process wrappings are wrapped, so we can study the wrapping processors with the same system. In addition, we view the user as a resource also. The user screens define what the system expects to show and tell the user, and what the system expects to get from the user (in the way of selections and input data). These resources are a way of describing the expected interactions.

Finally, there are several associated knowledge bases: the Planner KB (PKB) associates problems, resources, and available information, in the form of simple triples for now, and the Wrapping KB (WKB) contains all the wrappings. The wrappings contain a large amount of knowledge of what the resources do and how they should be used. They describe properties of resources from many

points of view. Wrappings greatly simplify the problem of integrating software, by localizing the information required about a program to a wrapping, and by explicitly describing many kinds of use of the program, so that the environment can both use it appropriately and reason about its use.

The wrapping should also respond usefully to certain other questions about its computational process and status, in order to produce a completely separable tool that can be used by other programs or people. Such questions include ones about the purpose of the program, its applicability (e.g., When is this program appropriate?), its limitations and assumptions, and the type and format of both the data it requires and produces. When applicable, it will contain the information for deciding whether the program has enough data now, or if not, what actions may be taken to produce the data needed. It describes how to invoke this resource, its different interfaces and any default values. It includes the domain of inputs (any constraints on the inputs) and how, if this input data does not fit the program assumptions, then how it can be altered to fit those assumptions. It also includes such overall characteristics as how long the resource takes and how much it costs usually to use it. It may include any error handling mechanisms or explanations of errors. Lastly, it includes information on the type of algorithms, and information internal to the program that may be needed for deciding whether to use the program or for explaining its use.

The language in which this information is to be represented is currently under development. It presently consists of keywords for the type of query, and one or more qualifiers to separate queries of different types. Much more semantic information will eventually be needed, in order to replace this rather *ad hoc* mechanism with a more formal one.

The wrapping methodology applies to all software resources, including databases, user interface descriptions, computational analysis tools, simulation programs, models and other external programs and data, including the programs that interpret the wrappings of other software. Any part of a complex software environment is considered to be a software resource, and everything gets explicit descriptions.

The descriptions are tailored to expected uses, so that the same software may have many different descriptions. A program may have many partial descriptions; complete descriptions are not needed (they are often much too complicated anyway). Descriptions for human users of the software are different from descriptions for programs that use the software.

A wrapping is more than a nice interface. In any environment, even if it is clear exactly which program to run, help may be needed to run it (getting the appropriate parameter values). Wrappings help find appropriate resources to apply to a problem, given the problem statement. This property is more important in the larger environments, and in the more creative modelling and design problems. Also, a program can ask a wrapping what the resource does, what interface and protocol it expects, and what information it requires and produces.

The problem of large programming environments has become important, since we have found that our large computer system methods fail repeatedly to provide the flexibility and ease of use that are needed. Since the environments have been getting large enough to be difficult to use, there is much more research activity now. A few other papers seem to be heading usefully in this direction, including work on engineering model databases [36], [37], work on "weaving" together program fragments [13], and on using partial views to coordinate among users with different type hierarchies [26]. There is also some work on environments for programming [42], example implementations of environments [39] or of software mechanisms that might be useful for the KBs we want [27]. and large-scale preliminary experiments on interconnection [11]. There has also been some relevant work on models of environments for software development [33], [32], [41], though the models are not typically placed within the system being developed.

3.1.1 Intelligent User Support Functions

One of the first things we noticed when we tried to write descriptions of some resources [3], [4], is that wrappings may be different for different uses of a resource; there is no single description of a resource that is universally sufficient. We have identified a few uses to which the wrappings will be put. They primarily differ according to what will be done with the resource: resources need to be distinguished from others, used with others in various ways, adjusted to fit into a certain context, and justified. The Intelligent User Support functions are:

- Select: Which resource is appropriate for this problem?
- Assemble (syntactic integration): Can these resources interchange data?
- Integrate (semantic integration): Is it appropriate to use these resources together?
- Adapt: How can this resource apply to this problem?
- Explain: Why was this resource used?

Wrappings support access to a large collection of resources. Wrappings help the Study Manager *select* an appropriate resource for a problem, given the context information at hand. This function is the most apparently useful for a large toolchest, and several implementations of this function are being constructed for mathematical function toolchests [18], [7], [28].

Wrappings help *assemble* resources together, by performing data format conversions (using XDR and other common formats). Many people use the term "integrate" for this low-level facility, but we regard this process as "syntactic integration" (i.e., answering the question "can these two programs talk to each other?") concerned merely of data syntax, not nearly strong enough for the full range of integration mechanisms.

Wrappings help *integrate* resources; instead of using explicit function calls from one resource to another, which is overly rigid and brittle in general, the program uses requests for information, which are fielded by the Study Manager and distributed appropriately according to the wrappings. This integration is more "semantic integration" (i.e., about a different question: "should these two programs talk to each other?"), representing a compatibility of information semantics, at each of several different levels of abstraction.

Wrappings help *adapt* resources; when a resource is appropriate to a problem, it still needs certain information about the problem, including its context and often more. The input files defining the specific problem must be built; this process often requires new information requests. In addition, some resources are configurable at run-time to use different data access mechanisms or strategies.

Finally, wrappings help *explain* the use of the resource: under what conditions it should be used, what kinds of information it requires and provides, and how to interpret the results of using the resource. Explanation is the most difficult of these uses, since it must facilitate an overview of the way the resource fits into a context of use.

In addition to the Study Manager, different kinds of planners can use wrappings to organize groups of resources to solve problems, or provide the user with insight into the problem.

3.1.2 Wrapping Methodology

This section describes our approach to organizing and using the wrappings. It includes a description of our wrapping semantics. There are basically two kinds of entries in the wrapping of a resource.

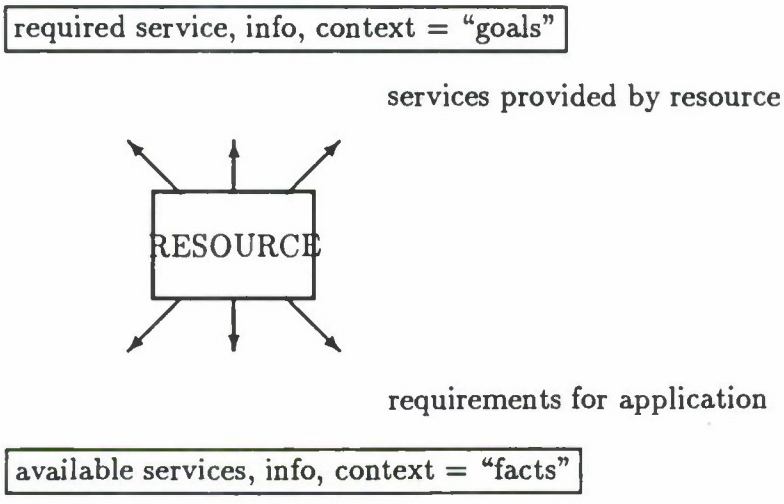


Figure 1: Resource View for Planning

The Application criteria describe how to use the resource (e.g., using XDR for the data description, CSP for the protocol description), and the Selection criteria describe whether and when to use the resource.

Context is used to help make selections, using a parser-based planner described below. The basic idea is to treat each resource as being appropriate only in certain problem contexts, providing certain services, and having certain service requirements, as shown in the picture in Figure 3.1.2. Then these service units are combined into a configuration that will provide the service required by the problem.

The wrapping semantics that support this combination include both local interpretation of problem language and restricted scope. Our approach to wrapping semantics assumes that resources will be examined in a context of goals, information, and services.

The resource wrappings are divided into several parts.

- Who am I? (Identification of resource)
- Problem name and (zero or more) qualifiers (Simplistic problem description keywords for now)
- What does this problem mean to me? (Problem interpretation maps external language to internal language)
- What context allows me to be considered? (Requirements this resource must conform to)
- What requirements do I need from the context in order to be applied? (Requirements this resource adds to the context)
- What new information do I produce when applied? (Or what service do I provide?)

First, there is an Identification Section that gives the resource a reference name, for use by the system. Then for each problem that can be interpreted by this resource, there is a *Problem Application Description*, which describes whether and how the resource applies to a particular problem.

The Problem Application Description has a simplistic problem name and (zero or more) qualifier keywords (so that the problems may be grouped into major classes, and distinguished within the classes by qualifiers). We expect that eventually problem statement notations will be devised that are interpreted to understand what problem is being addressed.

The problem description divides the rest of its information into four parts: Interpretation, Context, Requirements, and Products. The Interpretation Part maps the external problem language, which is suited to the problem, into an internal language, which is more suited to the resource. It tells what the problem means to the resource. This separation means that new applications can use their own problem language, and the wrappings arranged to interpret that language in their own terms. The Context Part defines what context allows the resource to be considered, including requirements the resource must conform to, available information, problem context, services available, and goals. The Requirements Part explains what requirements the resource needs from the context in order to be applied. It also defines requirements that the resource adds to the context for input data, supporting processes and services. The Products Part defines what new information is produced when the resource is applied, or what service is provided. This notion of having the resource itself say what it can do is important for reasons of locality and scaling, locality because the information is local to that resource wrapping, and scaling because there is no need to have the wrapping contain all possible uses of a resource.

We will use a parser for matching and resolving these wrappings. Below, we will explain the matching process for building scripts to invoke groups of resources together. We assume that we have both goals and information available. Goals are problems to be solved: information to be found or services to be constructed. Information is context, available data, or available services. The wrapping may mention these either as requirements or as products.

We use a variant of the Cocke-Kasami-Younger parsing algorithm [1], as modified by V. Pratt [34] and C. Landauer [19] to use Early algorithm type pre-filtering of rules (see [14] for another description and a complexity analysis). This parser matches both goals and information simultaneously (both bottom-up and top-down), which is both forward and backward reasoning in the planning context, and uses a pre-filter of applicable rules, which in this application uses PKB information.

We use it because it produces all partial parses, which in this application means that it computes all sets of resource applications that satisfy the original goal(s), and also produces all partial sets for diagnosing planning failures, both what information was missing and what resources could not be applied. Then the rest of the selection process can choose one plan to implement.

3.2 Processing the Wrappings

With the above collection of resources, we can now discuss how wrappings are used. We start with the notion of a *study*, which is our term for any organized analysis of a problem, or a system, or a model (e.g., a study need not be intending to solve a particular problem; it could be investigating the behavior of a certain model). The standard problem solver we have selected is called the *Study Manager*. It is the program that reads and interprets the wrappings, and organizes the studies.

3.2.1 Study Manager

The Study Manager is the heart of our problem solving strategy. It organizes problem solving into a sequence of standard steps:

- Pose problem
- Interpret problem (convert problem into resource)

  - Match (identify candidate resources using PKB)
  - Resolve (negotiate with wrappings in WKB)
  - Select (final resource selection)
  - Advise (notify poser for final selection)
  - Adapt (set up resource for problem, given current information)
- Apply (use resource on current information)
- Assess (evaluate results; maybe pose new problems)

We will describe the sequence in its simplest form, and then discuss its use on more complicated problems. The first step is "pose", which is to decide what the problem actually is. The problem poser can be a human at a terminal, another program using this system environment, or this system itself.

The next step is to "interpret" the problem. We have chosen a simple strategy here, based on the wrappings. The first part is to "match" the problem statement and the available information (which is whatever is in the current context) to the PKB entries, to determine a list of candidate resources, any or all of which may be useful. The second part is to "negotiate" between the Study Manager's knowledge of the problem and available information and the candidate resource's requirements and assumptions. The result is also a list of candidate resources, presumably smaller. The third part is to "resolve" the candidate resource list into a single resource selected to be applied. This choice process may involve automatic selection in some cases, or it may involve a request to the user to make the choice. The last part of problem interpretation is to "advise" the problem poser of the selection made.

The next step is to "adapt" the resource to the problem and the information at hand. This step may include more negotiation with the wrapping. The last two steps are to "apply" the resource to the problem, and to "assess" the results.

This problem solving sequence is completely recursive. The step of "posing" a problem is also a problem, and uses the same sequence to try to solve it. The step of "matching" a problem to a resource is another problem, and uses the same sequence also. The bottom of the recursion is some very simple processes (poser, matcher, negotiator, selector, advisor, adaptor, applier, assessor) that just read the different knowledge bases with syntactic searches. The poser reads the PKB to determine what information might be needed to fully specify the problem, and may pose a further problem to "specify this problem". The matcher reads the PKB to make its candidate list, or it poses further problems to determine where to find the needed information. The negotiator reads the WKB to check on other requirements for each candidate resource. The selector asks the user to choose unless it has selection information in the WKB. The advisor tells the problem poser something. The executor expects to use a simple function call, local or remote.

Because each of these processors is also wrapped, and because there will eventually be several matchers, several negotiators, etc., the problem of identifying relevant resources or negotiating information will be solved in much more clever ways as we learn more about these problems. Our study of these methods will use exactly the same wrapping mechanism and Study Manager as is used for applications now. This choice allows study of alternative problem-solving mechanisms and planners. In this context, a planner is a resource that transforms a problem into an organized set of smaller problems.

3.3 Experiment: Program *vs*

In order to demonstrate and study the wrapping approach, we implemented a study program called *vs*, for *Vehicles Simulator*. It was an attempt to build an entire system around wrappings, and to

study how they ought to be used. Its important features are the Study Manager, the knowledge bases, and certain other information sources and software components.

The Study Manager included recording of all activity, the recursive processing described above, with a simple matcher, resolver, selector, and assessor. It was implemented as a simple finite state machine. The first two versions of the knowledge bases were implemented, a preliminary WKB and a preliminary PKB, implemented as triples (problem, information list, resource) that are used to find potentially applicable resources.

Other information components included a Scenario Interpreter (that read a scenario data file, so we would need no user interface for the experiment), a database of models and user screens, a Workspace to hold the current state and any auxiliary information, and a Study Log in which the Study Manager recorded its activity.

Other software processes included knowledge base input via flex and bison (tools available from GNU), and dynamic loading of tools via *dld* (a program from UCDavis [15], [16]).

This program ran a few simple problem scenarios. It demonstrated the recursive nature of the Study Manager, the interaction between the different knowledge bases, and the inadequacy of its own architecture for larger-scale studies. The rest of this paper will describe the revised implementation that alleviates some of the scaling problems with *vs*.

4 Implementation: Program *vsim*

The wrapping approach is being implemented in a program *vsim* (for "*Vehicles* Simulation"), which is both an experimental platform for trying out new integration ideas and a prototype implementation of the wrapping approach to software. This section describes the implementation, and shows how the wrappings are constructed and processed. We call it a simulation because it is not a single implementation; wherever there are design questions, several possibilities have been implemented, with the choice made at run time. Some of this flexibility will remain in the system, but most of it is at too low a level for applications.

The most important architectural feature is its division into layers. The program is structured into four layers, each of which has a different view of the system. Parts of the overall architecture have been implemented in various experimental programs.

The lowest layer is the *vsim* Message Layer, which is simulated by a program *vk*. It is structured as a message-passing kernel or software bus, in the style of MACH [38], with very few primitive functions and very little knowledge of the semantics of the messages. All resources pass messages to each other using this layer; all parts of the higher layers are resources. We expect to use standard UNIX network services (the "Portmapper", XDR, etc., [43] to implement the message-passing kernel eventually, once we decide what is needed.

The next layer is the *vsim* Core Layer. It contains resources that process the wrappings, including the Study Manager, the knowledge bases that support wrapping of these resources, the programs that read those knowledge bases, and the various planners. It does not depend on which message-passing kernel is used, only on the message-passing interface provided. In particular, when the *vk* program is replaced with a different message-passing kernel, the Core Layer will not change. Also, if a different Core Layer is written to use the same interface for message passing, then it can be used with no change to the Message Layer. These two layers are the core of *vsim*.

The two higher layers are even more flexible. They interact with the core of *vsim* only through the wrappings, and the supporting knowledge bases. The third layer is the *vsim* Resource Layer. It contains a large number of software resources, both programs and data, user screens to interact

with the resources, and other general utilities. These are the domain-independent resources. Also included are tools for mathematical functions, experimental design, statistical analysis, graphics, etc.. It also contains the knowledge bases that support wrapping of these resources. The fourth layer is the *vsim* Application Layer. It contains the models and other resources for the particular application, and the knowledge bases that support wrapping of these resources. Three of the layers contain knowledge bases that support wrapping of resources. These knowledge bases have completely independent syntax and semantics. They could also be treated as single knowledge bases having up to three parts, one in each layer, and we will do that wherever it is convenient.

This implementation will contain some of the results from previous experimental implementations: the Study Manager and dynamic loader from *vs*, the syntax-directed KB readers, the layered architecture and message-passing kernel from the *vk* experiment, and the planner KB processing and network code from one of our applications.

There are also some new features, mostly in the Core Layer: we now have some defined semantics for KBs, and there will be KB readers and interpreters to process those semantics. There will be a state history tree that keeps track of all user and system activity, which will allow an extensive undo facility (actually, more like a "go back to that point and try something different" facility), and also a way of changing tack entirely, moving to a completely different branch of the tree. We also will have the planner that was described earlier to help satisfy goals that are presented, and a user interface language for defining new interaction resources.

Finally, we expect to bootstrap the development process. The wrappings will be used to help track software changes and compare alternative software architectures as the system is developed.

4.1 Message Layer

The message layer treats all resources as message senders and receivers. There are several features of the message-passing kernel that interact to provide the message service. The program uses registries (a kind of local database) to contain the message switching information, message queues to organize the acceptance and delivery of messages, a very simple message switcher function, and servers for various important functions. We make use of UNIX via a dynamic linker that allows function calls by name at run time.

Messages are distributed according to an association of message types to interested resources. A resource can register its interest in a set of message types. The message-passing kernel will send along every message of that type. There is no conflict between resources; multiple resources interested in the same message type will each get a copy. They must coordinate among themselves to resolve any resulting problems.

Registries are maps, from a resource name to the location of the object code, and to the function name for message delivery. All resource access is through function calls; a separately run program will have a small function that invokes it, implemented as a function call. These two maps are fairly static, primarily changed at build time. There are also more dynamic maps, changed at run time, from message type names to interested resources, and from resource names to interesting message types. This "interest" relationship is symmetric, and it is necessary to access it in both directions.

The desired function call code is not all linked into the *vk* program at link time. We use *dld*, the dynamic linker from UC Davis [15], [16]. It performs at run time (in the data space of a program) the same operations as the UNIX loader program *ld* performs at link time (in the text space of a program). It is therefore possible to call the resource functions by name, and rely on the dynamic linker to resolve the external references. One of the registries is used to find the object library or file that contains the object code of the resource function.

The message queues are all strict FIFO queues. There is one incoming message queue, for messages

waiting to be accepted and processed. There is one outgoing message queue for each resource, for messages waiting to be delivered. Sending a message is placing it on the incoming queue.

The message switcher is the heart of the message-passing kernel. It accepts messages from the incoming queue and delivers them from the outgoing queues. A message from the incoming queue is accepted by removing it from the queue. It is processed by distributing it (or a copy) to the outgoing queue for each resource that has registered interest in its type. A message from an outgoing queue is delivered by removing it from the queue and calling the delivery function of the corresponding resource. The various resources can coordinate message reception by setting flags in the message, since the messages are actually passed around via pointers, not copies. In particular, the first receiver can set a flag that is used by the other receivers to discard the message. The interleaving of acceptance and delivery is an unresolved design question. The program is written to allow two different styles of acceptance (one acceptance first, then deliver, or all acceptances first) and two different styles of delivery (deliver just one message before the next acceptance, or deliver all of them).

Finally, the style of processing is such that data items are often undefined when the program starts. There are functions (called "servers" in this program) that are invoked in those cases. The "code server" loads the code for a resource, using the appropriate registry to find it. The "type server" reads the message type definition, so the program can process it effectively. The "message server" reads the message interests, so that the message switcher can distribute it appropriately.

The message delivery function for a resource is called whenever a message is to be delivered. The function must dispatch the message data to the appropriate part of the resource, presumably by calling different functions according to the message type.

4.2 Core Layer

The core layer contains the functions that process the wrappings. There is a set of knowledge bases that organize problem solving information, a set of functions that process those knowledge bases, and a "Study Manager" that organizes the problem solving process.

The Study Manager is the behavioral foundation of the core layer. It runs through a sequence of operations that we have decided represents many kinds of problem solving activity. This sequence is described in Section refstmgr-sec: (1) pose problem; (2) interpret problem, which includes identifying the relevant resources, negotiating information (this is appropriate for systems with cooperative entities that interact to solve problems), resolving resource selection, advising poser on selection, and adapting the resource to the problem conditions; (3) apply resource; and (4) assess result.

We will describe the different knowledge bases first, then show how the processes in this sequence interact with them.

There are really only three kinds of entities considered by this program. There are "problem"s, that represent problem statements, "information", that represents associated data structures, either as part of the problem statement, the context in which the problem is being considered, the resource adaptation, or the current state of solution of the problem, and "resource"s, which represent the resources.

The Planner Knowledge Base is divided into three parts in *vsim*. The Resource Applicability Knowledge Base (RAKB) maps problems and information available into resources that may apply to the problem. The Information Source Knowledge Base (ISKB) maps information required into resources that may provide the information. The Required Information Knowledge Base (RIKB) maps problems into information that may be required to state the problem fully and precisely, or to solve the problem. Finally, the Wrapping Knowledge Base (WKB) contains all the wrappings.

There are other structures constructed dynamically during the processing. The Study Log records all activity of the wrapping processors. It is intended to serve as a coordinating mechanism for the problem context, but we do not yet know all of its uses. The Design Tree records the current state of the problem solving process, with a new node for each major operation (or sequence of minor operations), so that the user can pursue several lines of investigation in parallel, and not get lost (there are graphical ways of looking at the tree, to get an overview of the full study). The Workspace contains current information, and serves primarily as a temporary storage space. Other informative data structures are being considered.

The Study Manager sequence is the heart of our problem solving strategy. It was described above, so we only note here which parts of the PKB are used. The "match" step uses the RAKB mapping, to determine its list of candidate resources. The other parts use the WKB.

The bottom of the recursion is some very simple processes that just read the different knowledge bases with syntactic searches. The poser reads the RIKB to determine what information is needed to fully specify the problem. The matcher reads the RAKB to make its candidate list, or it reads the ISKB to determine where to find the needed information. The negotiator and selector use the WKB. The executor expects to use a simple function call, possibly with dynamic loading, implemented as sending a message to the resource.

Program *vs* is a partial implementation of the Core Layer processing. It includes the Study Manager and its Study Log, the various knowledge bases, and the use of *dld* to load object code for functions cited by name.

4.3 Resource Layer

The resource layer will contain many tools, databases, and other resources that can be used in different application domains. It is much like the libraries of tools in other environments (the difference is in the way the wrappings support access to the library). Below we briefly review a number of different types of resources in this layer that we have worked on. This includes mathematical tools, discrete event simulations, rulebases of various sorts, and screen languages.

4.3.1 Resources

Mathematical tools are a large part of most resource layers [21], [22]. Many mathematical methods are available in good numerical programs (see [35] for a widely used set), that have the usual problems with rigid interfaces. Making these tools more readily available was one of the original drivers in the wrapping research.

Many programs for these methods have been collected; they are currently being described in wrappings which we are constantly experimenting with. For example, there is a well-described multivariate optimization program [10] that contains functions for optimizing under several different conditions of available information. In addition, there are two good ordinary differential equation solvers (from [35]; a description is in [24]), one of which implements the usual Runge-Kutta method and the other the Bulirsch-Stoer method (which interpolates with rational functions instead of polynomials). Both are implemented in a very nice and modular way, and have exactly the same data and protocol interface. Wrappings for these functions are being constructed.

Discrete event simulation is an important modelling activity. We have defined a set of functional roles that support simulation, and are incorporating an existing implementation of those functional roles [24]. Experimental design packages, statistical analyses, and various plotting programs will be included here. Wrappings for these functions are being constructed, as well as simulation generators for the application area of communication network analysis.

In addition to the already-utilized rule inferencing available in the *Vehicles* System, we have been exploring the incorporation of other expert system shells. Rule-based inference is an important programming paradigm; the expert system shell CLIPS [8] is designed to be incorporated into a larger system.

Lastly, like other resources, we intend *vsim* to have a choice of user interfaces. There will be screen languages used to define interactions with the user, both the presented screens and options (buttons, menus, etc.) and the expected responses (menu item list generators, constraints on typed input). Many such languages exist (see [31] for one example); we are in the process of selecting a few of them to use.

Also, clustering algorithms, sorting algorithms and other UNIX utilities are also being made available in this environment.

4.3.2 Service Requests

The first point is that flexible combinations of resources means that the service requests are sent to the Study Manager, not to other resources. This frees the individual resources from having to know where to get the information they need, but it also requires there to be an explicit description of that information so that it can be found.

Service requests can take any of several forms: they can be information requests, processing requests, or connection requests (to connect one resource to another to bypass the wrapping processes). In this context, we can view a computational process as a provider of information (i.e., the process "output"), or as a modifier of the current state (i.e., the process "effect"). Under some conditions it is preferable to consider what the process does (the process "behavior") instead of its output or effect. In particular, such a description is more appropriate when it is a control structure for running other processes.

For these definitions, *output* is always a data structure (i.e., an instance of a data type), though it can be quite complicated (e.g., a simulation scenario file), *effect* is a change to a data structure (changes include creation and deletion of components and entire structures), and *behavior description* is an organized collection of event descriptions.

4.4 Application Layer

The application layer contains application domain models and processes. We briefly describe two current example applications.

ICE is the ISCS Concept Evaluator. ISCS is the Integrated Satellite Control System, which is a project that intends to combine most or all system-unique satellite control stations with the existing satellite control network. Various cost savings are expected to result. The ICE is intended to help evaluate different configurations of the control network, by looking at both performance under many normal and anomalous conditions, and at costs of development and maintenance. The approach is to use the wrappings to help integrate communication system performance models, ground contact scheduling models, various anomaly models, and development and maintenance cost models. Some of these models are already in use; some are being developed for this application. The integration problem here is to understand what the models mean, and to change the corresponding programs as little as possible.

For this application, we wrote a simple menu interpreter, with a simple menu definition syntax: a menu is a tree of nodes, with non-terminal nodes representing subsidiary menus and terminal nodes representing functions. Each line specifies an action to take after selection (e.g., "done"

with this menu or "wait" for more selections) Then all ICE interaction with the user is through the menus. The separate models have their own user interfaces. All interaction among models is through the knowledge bases (all information requests, responses, and announcements). There are no direct function calls at all. This frees the individual programs from knowing how the service they need will be provided. They only need to know how to ask for it. The knowledge bases are used to coordinate the service requests with the service providers. We use UNIX network services to coordinate the external programs (this part is what the Message Layer will replace).

DAW is the Debris Analysis Workstation. Space debris is a major problem for any space program, for determining where and when it is safe to launch, what orbits are safe from intercepting existing debris clouds, where a given space object is likely to land, and where to safely dispose of a dead satellite so it does not cause a hazard to other spacecraft or to people or places on the ground. There are several models for different aspects of the problem, including an impact model that describes collisions and explosions, a debris cloud propagation model that describes how the cloud changes in time, a lifetime model that describes how long suborbital cloud particles stay up, and a footprint model that describes where things hit when they hit. The approach is to use the wrappings to help integrate these models and others, to provide a debris analyst with access to all the models in a coordinated way. There are several alternatives for each of these models, and part of the interest here is in using the wrappings for model development.

5 Variable Resolution Modelling

In Variable Resolution Modelling (VR), we are mainly concerned with large-scale simulations, in which the resources are individual simulations, analysis tools, hardware and emulation components, and humans.

There are many very large simulations and emulations being proposed and designed for system-level modelling of military systems (note on terminology: we call all of these programs *simulations*, without presuming that they include only software, and without presuming that they are necessarily low fidelity models). These large simulations model many different aspects of systems at many different levels of fidelity (according to the main focus of the application).

Our wrapping approach to integration leads to a new methodology for requirements definition, system design and specification (for VR and other simulation models). Systems built according to this methodology support "problem-based configuration": The wrappings are used to collect the appropriate resources (or promises of connection to them at appropriate time) and to construct the links among them. The software integration support provided by wrappings can ensure that the resulting simulation programs are flexible enough to allow evolutionary changes in the system, and robust enough to allow many different models of parts of the system to be studied in a common framework.

Of course, there is a hard problem that we do not address: that of ensuring that the individual simulation programs use consistent data objects. One simple version of this problem is making sense of an interface between two different resources operating at different levels of detail.

5.1 Overview of Concerns About VR Modelling

In our work, we have seen increasingly complex and large simulations proposed. The interaction among system elements in a large system is an extremely difficult coordination problem. Some of the simulation programs we are concerned about are intended, among other things, to study architectures supporting the coordination among several types of system components. One is supposed to, for example, "... establish and refine the ... integration process, demonstrate functional inte-

gration of the element activities". The development of software and hardware components of any large system requires appropriate testing, during design, development, and manufacturing. Some of the simulations we are studying, are intended to allow component models to be tested in a realistic environment, from both the software algorithm and mission data viewpoint and from the hardware and electronics viewpoint.

Each of the example simulations we have seen (of the above types and others) has an extremely rigid interconnection architecture:

- The integration is only at the level of passing certain data structures around. There seems to be no higher-level model control functions (resources that help organize the simulation input and output, monitor the behavior during execution, analyze the results after execution, etc.).
- There does not seem to be any development and maintenance strategy for the software.
- The simulation architectures described are each just one model. Each is an existence proof, that a certain architecture can "work" (according to one definition of "work"), but nothing can be studied with it.

There are many medium- and high-fidelity individual models that have been developed for these and other applications, but it appears that each is being rewritten to fit the proposed modelling systems, or even written over again from scratch. It is our belief that this problem results in part from an integration concept that is not sufficiently robust to accommodate diverse models, and in part from a lack of specified interaction strategy for the component models when they were first designed or developed.

Typically, large simulation programs can test just one hypothesis:

> How does this configuration perform?

under a variety of variously sophisticated scenarios. This question is then answered relative to some measures that were defined to evaluate performance.

Few (if any) of the proposed and existing designs for large simulation programs can compare hypotheses, do trade-off studies for different system architectures, study system effectiveness from many points of view, or many other required analyses mentioned below. For each of the simulations, many architectural decisions about the system under study need to be made before the simulation can run; these decisions cannot be studied within the simulations; there is no facility for studying those decisions outside the simulations. The result is that important architectural decisions must be made with no supporting model, only an after-the-fact test. We are concerned that these and other large systems are not being modelled; they have been designed from scratch without an integration concept, as if it is sufficient for integration simply to have the right pieces.

5.2 Modelling Requirements

Our research has involved building models to study proposed software and hardware architectures for simulations, and to study the roles of the simulations: what questions they are intended to answer, what questions they will probably be able to answer, and especially what important questions are not being addressed. Instead of trying one configuration at a time, we want to study and compare algorithms, models, architectures, design decisions, etc.. There is seldom any discussion of the software and hardware architecture needed to support the required studies.

We need to describe the purposes of the models, the questions that the user wants answered and the decisions that the software supports, and then map these user needs and modelling requirements into the requirements for the underlying software architecture.

We list below a few of the modelling requirements for large simulations. The lists are by no means expected to be exhaustive; they merely illustrate some important aspects of building effective modelling environments.

First, let's take a very general view of the specific (or domain-dependent) requirements for three types of very large simulations. Hence, for a simulation intended to model the integration of elements in a large system, we would need:

- Capabilities of models in the application domain: scenario development and elaboration; algorithms for surveillance, tracking, command and control; engagement models; environmental effects on surveillance and tracking; effects of component losses on command and control
- What effectiveness measures will be used? surveillance effectiveness, tracking performance, command and control failures (the usual tracking and and command measures)

These are all performance measures; there are others, such as cost, that are quite important for effectiveness, and especially for evaluating design trade-offs (though they are also harder to model).

For a large simulation (emulation) supporting the testing of a large system, we would need:

- Capabilities of models in the application domain: development and elaboration of real-time scenarios; realistic local test article environment, at both the electronic level for the hardware, and at the information level for the software; anomalous data insertion (to test robustness)
- What effectiveness measures will be used? correctness of operation, speed of operation, failure rate, degradation behavior during overload (the usual testing measures)

These are all performance measures also.

In large simulations modelling complex communication networks, one would need:

- Capabilities of models in the application domain: creating, disseminating, and accepting messages; models of the transmission mechanisms (to include losses and other reductions in service); traffic models and interference from excess loading; network management algorithms
- What effectiveness measures will be used? throughput, timeliness, traffic loading, required buffer size, response to failures (the usual communication system measures)

These are all performance measures also.

The rest of this section contains some requirements that hold for each of these types of large simulations.

What questions must the simulation models answer?

- Is this (one) architecture more effective than that (another)?
- What is the effect of this component change? (e.g., new antenna technology, new surveillance algorithm)

- What is the effect of this configuration change? (e.g., scaled-down or eliminated program, changed mission)

There is a long history of measuring only the performance aspects of a system model, and leaving the cost to be worked out later. Then the design changes forced by cost compromises remain unmodelled, which sometimes leads to big surprises in the fielded systems. Modelling cost explicitly allows trade-offs between the performance and costs associated with design modifications.

Some of the required analysis tools include:

- trade-offs,
- comparison of alternatives,
- parametric study,
- sensitivity analysis, and
- optimization.

Programs for each of these kinds of analysis exist. The combination of these mathematical algorithms with our flexible approach to using data from combinations of models leads to a powerful analytical capability in a modelling environment.

Some of the required documentation tools include explicit recording of:

- sources for modelling choices,
- all system activity during execution, and
- system inputs and assumptions with results.

All of these tools are important to ensure that the results obtained from any of the simulation programs are labeled with appropriate restrictions.

What software and hardware capabilities are required to support these studies?

- repertoire of models, tools, services
- explicit architecture models
- explicit treatment of multiple models: selection, comparison and others

Any large simulation is a very large software and hardware system, so modelling of them is also essential. We propose to model various software and hardware architectures for the simulations, to study their effectiveness as modelling environments. We need appropriate questions to define measures of effectiveness (What is the simulation going to be used for? Who is going to use it? e.g., nationally-distributed contractor teams implies certain networking and data repository requirements) We need requirements specifications for the simulations as systems: not just the individual models, but also the software and hardware support for integrating those models. Then the simulation architecture can be designed accordingly.

6 Conclusions

In this paper, we discussed some of the software requirements necessary to support large complex modelling projects. We paid special attention to the flexibilities, and the associated integrative mechanisms we have found to be necessary in a modelling environment. We introduced our *Wrapping* approach to providing flexible, extendible, and integrated modelling environments. Our methodology requires both understanding what should be in the wrappings and devising algorithms for processing the wrappings. We have been building an experimental implementation of the wrapping approach to study both problems. We are convinced that the Study Manager and its supporting knowledge bases are one key to success for these ideas, and that the layered software architecture and message passing are another key to success for programs implementing these ideas. Wrappings are under construction for several generally useful capabilities, including mathematical functions (e.g., equation solving, constrained optimization), and discrete event simulations.

References

[1] Alfred V. Aho, Jeffrey D. Ullman, The Theory of Parsing, Translation, and Compiling, Vol. I: Parsing, Prentice-Hall (1973)

[2] Kirstie L. Bellman, "The Modelling Issues Inherent in Testing and Evaluating Knowledge-based Systems", *Expert Systems With Applications J.*, Volume 1, pp. 199-215 (1990)

[3] Kirstie L. Bellman, "An Approach to Integrating and Creating Flexible Software Environments", *Proceedings of SOAR '91: The 1991 Symposium on Space Operations, Automation and Research*, 9-11 July 1991, Houston, Texas (1991)

[4] Kirstie L. Bellman, "An Approach to Integrating and Creating Flexible Software Environments Supporting the Design of Complex Systems", pp. 1101-1105 in *Proceedings of WSC '91: The 1991 Winter Simulation Conference*, 8-11 December 1991, Phoenix, Arizona (1991)

[5] Kirstie L. Bellman and April Gillam, "A knowledge-based approach to the conceptual design of space systems", pp. 23-27 in *Proceedings of the 1988 SCS Eastern MultiConference*, March 1988, The Society for Computer Simulation (1988)

[6] Kirstie L. Bellman and April Gillam, "Achieving Openness and Flexibility in Vehicles", pp. 255-260 in *Proceedings of the SCS Eastern MultiConference*, 23-26 April 1990, Nashville, Tennessee, Simulation Series, Volume 22(3), SCS (1990)

[7] Kevin A. Broughan, "SENAC: A High-Level Interface for the NAG Library", ACM Trans. Mathematical Software, Volume 17(4), pp. 462-480 (December 1991)

[8] Chris Culbert, "CLIPS Reference Manual (Version 4.2)," NASA Johnson Space Center (April 1988)

[9] Paul K. Davis, Reiner K. Huber, "Variable-Resolution Combat Modeling", RAND Note N-3400-DARPA, RAND Corp. (1992)

[10] J. E. Dennis, D. M. Gay, and R. E. Welsch, "Algorithm 573 – An Adaptive Nonlinear Least-Squares Algorithm", ACM Trans. Mathematical Software, Volume 7, pp. 369-383 (1981)

[11] Michael R. Genesereth, "An Agent-Based Framework for Software Interoperation", pp. 359-366 in *Proceedings DARPA Software Technology Conference*, 28-30 April 1992, Los Angeles (April 1992)

[12] April Gillam, "A knowledge-based approach to planning the design of space systems", *Proceedings 1989 SCS Eastern Multi-Conference*, The Society for Computer Simulation (1989)

[13] Michael M. Gorlick, Rami R. Razouk, "Using Weaves for Software Construction and Analysis", *Proceedings 13th International Conference on Software Engineering*, 13-16 May, 1991, IEEE (1991)

[14] Susan L. Graham, Michael A. Harrison, Walter L. Ruzzo, "An Improved Context-Free Recognizer", ACM Trans. Programming Languages and Systems, Volume 2(3), pp. 415-462 (July, 1980)

[15] W. Wilson Ho, "Dld: A Dynamic Link / Unlink Editor", preprint, U. C. Davis (1990)

[16] W. Wilson Ho, Ronald A. Olsson, "An Approach to Genuine Dynamic Linking", CSE-90-25, Dept. Computer Science, U. C. Davis (1990)

[17] C. A. R. Hoare, *Communicating Sequential Processes*, Prentice-Hall (1985)

[18] E. N. Houstis, J. R. Rice, R. Vichnevetsky (eds.), *Expert Systems for Scientific Computing*, Proc. Second IMACS Int. Conf. on Expert Systems for Numerical Computing, 24-26 April 1990, Purdue University, North-Holland (1992)

[19] Christopher A. Landauer, "Table Parser Description", preprint, Pattern Analysis and Recognition Corp. (1977)

[20] Christopher A. Landauer, "Graphics for Simulation Model Definition", pp. 127-134 in *Proceedings of the 1985 IEEE/NBS Trends and Applications Conference*, May 1985, Silver Spring, Maryland (1985)

[21] Christopher Landauer, "Wrapping Mathematical Tools", pp. 261-266 in *Proceedings of the 1990 SCS Eastern MultiConference*, 23-26 April 1990, Nashville, Tennessee, Simulation Series, Volume 22(3), SCS (1990)

[22] Christopher Landauer, "Wrapping Mathematical Tools", *Proceedings of the 1990 Symposium on the Interface (between Computer Science and Statistics)*, 17-19 May 1990, East Lansing, Michigan (1990)

[23] Christopher Landauer, "Correctness Principles for Rule-Based Expert Systems", *Expert Systems With Applications J.*, Volume 1, pp. 291-316 (1990)

[24] Christopher Landauer, "Incorporating Simulation into a Design Environment", pp. 1180-1186 in *Proceedings of WSC '91: The 1991 Winter Simulation Conference*, 8-11 December 1991, Phoenix, Arizona (1991)

[25] Christopher Landauer, "Integrated Software Environments", pp. 164-170 in *Proceedings of CAIA '92 The 8th Conference on AI Applications*, 2-6 March 1992, Monterey, California (1992)

[26] JinTae Lee, Thomas W. Malone, "Partially Shared Views: A Scheme for Communicating among Groups that Use Different Type Hierarchies", ACM Trans. Information Systems, Volume 8(1), pp. 1-26 (January 1990)

[27] Barbara Liskov Preliminary Design of the THOR Object-Oriented Database System, Programming Methodology Group Memo 74, Laboratory for Computer Science, MIT (March 1992)

[28] Michael Lucks, Ian Gladwell, "Automated Selection of Mathematical Software", ACM Trans. Mathematical Software, Volume 18(1), pp. 11-34 (March 1992)

[29] Pattie Maes, "Concepts and Experiments in Computational Reflection, pp. 147-155 in *Proceedings OOPSLA '87* (1987)

[30] Robin Milner, *A Calculus of Communicating Systems*, Lecture Notes in Computer Science 92, Springer (1980)

[31] Brad A. Myers, "A New Model for Handling Input", ACM Trans. Information Systems, Volume 8(3), pp. 289-320 (July 1990)

[32] Robert Neches, William R. Swartout, Johanna D. Moore, "Enhanced Maintenance and Explanation of Expert Systems Through Explicit Models of Their Development", IEEE Trans. Software Engineering, Volume SE-11(11), pp. 1337-1351 (November 1985)

[33] Dewayne E. Perry, Gail E. Kaiser, "Models of Software Development Environments", IEEE Trans. Software Engineering, Volume 17(3), pp. 283-295 (March 1991)

[34] Vaughan Pratt, "LINGOL - A Progress Report", preprint, MIT (January 1975)

[35] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press (1986)

[36] Purtilo, James M., "POLYLITH and Environments for Mathematical Computation", University of Illinois Dept of Computer Science, Report No. UIUCDCS-R-84-1135 (1984)

[37] Purtilo, James M., POLYLITH: An Environment to Support Management of Tool Interfaces, preprint, ACM 0-89791-165-2/85/006/0012 (1985)

[38] R. F. Rashid, "Threads of a New System", UNIX Review, Volume 4, pp. 37-49 (August 1986)

[39] Steven P. Reiss, Interacting with the FIELD environment, Software - Practice and Experience, Volume 20(S1), pp. S1/89-S1/115 (June 1990)

[40] Nicholas Rescher, Alasdair Urquhart, *Temporal Logic*, Springer-Verlag (1971)

[41] Izhar Shy, Richard Taylor, Leon Osterweil, "A Metaphor and a Conceptual Architecture for Software Development Environments," pp. 77-97 in *Proceedings of International Workshop on Environments*, 18-20 September 1989, Chinon, France, Lecture Notes in Computer Science 467, Springer (1989)

[42] Douglas R. Smith, KIDS: A Semiautomatic Program Development System, IEEE Trans. Software Engineering, Volume 16(9), pp. 1024-1043 (September 1990)

[43] SUN Microsystems, Network Programming Guide, Part Number 800-3850-10, Revision A of 27 March 1990, SUN Microsystems (1990)

[44] Donald O. Walter, Kirstie L. Bellman, "Some Issues in Model Integration", pp. 249-254 in *Proceedings of the SCS Eastern MultiConference*, 23-26 April 1990, Nashville, Tennessee, Simulation Series, Volume 22(3), SCS (1990)



JOINT MODELING AND SIMULATION SYSTEM

J-MASS
PURPOSE

To Develop A Standard Modeling Architecture And Simulation Support System For Use In The Development Of Validated Digital Models Of Threat And Friendly Systems And The Analysis Of Those Systems.

J-MASS
STANDARD ARCHITECTURE

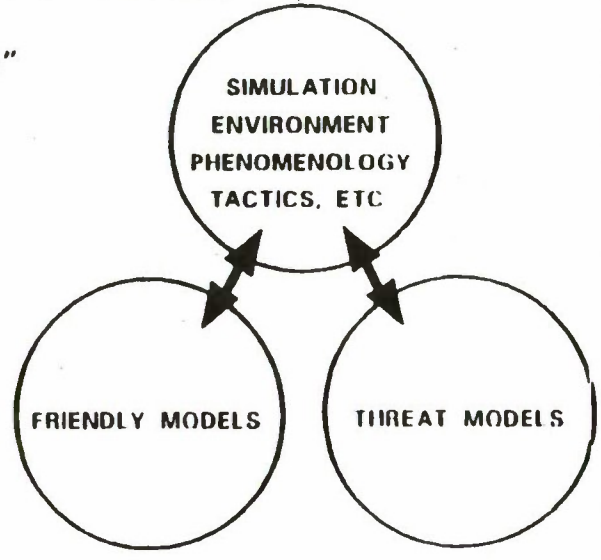
WHAT IS J-MASS?

- A FLEXIBLE MODELING AND SIMULATION SYSTEM THAT
  - IS INTERACTIVE AND OBJECT ORIENTED
  - ESTABLISHES STANDARDS FOR MULTIPLE APPLICATIONS IN DOD AND INDUSTRY
  - SUPPORTS TRADE-OFF ANALYSIS (COEAs, MAA, etc.)
- AN EFFICIENT SYSTEM THAT PROMOTES SOFTWARE REUSABILITY
  - USES DOD STANDARD SOFTWARE LANGUAGE - ADA
  - HAS A LIBRARY OF REUSABLE SOFTWARE
  - MAKES VV&A EASIER
- A PORTABLE SYSTEM THAT IS HARDWARE INDEPENDENT
  - WORKS ON DESKTOP-SIZED COMPUTERS
  - TAKES ADVANTAGE OF COMMERCIAL MARKETPLACE INNOVATIONS

WHAT IS J-MASS?

J-MASS IS A SYSTEM THAT:

- DEVELOPS MODELS USING STRUCTURED REUSABLE SOFTWARE OBJECTS (AT ANALYTIC THROUGH EMULATIVE LEVELS OF DETAIL)
- CONFIGURES SCENARIOS/SIMULATIONS ("ONE ON ONE" THROUGH "M ON N")
- REALISTICALLY EXECUTES THE SIMULATION
- SUPPORTS DETAILED ANALYSIS OF THE SIMULATION



WHICH RESULTS IN:

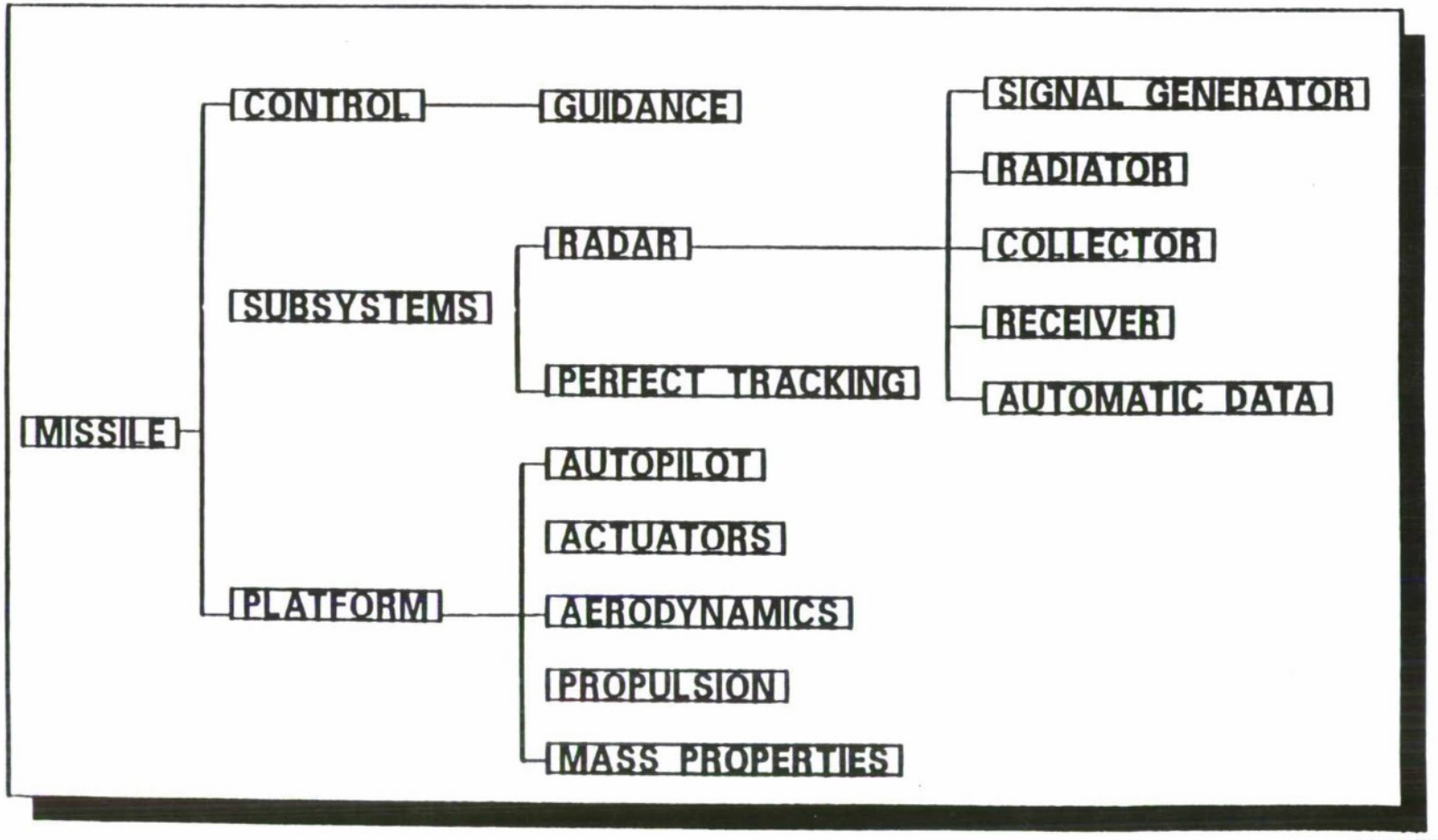
- IMPROVED MODELING AND SIMULATION CAPABILITY
- SUPPORT FOR TRADE-OFF ANALYSIS
- REDUCED COST AND TIME FOR MODELING AND SIMULATION
- MAKES VV&A EASIER AND COST EFFECTIVE

J-MASS PROVIDES A TRUE "APPLES-TO-APPLES" COMPARISON OF DIFFERENT MODELS

92-02JMA-01

433A

J-MASS
OBJECT ORIENTED



91-07JMS-11A

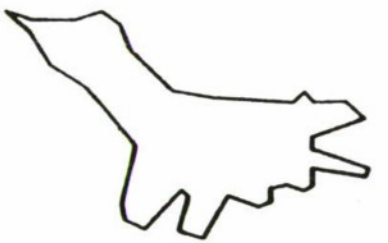
J-MASS
MULTIPLE LEVELS OF DETAIL

EMULATIVE



DYNAMIC

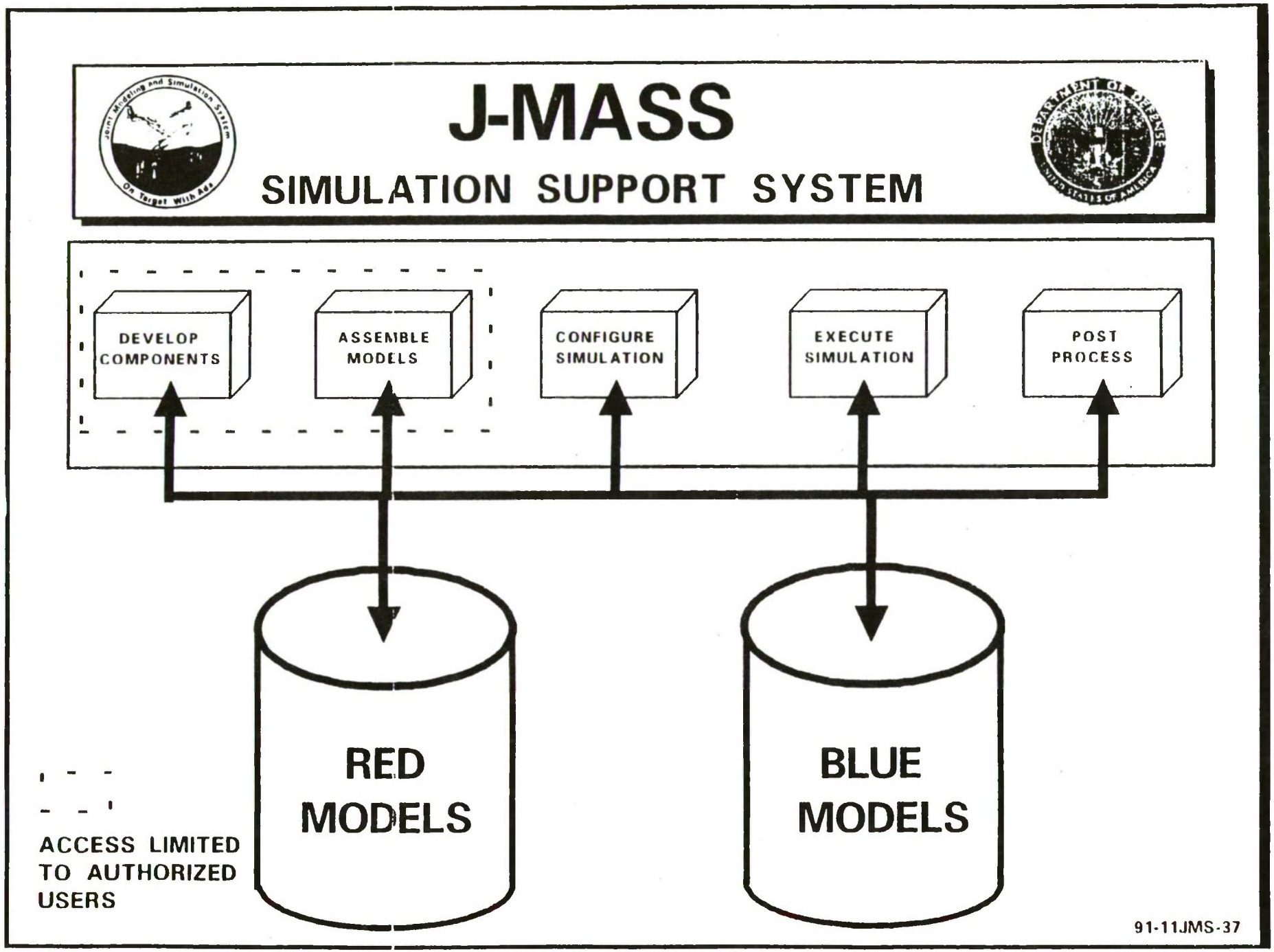
ANALYTIC



INCREASED FIDELITY

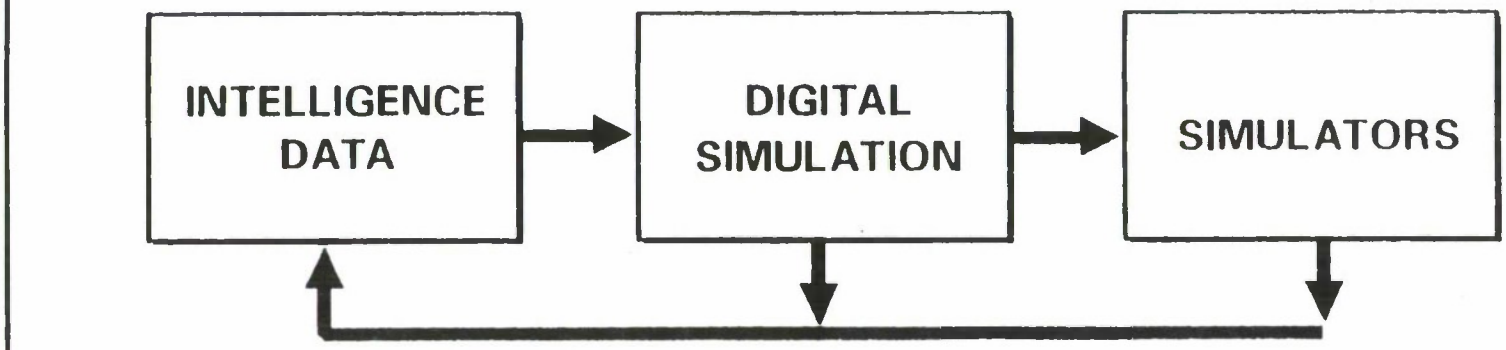
J-MASS
ANALYSIS LEVELS

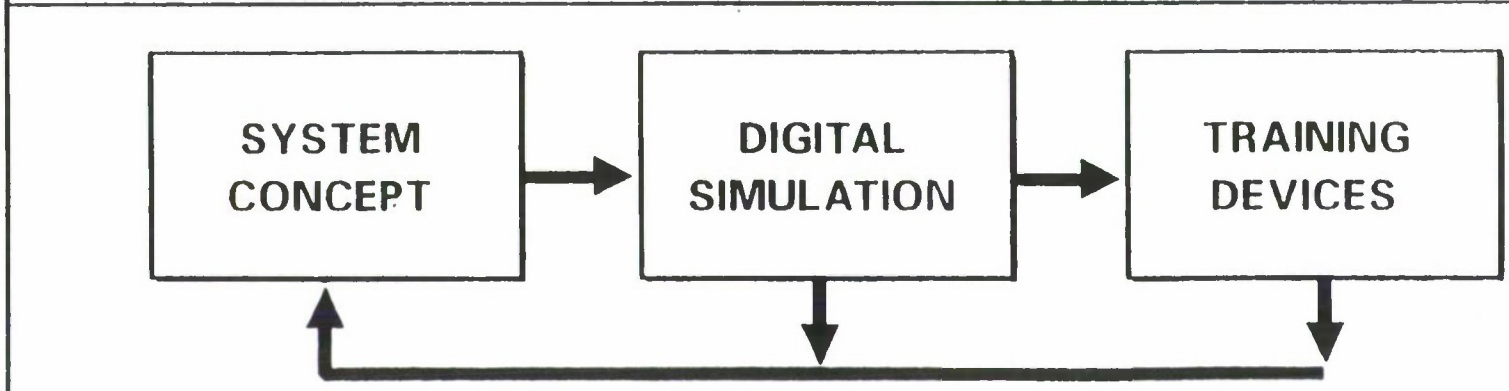
LEVEL	OBJECTIVE	SCOPE
I TECHNIQUE/ SYSTEM	SYSTEM PERFORMANCE	INDIVIDUAL SYSTEM/ COMPONENTS
II PLATFORM	DETERMINE WEAPON SYSTEM AVIONICS/ TACTICS	INTEGRATED WEAPON SYSTEM AND TACTICS
III MISSION	EVALUATE INVESTMENT TRADEOFFS AND FORCE SURVIVABILITY	MULTI-PLATFORM COMPOSITE FORCE
IV FORCE	ESTIMATE CONTRIBUTION TO FORCE CAPABILITY	JOINT OPERATIONS


J-MASS
SIMULATION SUPPORT SYSTEM
DEVELOP COMPONENTS
ASSEMBLE MODELS
CONFIGURE SIMULATION
EXECUTE SIMULATION
POST PROCESS
RED MODELS
BLUE MODELS
ACCESS LIMITED TO AUTHORIZED USERS
91-11JMS-37

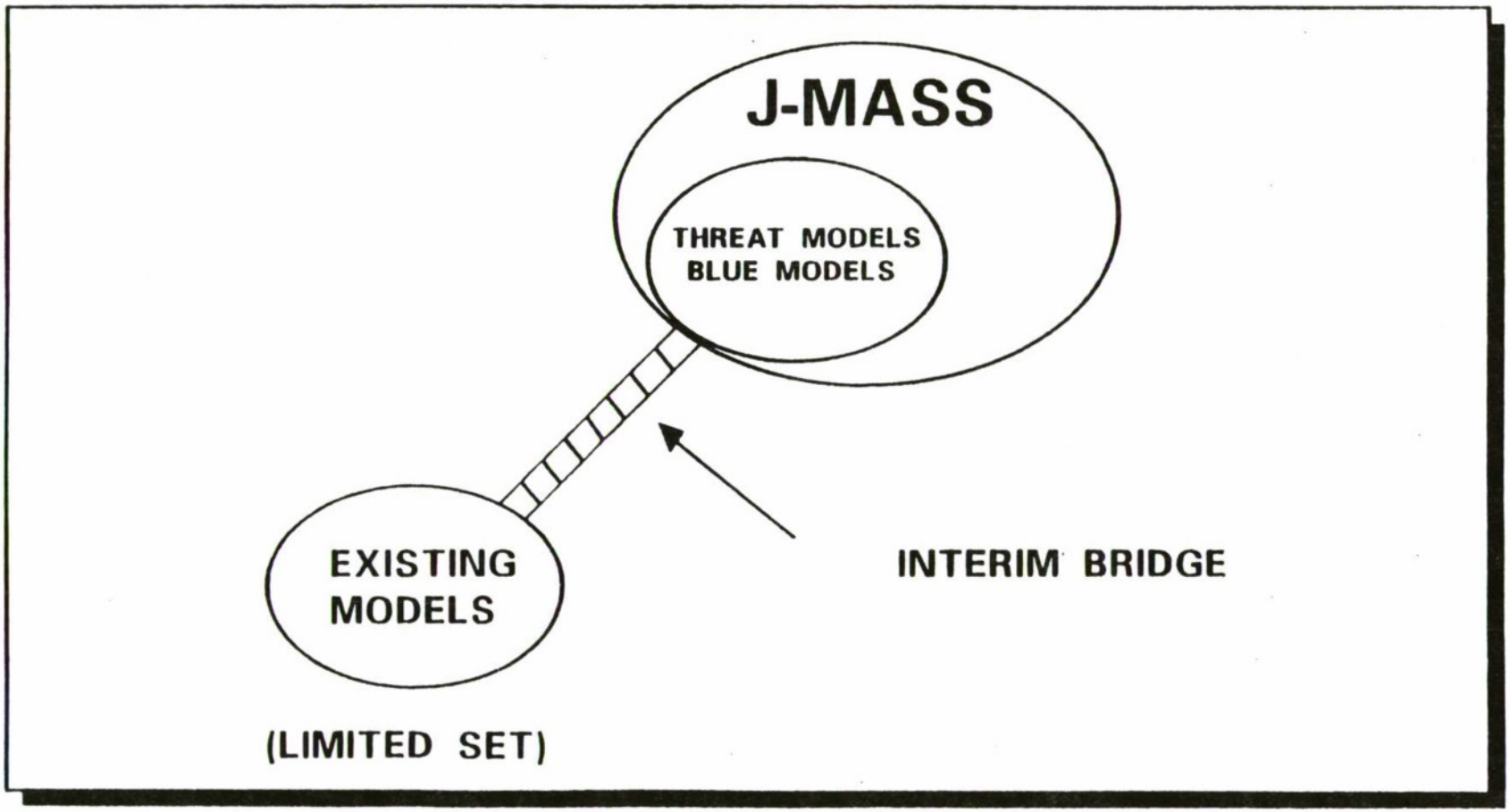
J-MASS

SIMULATOR AND TRAINING SUPPORT





J-MASS
"GRACEFUL" TRANSITION



436A

91-07JMS-19A

J-MASS
SUPPORTS

- TESTING COMMUNITY
  - •• DEVELOPMENTAL
  - •• OPERATIONAL
- RESEARCH AND DEVELOPMENT
- TECHNICAL ANALYSIS
- INTELLIGENCE ASSESSMENT
- TEST/DEVELOPMENT PROCESS

J-MASS
TEAM

SAF, AF

AFSC

ASD
Program Office

DDDR&E

DDDR&E(T&E)

Combined Test & Evaluation Investment Program

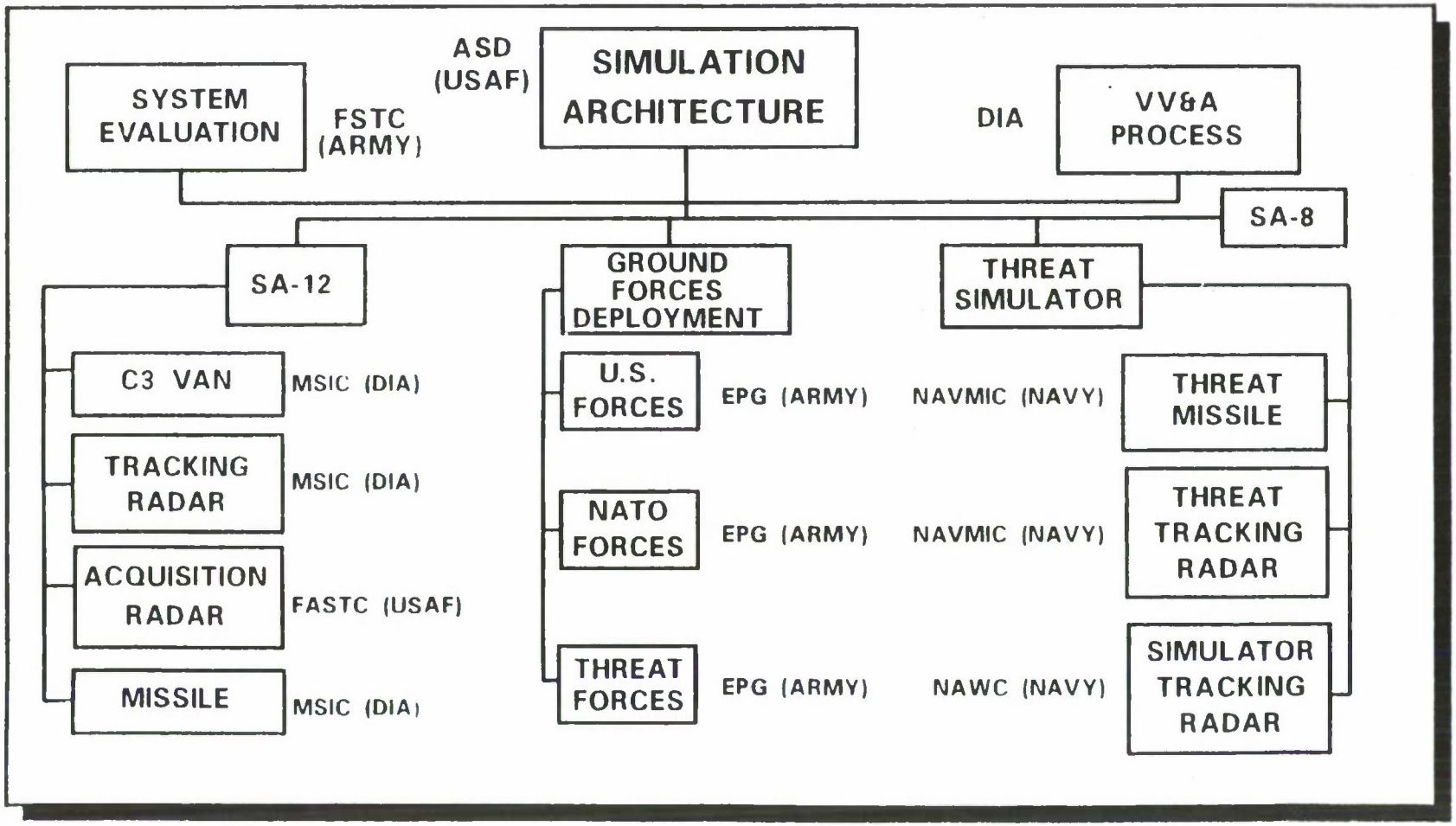
CROSSBOW-S Digital Simulation Steering Group

Tri-Service Development

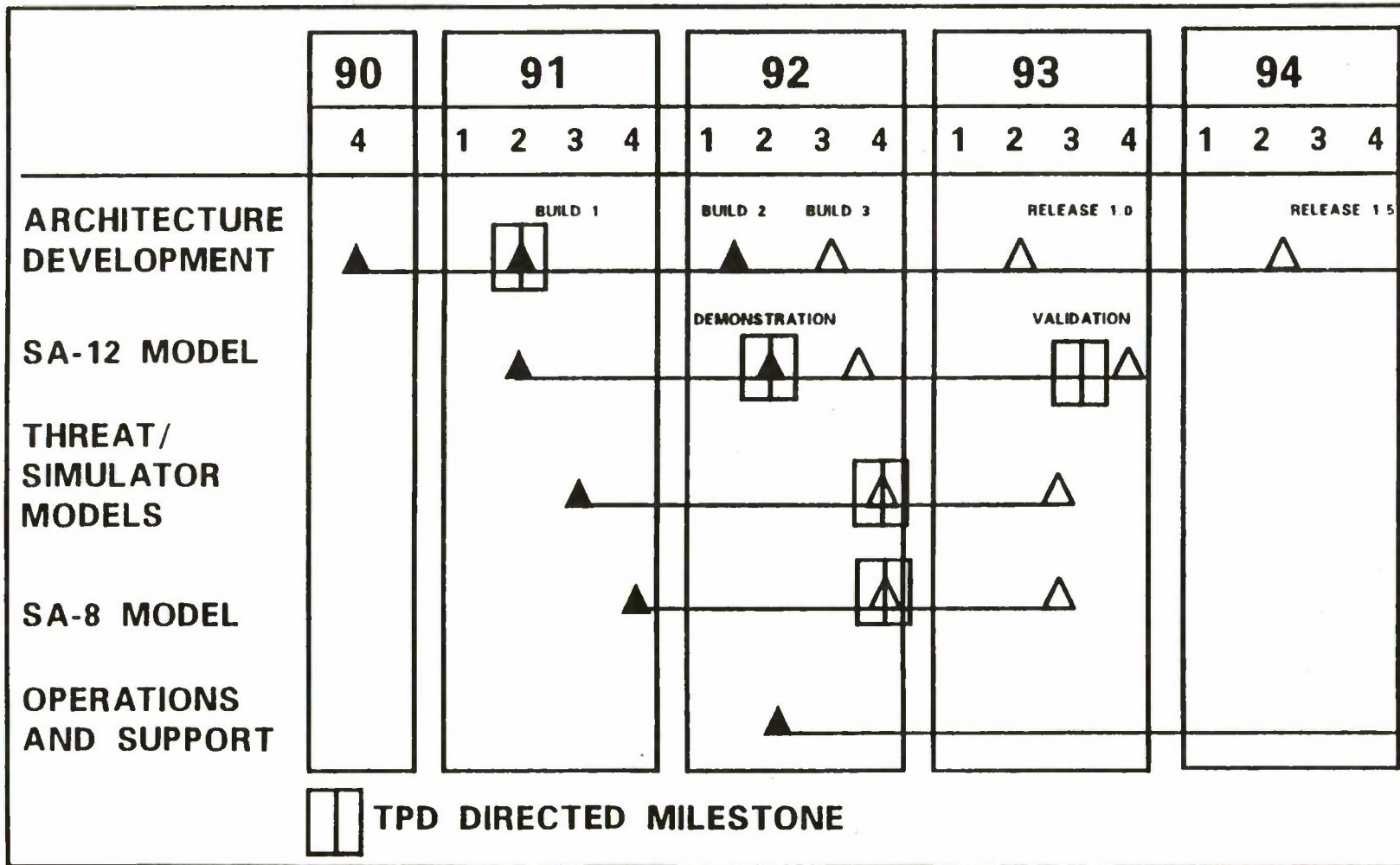
Tri-Service Support

91-07JMS-42

J-MASS
TRI-SERVICE APPROACH



J-MASS SCHEDULE



92-02JMA-07



J-MASS BENEFITS



EASIER, FASTER, HIGH QUALITY PROGRAMMING

- SA-12 TARGET TRACKING RADAR DEVELOPMENT USES 45 REUSABLE SOFTWARE OBJECTS TO DESCRIBE 2600 ACTUAL COMPONENTS
- NEW VERSIONS ARE EASILY CREATED IN MINUTES -- DATA DRIVEN
- OBJECTS READILY AGGREGATED INTO HIGHER LEVEL ASSEMBLIES; ASSEMBLIES INTO MODELS
  - VALIDATION "TRAVELS" WITH OBJECT
  - OBJECTS APPLICABLE TO OTHER MODELS

50% - 80% SOFTWARE REUSABILITY BETWEEN FIRST AND SECOND J-MASS MODELS; REUSABILITY WILL CONTINUE TO INCREASE

GREATER THAN 50% REDUCTION IN TIME AND COST TO PRODUCE MODELS; TIME AND COST WILL CONTINUE TO DECREASE

TRI-SERVICE INVOLVEMENT

92-02JMA-02

J-MASS USERS

ARMY	NAVY	AIR FORCE	DOD	INDUSTRY
FT. HUACHUCA	NWC	WRIGHT LABS	DIA	LOCKHEED
FSTC	NMIC	AFOTEC	MSIC	BOEING (PENDING)
		FASTC		TRW (PENDING)
		ASD ENGINEERING		BALL SYSTEMS
		EC SPO		
		SOFTECH	DYNETICS	
		SAIC	SPARTA	
		MCDONNELL DOUGLAS		

NUMEROUS OTHER GOVERNMENT AND INDUSTRY SITES WANTING TO USE/EVALUATING J-MASS

92-02JMA-10

439A



J-MASS

GOVERNMENT/INDUSTRY USER GROUPS



- SIMULATION
- GRAPHICAL USER INTERFACE
- PHYSICAL ENVIRONMENT MODELING
- REAL TIME APPLICATIONS
- EXTERNAL INTERFACE APPLICATIONS
- DATA MANAGEMENT

91-09JMA-02

J-MASS
RESULTS

- SIMULATION ARCHITECTURE,
- MODELING LIBRARY, AND
- VALIDATED THREAT MODELS TO SUPPORT MANY USERS

F-22
B-2
LX
AX
ECITF
ACETF
And Others

Engineering Development

Acquisition Process

Test & Evaluation

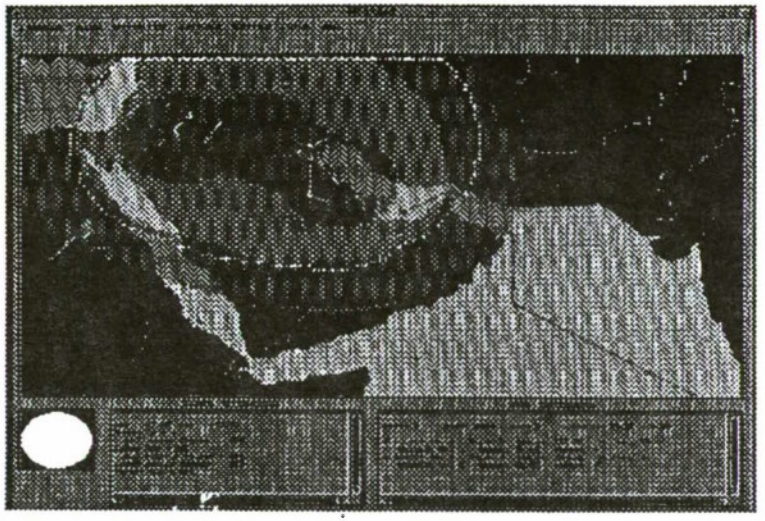
91-07JMS-41

STRUCTURED MODEL ENVIRONMENTS

Candace Conwell
Larry J. Peterson
Richard F. Freund

NAVAL COMMAND, CONTROL AND OCEAN SURVEILLANCE CENTER
RDT&E DIVISION

HIGH PERFORMANCE COMPUTING FOR C3I MODELING AND SIMULATION



OBJECTIVE

IMPLEMENT NEW METHODS OF OBJECT-ORIENTED MODELING THAT UTILIZE HIGH PERFORMANCE MACHINES TO AUTOMATE CAMPAIGN SIMULATIONS

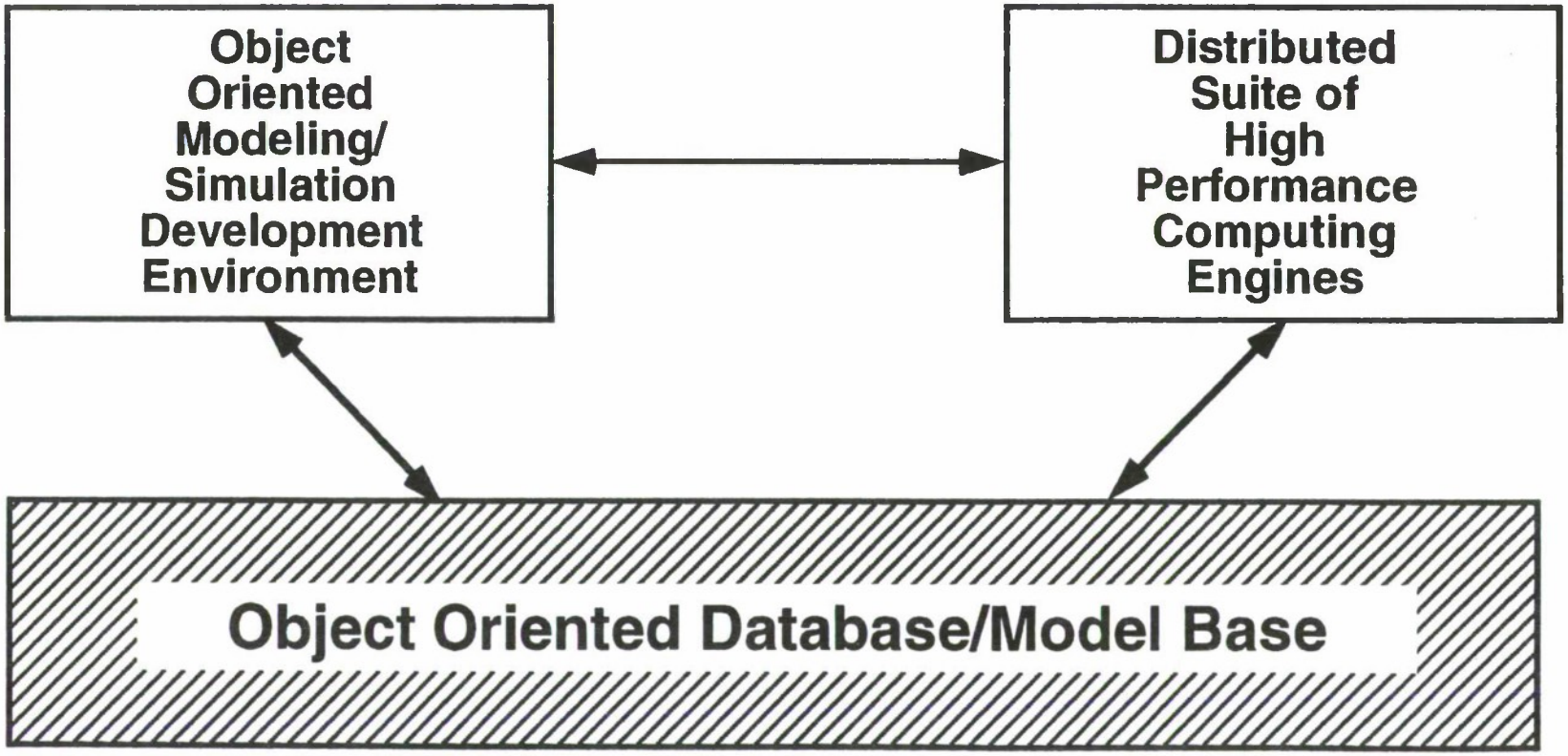
APPROACH/THRUST

- C3I MODEL IMPLEMENTED AS OBJECT ORIENTED MODULES ON SUPERCOMPUTER SUITE
- MODELS IMPLEMENTED USING OBJECT ORIENTED CONSTRUCTS
- IMPLEMENT OBJECT ORIENTED MODELS AND OBJECTS INTO DIVERSE GRANULARITY CAMPAIGN SIMULATIONS
- C3I OBJECTS MAINTAINED IN OBJECT ORIENTED DATABASE
- TRANSITION TECHNOLOGY TO MILITARY OPERATIONS SYSTEMS

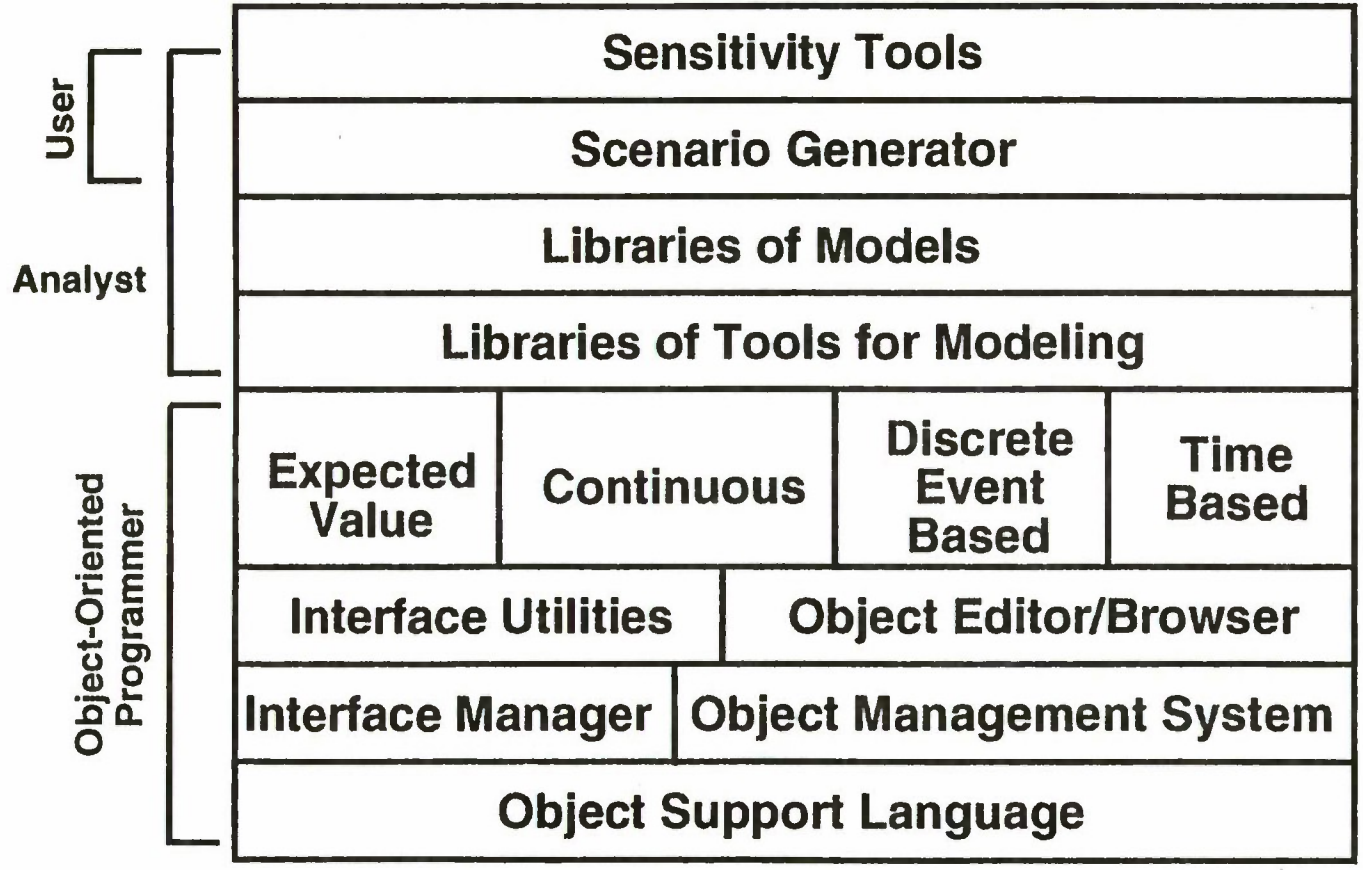
PAYOFF

- OBJECT ORIENTED APPROACH WILL EASE PROGRAM DEVELOPMENT, MODIFICATION AND MAINTENANCE
- USE OF PARALLEL SUPERCOMPUTERS WILL GREATLY IMPROVE PERFORMANCE AND GRANULARITY OF MODELS
- BETTER MODELS WITH FASTER RESULTS WILL IMPROVE EFFECTIVENESS OF MILITARY PLANNING

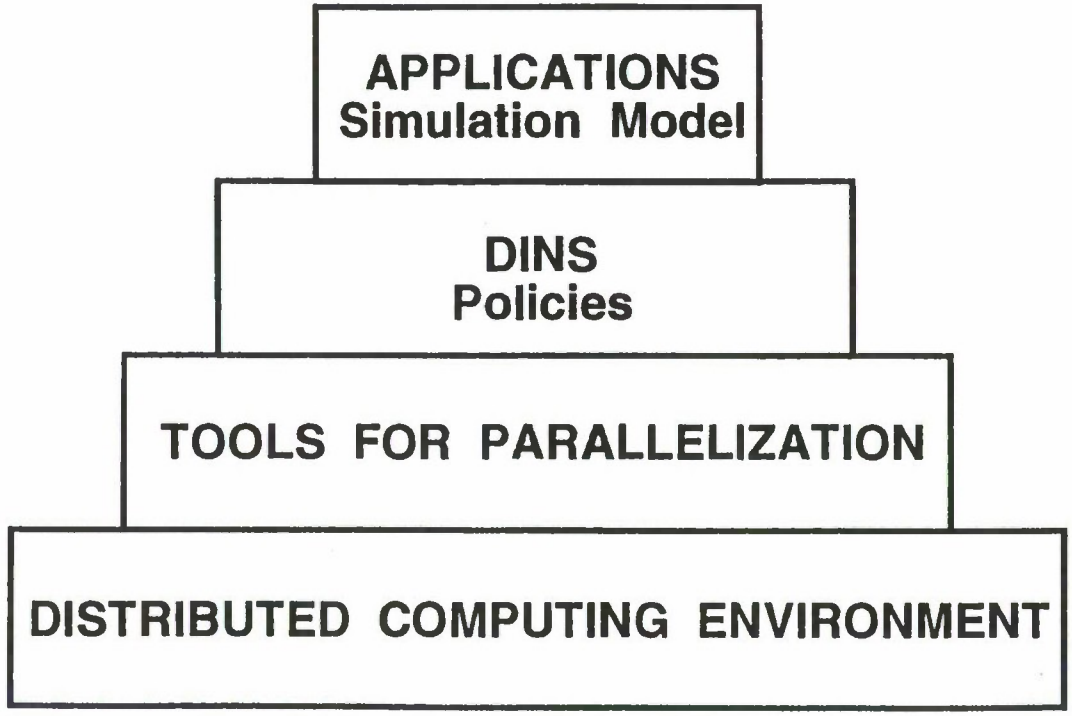
Concept

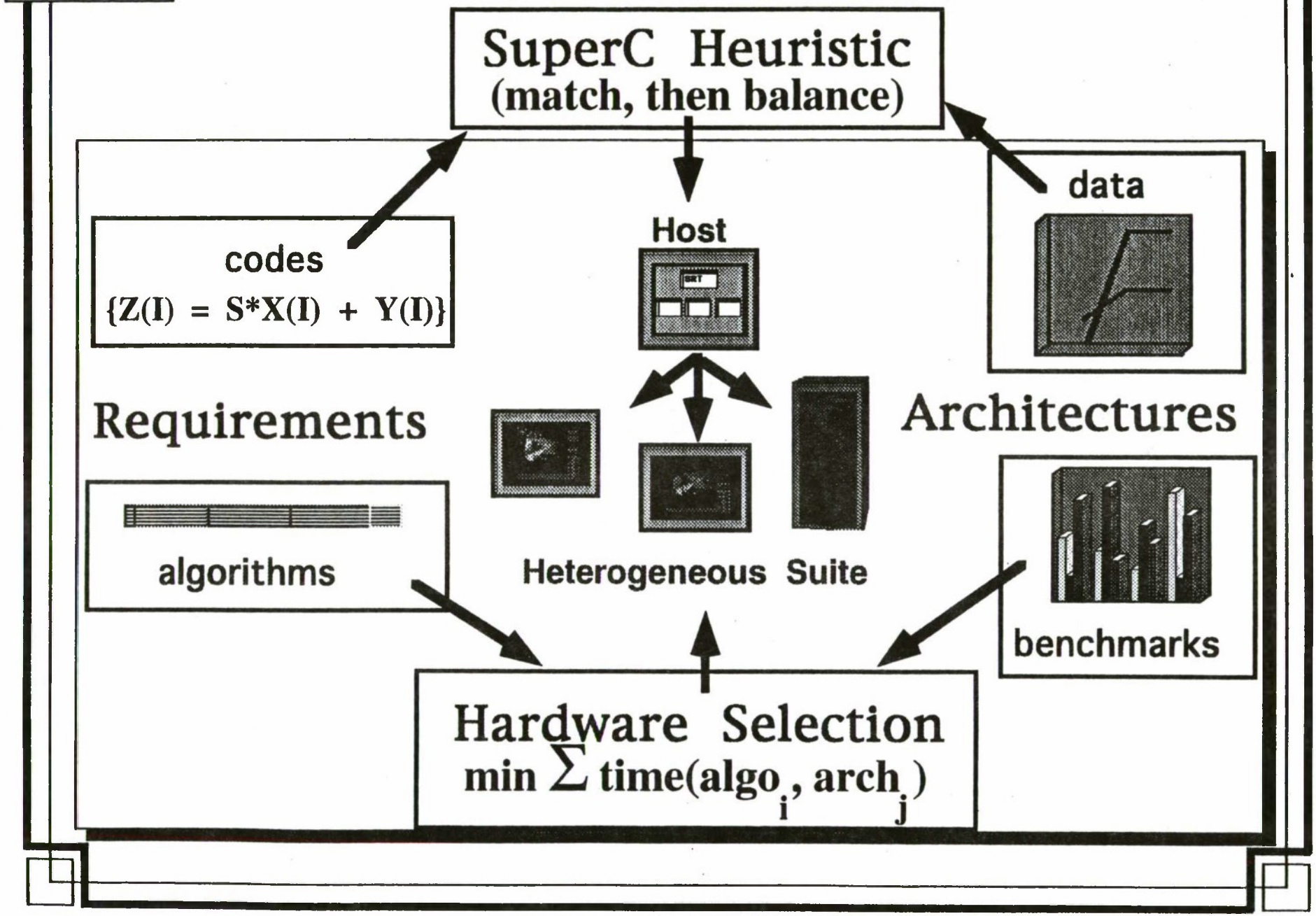

Object Oriented Modeling/ Simulation Development Environment
Distributed Suite of High Performance Computing Engines
Object Oriented Database/Model Base

Layering in the System Structure



Layering in an HPC Environment



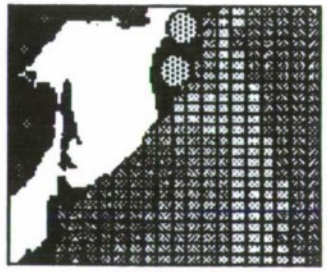

SuperC Heuristic
(match, then balance)
codes
{Z(I) = S*X(I) + Y(I)}
Host
data
Requirements
Architectures
algorithms
Heterogeneous Suite
benchmarks
Hardware Selection
min Σ time(algo_i, arch_j)

Superconcurrency Example

Single pre-Strike

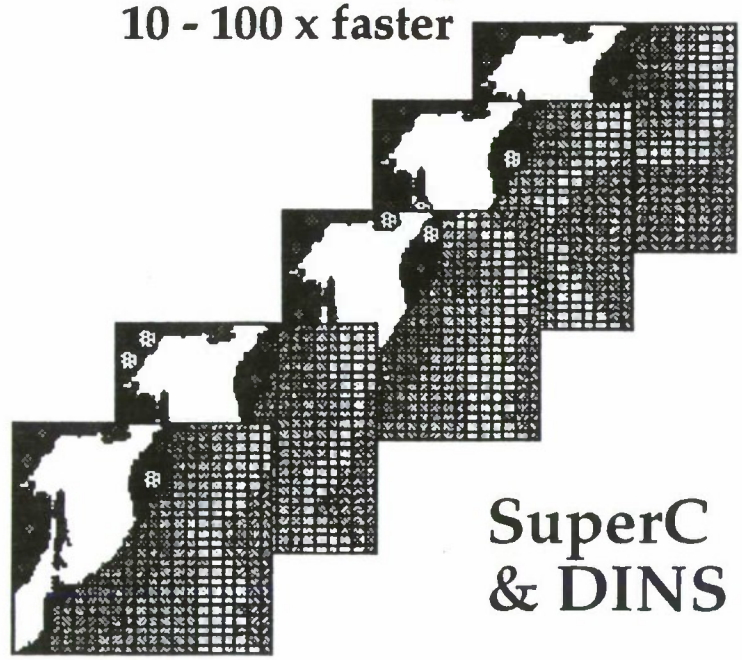


Conventional Computers



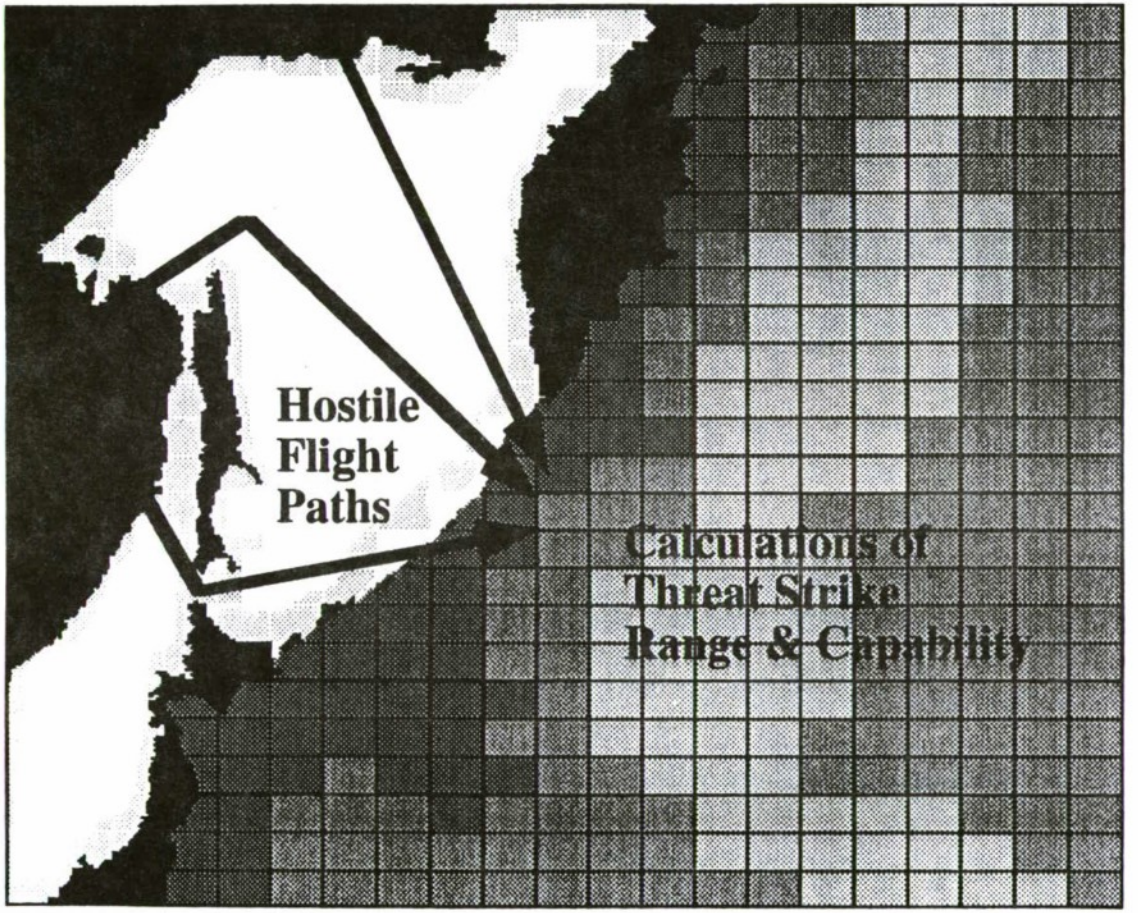
Single post-Strike

Multiple pre- & post-Strike:
Simultaneously
10 - 100 x faster



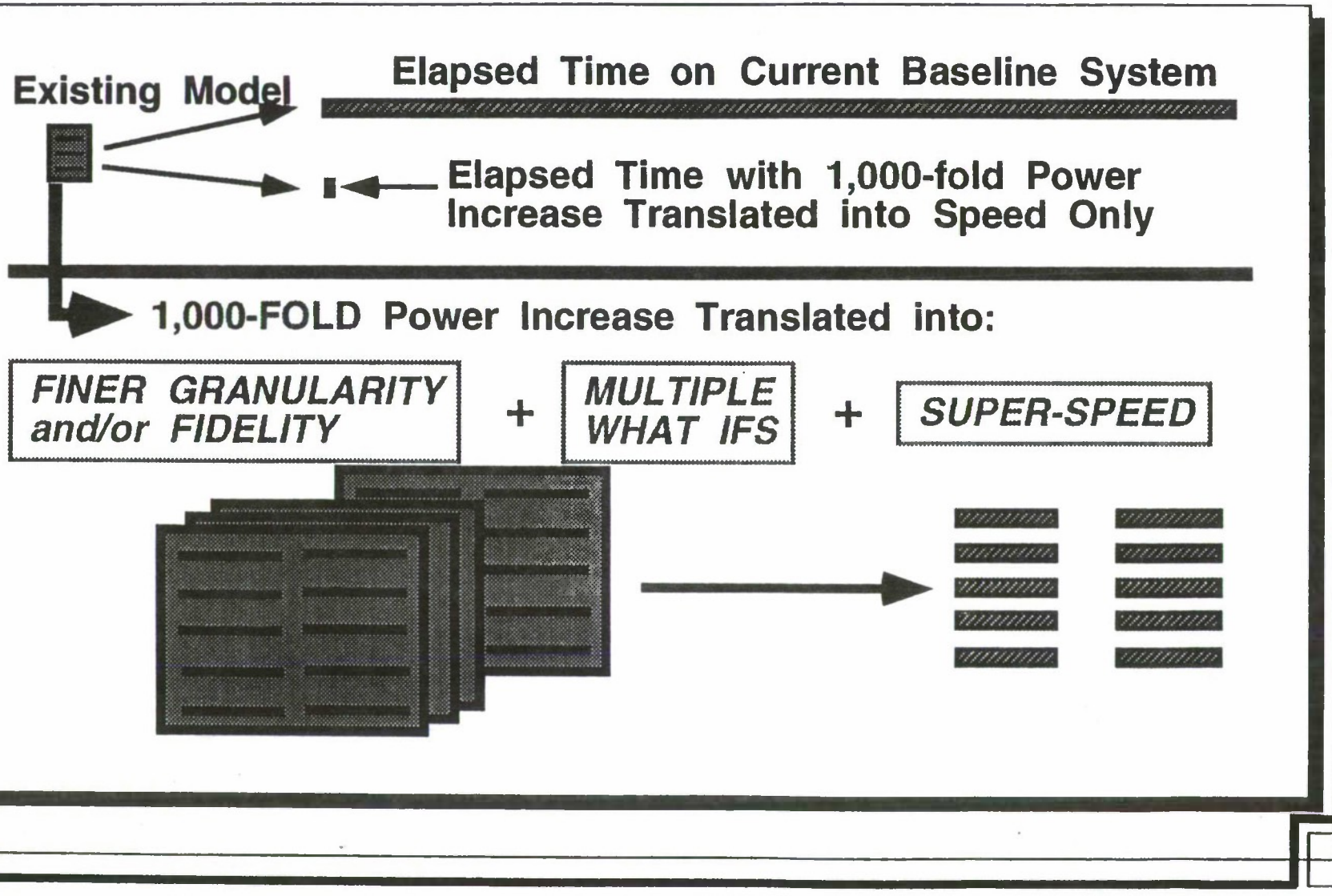
SuperC & DINS

AIR THREAT



- Calculates Red Air Threat to any given surface point using Air Order of Battle and doctrine
- Original model developed at CPF

SUPER-C POWER



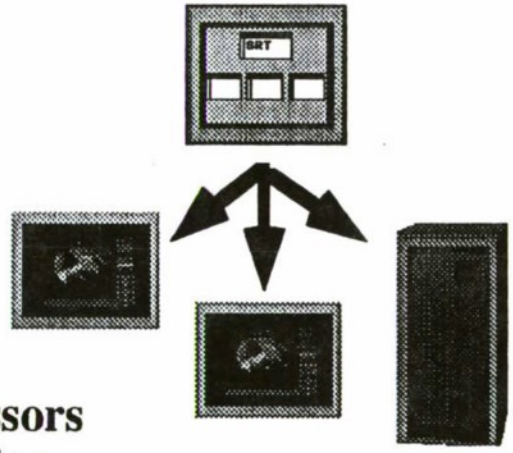
Superconcurrency Forms

Global



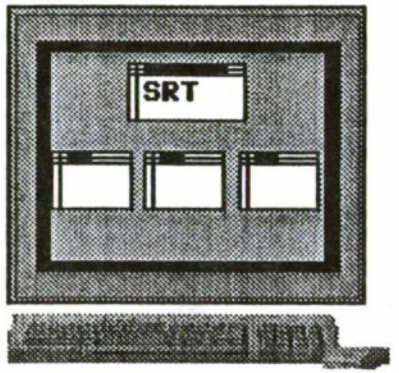
Span large geographical areas

Site



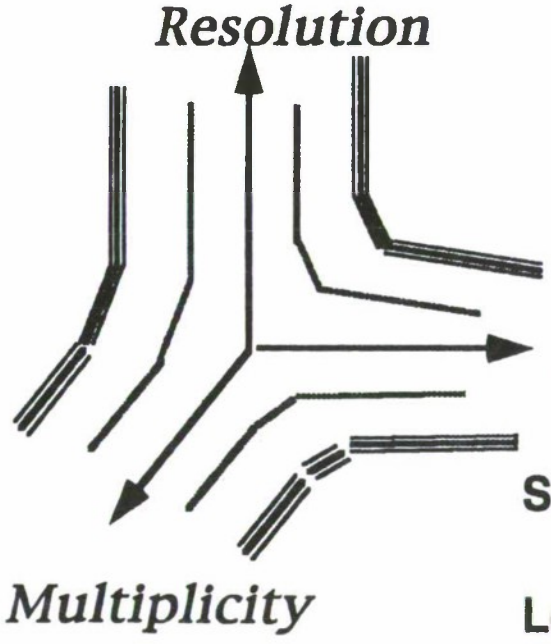
Diverse processors at one location

Micro



Diverse boards in one machine

VR & SuperC Power



Constant-Time Curves (of form y = 1/x)

- High-Power (SuperC)
- Low-Power (workstation)

- SuperC provides greater power and volume under the constraints
- Low-end systems require low-res variants for same multiplicity/functionality
- Common systems for fixed/mobile require this flexibility

Profiling and Benchmarking

Baseline Application



Execute on a vector supercomputer



2 Times faster than baseline

Execute on a heterogeneous suite of mini-supers



Distribution Overhead

10 Times faster than baseline

Model Development System

- Assemble set of tools for an initial demonstration of concept
  - object oriented simulation language (MODSIM)
  - object base management system (Zeitgeist)
  - network orchestration tools (PVM)
  - model base management tools (DEVS)
- Investigate and resolve compatibilities among tools
- Keep abreast of developments in parallel simulation and computing environments, and incorporate applicable new developments

THE SCIENTIFIC METHOD OF CHOOSING MODEL FIDELITY

By

Michael P. Bailey

Department of Operations Research

Naval Postgraduate School
Monterey, CA 93940-5000
(408)646-2085

5028p@NAVPGS.BITNET

March 12, 1992

This study was sponsored by the United States Marine Corps, MCCDC Warfighting Center,

Quantico VA

Abstract

Simulation modeling currently enjoys great popularity as a tool for solving problems within Department of Defense (DoD) activities. In this work we consider the process of upgrading an existing simulation model by increasing the fidelity of the model. The submodels to be upgraded and the degree to which they are upgraded should be chosen in a coherent, scientific manner. This is currently not the norm.

The pace at which the computer industry places power in the hands of analysts and their sponsors is accelerating. Sponsor organizations see this as an opportunity to enhance their model by increasing model resolution. Thus, fidelity enhancement decisions will be made more often, and in an environment of continual model growth.

Key Words:Simulation Modeling, Model Development

Contents

Abstract ii

1 Introduction 1

2 Fidelity 3

2.1 Definitions . . . 3

2.2 Assumptions . . . 4

2.3 Example . . . 4

3 Assessing Costs and Benefits of Increasing Fidelity 5

3.1 The Impact to Usability of the Model . . . 5

3.1.1 Data Hunger . . . 5

3.1.2 Performance Degradation . . . 6

3.1.3 Increased Developer/User Sophistocation Required . . . 6

3.1.4 Measuring Cost and Budget Constraints . . . 6

3.1.5 The Negative Impact of the Traffic Model in MCCAAM . . . 7

3.2 Benefits . . . 8

3.3 Independent Selection Options . . . 9

3.4 Interrelated Selection Options . . . 10

3.5 Evaluating the Benefit of a Package of Upgrades . . . 10

3.6 MCCAAM . . . 11

4 Issues of Practice: Implementation and Politics 11

5 The Future 12

1 Introduction

Emerging computer environment standards and technology, as well as software design improvements, have produced an unprecidented opportunity to improve existing models used in support of military analysis. The model enhancements selected for implementation come from a set of initiatives generated by the developer/user community. Although the designers, developers, and users of simulation models are sophisticated technologists and decision scientists, the model improvements they select are usually chosen in an entirely unscientific way.

In this work, I propose a fledgling methodology for determining which aspects of a given model should be chosen for improvement, and discuss the implementation of this method using examples from my own experiences in model development.

Emerging technologies in the computer science realm have produced an opportunity to change model fidelity expediently. Simulation modeling software using the object-oriented programming paradigm has existed for over a decade, but object-oriented models have a reputation for being poor performers and being hard to develop. A diversity of products have now become available which combine modern software production environments, object-oriented simulation (OOS), and performance competitive with FORTRAN models.

The ramification of the so-called OOS revolution to the present discussion is that models are forced to be modular, so that exchanging an existing submodel with an enhanced one does not carry the cost it used to. Thus, growth of simulation models from primative to sophisticated is more graceful, costs less, and does not rely on personnel possessing extreme intimacy with the model's implementation.

DoD policy has often motivated the evolution of computer environments and operating systems. The surge of *open architecture* system development within DoD is widely evident. The term open architecture implies that some degree of standardization has been achieved in

- operating systems,
- graphical user interfaces,

- data base management interfaces,
- network operations and protocols,
- interfaces to presentation graphics programs.

The impact of this movement is that, because of the degree of standardization, models are portable to an unprecidented degree. The future of model migration is extremely bright. Models will no longer adhere to the political boundries of the computer environment map. Model developers will be capable of producing machine-independent implementations so that models can keep pace with the growth of computer environment capabilities. Improving model performance and capabilities no longer comes at the high cost of reimplementation on new computers.

Model extensions will be pursued by organizations other than the model's sponsor. Hence, the sponsor will be presented with a variety of model upgrades whose software development costs are **sunk costs**. The sponsor will select a subset of the already-implemented upgrades for inclusion in the supported model. In this environment of continual upgrades with negligible software development costs, the model's sponsor organization should make upgrade choices in a structured manner.

The individuals who have ushered in this paradigm leap in computer environment standardization and object-oriented programming are the Henry Fords of the information era.

The changing nature of the perceived threat and combat environment will motivate organizations to expand models to accomodate scenarios different from those for which the models were designed. Reduced RDT&E budgets and budgets allowing less experimentation in weapons design will place more pressure on the modeling community to produce tools capable of reliably evaluating weapon systems' performance, tactics, decisions, and policies.

The simulation modeling community has a well-established record of exploiting emerging computer technology. Hence, fidelity of models will improve at a rate similar to that of available computer performance. *What submodels* to improve is a choice we will face more often, and one we should deal with scientifically.

2 Fidelity

In this section, I attempt to attach some formality to the notion of model fidelity. Some underlying assumptions concerning the nature of the level of fidelity problem will be presented so that the universe of the subsequent mathematical formulation is understood.

2.1 Definitions

Model **level of fidelity** refers to the degree to which the model produces the same outcomes as the tangible, physical system. Thus, a policy constructed with the aid of a model with infinite fidelity would be identical to a policy produced using unlimited experimentation with the real system.

Model validation is the (underexercised) practice of comparing the model to the physical system. The hope is that the model output is similar to the physical system to the degree that the model is usable. Model validation is often confounded by several factors:

- the physical system is not observable because it does not yet exist, observing it is a hazard to the system or the observer, the agency in control of the system prevents it from being observed, or observation itself causes the system to change its behavior;
- the environment for which the model is intended does not yet exist, or is inaccessible for some reason, or the physical system does not operate in an environment as pristine or consistent as that generated by the computer;
- the model is steady-state.

Thus, the edict *thou shalt validate* is often untenable in the military modeling discipline. The usual response to this situation is to increase **model resolution**. Here we define resolution to be degree to which submodels are disaggregated. Increasing resolution means replacing larger objects by semiautonomous subobjects, replacing simple decision logic with more complex logic, using more source data such as higher resolution terrain data, including more objects, or simply improving approximations at the cost of computational performance. Increasing resolution makes submodels more data-dependent.

2.2 Assumptions

The hidden assumption in the above approach is that the model fidelity *must* increase with model resolution. Simulation modeling may be thought of as the modeling of the interactions of objects and their environment. As the resolution becomes greater, the logic within an object becomes more environmentally dependent because more of the system is considered environmental to the (smaller) object. Thus, user confidence grows with resolution because more of the physical system is taken into account by the actions of the model's objects.

Is this reasonable? Analysts have often found themselves arguing in the negative, often to the frustration of both themselves and the sponsoring organization. The analysis community carries a reputation for favoring form over function, a reputation not wholly undeserved. The sponsor cannot fathom why the analyst insists on ignoring physical realities of the system modeled. The method proposed by the analyst is often much too mathematically sophisticated to be useful for the sponsor. Intercession in this often-occuring debate is critical to the future growth of the modeling community.

2.3 Example

In [1], an object-oriented simulation model of a Marine Air Ground Task Force (MAGTF) communications network, called the Marime Corps Communications Architecture Analysis Model (MCCAAM), is presented. The Marine Corps is preparing to procure a next-generation single-channel radio with ECCM capability. MCCAAM was designed to evaluate performance of allocations of next-generation radios to units in the MAGTF, where the measure of performance of an allocation is tactically driven. The ultimate goal is to select the best mix and allocation of radios–an optimization problem where the objective function is evaluated via simulation.

The message traffic generation scheme of this model is quite novel. With the help of the Marine Corps, we decomposed the communications network workload into basic operational tasks (BOSTs), each consisting of a group of message exchange occurrances (MEOs) arranged in a PERT network structure. Thus, the message traffic between a pair of units is highly dependent on traffic elsewhere in the network. We feel that this is a great advancement over the usual *iid* traffic generation schemes

seen in almost all other communications analysis models. We will focus on the inclusion of this communications traffic model as a proposed upgrade.

The question remains–was this upgrade worth the time and effort? How can we defend the premise that the new traffic model is superior to the usual one?

3 Assessing Costs and Benefits of Increasing Fidelity

Why aren't current models of extremely high resolution? The answer is fairly obvious, though sometimes difficult to elucidate. Model developers have cultivated a framework of constraints regarding the capabilities of simulation, an intuition concerning the level of importance of a submodel to the performance of the overall system, the availability of data, and the sophistication of the user. Finally, the budget afforded a given development effort dictates certain constraints on the fidelity of a model. This framework must be challenged, as it represents inertia which the DoD cannot afford.

3.1 The Impact to Usability of the Model

Any constraint in model fidelity reflects some cost. As stressed above, development costs will not be major contributors to the negative impacts of model upgrade. Thus, what are these negative impacts, and what do we include as a negative impact?

3.1.1 Data Hunger

It is clear that one of the major shortcomings of an increase in resolution is an attendant increase in data hunger. As a submodel becomes more specific and detailed, the requirement for supporting data increases. Collection of the appropriate data to support a submodel is vulnerable to the same shortcomings as model validation. Furthermore, some of the required data is often used to support some judgemental process, such as the probability that a SAM battery acts on a partially jammed tracking solution.

An axiom of model development is that more resolution requires more data. DoD does a great deal of data collection for the systems it owns, but this data is not usually collected with modeling

in mind. Open architecture standards will force all of these data sources to be compatable, so that access to data will not represent much cost. Thus, the **degree of confidence** in data sources is our primary concern.

3.1.2 Performance Degradation

Closely associated with the resolution of the supporting data used, poorer performance of the model is seen as a negative impact of model fidelity increase. While it is certainly true that computers are getting faster, increasing resolution of a submodel can cost several orders of magnitude in algorithm complexity. The analyst who develops a model's upgrades must evaluate the new platform, consider each possible upgrade, and budget his new capability among the alternatives.

3.1.3 Increased Developer/User Sophistocation Required

By including more detail in a submodel, the developer increases the required level of understanding for himself and the user. To a great extent, the user can approach submodels he does not understand as *black boxes*, and experiment only with the processes he understands. This approach carries some risk of model misuse, but defensive software design can reduce this risk beyond consideration.

The model developer's level of system knowledge is not optional. In order to produce the appropriate submodel, the developer must be expert in the model and in the system. Aggregated models are often produced because the modeler does not have the background, the appetite, or the time to become a system expert. This problem is particularly prevalent in models produced in an academic environment, where depth of knowledge of the physical system may not be a high priority.

3.1.4 Measuring Cost and Budget Constraints

Measuring uncertainty of the data sources for alternative submodels is fairly straightforward. There are well established methods for determining the sensitivity of a model to a single datum, see [4], [2], or [3] for academic treatments of this topic. In [5], a simpler though less developed method for estimating model sensitivity to submodels is presented. By combining sensitivity and uncertainty

measures in a manner appropriate for the problem, relative risk can be computed for each upgrade option. By analyzing these values, program managers can constrain this summed risk, so that any set of upgrades undertaken does not exceed the comfort level of the program manager.

Suppose that we are considering a package of submodel upgrades involving submodels $1, 2, \ldots, N$. If the risk associated with submodel i is given as r_i, then the combined risk is $r_N = \sum_{i=1}^{N} r_i$. The project director's comfort level, given by R, constrains the data-hunger-generated risk as $R \geq r_N$ We must remind ourselves that this risk is to the integrity and reliability of the model, not to the developing organization.

Accessing performance costs is straightforward. We want the model to complete in a specified amount of time. Upgrade negative impact in terms of time should be seen as a multiplier. Thus, if we have upgrades $1, 2, \ldots N$, and testing upgrade i seems to make the time until completion m_i longer, then the performance cost of including the N submodels is $m_N = \prod_{i=1}^{N} m_i$. If we expect our new platform to provide 1.5 times the speed of the current platform, and we cannot sacrifice any turn-around time for the model, then we must have $m_N \leq 1.5$.

Finally, in considering cost of sophistication, measurement seems very difficult. Certainly, budgetary and personnel constraints arise from the developer sophistication cost. User sophistication, to the extent that it is a problem, must be addressed. Certainly, making a simulation model *sailor-proof* requires extra design and implementation assets.

3.1.5 The Negative Impact of the Traffic Model in MCCAAM

In MCCAAM, all three of these factors were considered.

Data for the new traffic model was sought throughout the Marine Corps and then the U. S. Army. While data supporting an *iid* intermessage time model is readily accessible, no organization currently collects information on the intergeneration time of BOSTs. However, the development team was convinced that only relative frequency information was required to continue. Hence, the team set off to construct this relative frequency data using expert judgement of Marine Corps C^4I specialists. Extensive sensitivity analysis will be performed on these relative frequency measures.

Performance loss due to the new model could be significant. The impact of the dependence of message exhanges will cause a look-up of the appropriate reply and information distribution requirements. In the event that the BOST cannot be executed in a straightforward manner, complicated message routing routines are used to find alternate communication paths between units.

The requirement for developer sophistication for the upgrade is significant. I found my background in queueing networks to be of minimal assistance in designing MCCAAM. Rather, I strive to become indoctrinated into the world of Marine Corps C^{4}I, a process that will continue in perpetuity. Through active guidence of students, Marine Corps sponsors, and C^{4}I specialists, the design of MCCAAM proceeded expediently.

User sophistication was not a constraint in any way, as the potential users of MCCAAM are C^{4}I specialists. One of the design team members was dedicated to producing a user interface and input validation scheme which will ensure that the model is not misused.

3.2 Benefits

From the above definitions, we can say that we seek fidelity, yet we would like it to come at the expense of small increases in submodel resolution. How is the benefit of the increase in fidelity manifested? How do we measure it? We might call any such measurement a measure of overall model fidelity.

In this discussion, we will consider the case where the simulation is used in support of making some selection from a number of alternatives, (eg. C^{4}I architectures, ammunition round reliabilities, EA-6 jamming assignments, or submarine tactics). A different approach must be developed for simulations used for training.

Let S be the set of feasible alternatives for the selection, and measure the distance from s_1 to s_2, both members of S, as $d(s_1, s_2)$. The function d is seen to be a penalty for selecting s_1 when s_2 was indeed the optimal choice, with $d(s_1, s_1) = 0$ for all $s \in$ S. In the MCCAAM model, where the selections are the allocations of next-generation radios to MAGTF units, distances between selections s_1 and s_2 could be the number of units having the new radios in allocation s_1 they do not

have the new radio under s_2. Selecting s_1 when we should have selected s_2 is not a terrible mistake if $d(s_1, s_2)$ is small, but is catastrophic if $d(s_1, s_2)$ is huge.

Suppose that we *could* experiment with the real system as if it were a simulation model, and that we could sample an unlimited number of replications. Let p_j be the relative frequency of the event that s_j is the best selection. Depending on the underpinning of the reader's philosophical framework of probability and optimization, p_j could be seen as the probability that selection s_j is the optimal choice. Henceforth, I treat $p_j : s_j \in \mathbf{S}$ as probabilities. I propose two measurements of increased fidelity, one for the case $d \equiv 1$ for distinct selections, and the other where $d(s_j, s_k) \in [0, 1]$, varying within its domain.

We run our selection process for M independent runs for the current model and a model with submodel i upgraded, heretofore known as the baseline and upgraded models, resp..

3.3 Independent Selection Options

If $d \equiv 1$ for distinct selections then we perceive all of the elements to be equally (un)suitable to be mistaken for the right selection–wrong is wrong, and that's all that counts. In this case, we can use a straightforward frequency table to measure the differences in simulation output.

Let $f_b(s_j)$ be the frequency with which the baseline model chooses selection s_1 as the solution, while $f_u(s_j)$ is the frequency of s_1 for the upgraded model. $f_b(s_j)/M$ is the baseline's point estimate of p_j, so that the worth of the upgrade may be measured by a simple sum of squares

$$SS_i = \sum_{j \in \mathbf{S}} \frac{[f_b(s_j) - f_u(s_j)]^2}{M^2} \tag{1}$$

It is well known that sums of squares tend to be somewhat oversensitive to outlier data. Summing absolute values of the differences may be more appropriate in some applications. If the proposed upgrade of submodel i has a large impact on the output of the model, we will detect it as a high value of SS_i.

3.4 Interrelated Selection Options

If our difference function d does indeed carry some useful information, we should exploit this information when choosing our upgrades to pursue. In this setting, we make the following two assumptions:

1. the upgraded model is assumed to be making better selections;
2. each pair of model runs have been made with the same environmental sample path, so that any discrepency in selection is directly attributable to the difference in submodel i's fidelity.

The data we collect from each run is

$$I_k(l,j) = \begin{cases} 1 & \text{iff run } k \text{ selects } l \text{ for the baseline run, } j \text{ for the upgraded run} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Thus, we can estimate the expected degree of mismatch EM between the selections made by baseline and upgraded models via the formula

$$EM = \sum_{j \in \mathbf{S}} \hat{p}_j \hat{P}[mistake\ l\ for\ j] d(l,j) \tag{3}$$

$$= \sum_{k=1}^{M} \sum_{j \in \mathbf{S}} \{ \frac{\sum_{l \in \mathbf{S}} I(l,j)}{M} \{ \frac{\sum_{l \in \mathbf{S}} I(l,j) d(l,j)}{M} \} \} \tag{4}$$

where we use the upgraded model for more reliable estimates of p_i. When $d \equiv 1$, the value of EM reduces to the estimated probability of mismatch

$$EM = \sum_{j \in \mathbf{S}} \hat{p}_j \hat{P}[mismatch j] \tag{5}$$

$$= \hat{P}[mismatch] \tag{6}$$

3.5 Evaluating the Benefit of a Package of Upgrades

Using either of the above selection interrelationship models, a set of upgrade alternatives K might be evaluated using a sum of individual upgrade benefits:

$$B_K = \sum_{i \in K} SS_i$$

or

$$B_K = \sum_{i \in K} EM.$$

Obviously, B_K does not take the interaction of the upgrades into account. It is likely that the interaction of several simultaneous upgrades will cause cancelation of benefit effects, so that B_K is a first-order approximation, and probably an upperbound. Given the crude nature of the preceeding analysis, this approximation seems serviceable.

3.6 MCCAAM

The MCCAAM message traffic generation scheme is being analyzed at the time of this report. I believe that, under the assumption of *iid* message generation, the SINCGARS radios will be allocated to the nets carrying the most traffic. Under the BOST structure, preventing transmission of one or two critical messages will cause the entire BOST to be stopped, causing a great decrease in measurable network performance. Thus, I believe that allocation of SINCGARS in the upgraded model will place SINCGARS on nets carrying traffic *operationally critical* to a large number of tasks which the MAGTF undertakes.

Because there exists a natural way to measure the distance between selections, we will seek to exploit the distance function d described above to measure the value of the BOST-structured traffic model as an upgrade.

4 Issues of Practice: Implementation and Politics

Given the evidence of the utility of adopting the standards of open architecture and object-oriented simulation, DoD should support the reimplementation of models to position them for future growth. This step must be taken before any of the above analysis gains any degree of validity.

It would be extremely naive to propose that program managers construct the constraints and benefit functions as in the previous sections, and plow forward with the upgrade policy generated by the implied constrained optimization problem. To pursue the evaluation method described above will force development teams to compare the impacts of increasing resolution of submodels in a consistant manner.

Alternatively, suppose that a group of upgrades have been proposed by the organization responsible for the growth of a model. Oversight of these decision makers should involve examination of the analysis behind the selections, and possibly a formulation similar to that described here. It should be easy to verify that the selection of upgrades was made under reasonable assumptions, and that the constraints generated use reasonable models for performance loss, data source risk, and user or developer sophistication.

As has been illustrated in the example of MCCAAM, this framework for considering model fidelity is serviceable in the abstract, as well as when used on an existing model. Model developers should be obligated to produce similar reasoning whenever presenting plans for upgrades.

Finally, most developers realize that their job includes advocating model growth. I believe that this framework gives them a quantitative tool for arguing their case for model enhancement with program managers.

5 The Future

I believe that the future of military simulation modeling relies heavily on the adoption of OOS and open architecture standards. Once this occurs, model growth can accelerate at rates comparable hardware capability growth. In this emerging environment, I believe that there is a need for quantitative tools for evaluating the utility of proposed upgrades, for setting growth priorities, and for assessing the negative impact of growth. While model development analysis remains one of the softest sciences known, I believe that the analysis presented above can be used to structure the problems of model growth and facilitate critical thinking.

References

[1] Bailey, M., W. Kemple, M. Sovereign, M. West, C. Chase, and C. Reece. 1991. An Object-Oriented Model of Communications for the MAGTF, *Proceedings of the Summer Simulation Conference*, Baltimore.

THE SOFTWARE ENGINEERING PARADOX:

UNEXPLOITED COST SAVINGS & PRODUCTIVITY IMPROVEMENTS

Barry G. Silverman

George Washington University
Washington D.C. 20052

ABSTRACT

When confronted by new problems, most programmers and modelers react intuitively and think analogically about past programs that could be reused in the new one. A paradox lies in the fact that software educators, managers and organizations frequently believe that reuse does not occur. This dichotomy between the discipline and the practice has created a number of immediate opportunities for cost savings and productivity improvements that are explored in this paper.

INTRODUCTION

This paper asserts that (1) software programmers reuse previous programs despite the widespread belief that software is not portable*, and (2) the very belief that portability does not occur leads to both organizational obstacles to and suboptimal performance in the creation of new software programs. In short, belief in the software nonportability myth leads to increased software costs and less effective programming efforts.

Everyone agrees that software is already too expensive and that software costs continue to rise: in many advanced automated systems software comprises 90% of the purchase cost. The current debate over how to reduce these costs centers around the need for a universal compiler (ADA), a universal operating system (UNIX?), an automated programmers' support environment (APSE), automatic programming by a computerized software "factory", and numerous related initiatives: e.g., see [1], [2]. Each of these represent extremely valuable innovations with major "payoff" potential over the five to 20 year time horizon. It is entirely understandable as to why the bulk of innovative attention is focused upon their achievements. However, careful inspection of the software engineering life cycle will reveal that immediate software cost savings and programmer productivity improvements can be accrued right away simply through small alterations in the way projects and knowledge are managed.

The following section contrasts the two possible views of the software engineering life cycle. This is followed by empirical verification of the alternative (portability) view within both NASA and the military as well as a discussion of the immediate and near term managerial implications.

THE SOFTWARE ENGINEERING LIFE CYCLE

As portability is associated with the reuse of familiar, pre-existing elements or building blocks, it is useful to understand the logical underpinnings of such an approach. Specifically, the paradigm which provides an account of reuse of the familiar is the analogical one. An analogy is a common and mundane human technique for indicating one or more respects in which two entities are similar. Analogies are frequently used in an expressive or explanatory way (e.g., "the moon is a ghostly galleon", "that car is a lemon", etc.). However, most of us instinctively reason by analogy upon encountering a new problem situation. There we are reminded of a past situation (the Base) that bears strong similarity to a present problem (the Target) at varying levels of abstraction. This type of reminding behavior, called here practical analogy, serves to retrieve solutions that were appropriate in earlier problem-solving episodes, whereupon past solutions are then adapted to meet the demands of the current situation. That is, reasoning by analogy may be viewed as a comparison between two "items" or situations that can be broken down into three parts: (1) an analysis of the first item (the Base) into some abstract description (or deep structure) consisting of a

---

* Software "portability" is defined here to mean that a new piece of software can be traced back to a substantial number of older, pre-existing software modules, components, and subroutines that went into its construction.

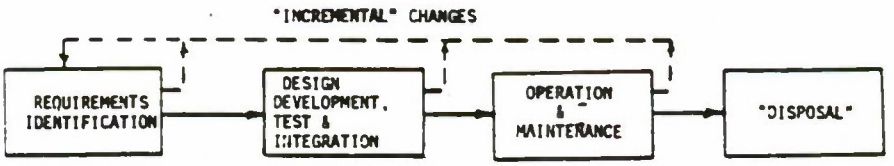


Figure 1 - The Classical View of the Software Engineering Life Cycle

problem-solution pair; (2) an analysis of the second item (the Target) into another abstract description, this time of a problem alone (the solution is as yet unknown); and (3) a relevant mapping between the two descriptions that permits the solution to the target problem to somehow be determined (i.e., analogies to be preserved, disanalogies to be discarded, and partial analogies to be adjusted).

In all types of analogies, a common and fundamental feature is that the relevant mapping is determined by the purpose for which analogs are being applied rather than by the number of similar features. This is most readily apparent in the expressive analogies which frequently compare two highly dissimilar things (e.g., moon and galleon, car and lemon, etc.). However, even the practical or everyday analogies are constucted in terms of their purpose or causal relevancy, not their literal similarity. Thus, figuring out how to make a reservation for a cruise might be compared to how we did it for a plane ride (even though the boat and plane differ). Or, a design for a new device is determined not by looking for similar devices, as analogs, but by looking for devices that fulfill similar functions as analogs.

This type of behavior is postulated here as playing a role in the problem-solving approach of subject matter experts when confronted by new and ill-structured problems in their field of expertise. Ill-structured problems are defined as those for which the problem itself is not well-understood (it is ill-structured or ill-defined) and hence, well-known,procedures or decision making tools cannot be readily applied to obtain a solution. Rather, an analogical problem-solving approach is pursued that permits the expert to better define both the problem and the solution at the same time.

Returning to software problem solving, the theory of analogy would suggest that programmers think analogically about past programs when confronted by the need to create new ones. However, the software engineering discipline that has emerged in the past 15 years makes no mention of such an approach: e.g., see [3-5].

The foundation of this disciplined approach is the life cycle model that in simplified terms begins with definition of requirements, progresses through top down design and development, and finally accounts for usage and maintenance as indicated in Figure 1. This is a powerful approach, that with all its implications and ramifications, has already and continues to lead managers to better quality products at lowered life cycle costs.

The principal weakness of this approach is that the life cycle model literature and discipline provides a structured approach only for those situations in which new products are to be created. It neither recognizes the usefulness of analogous products nor tells the manager or programmer how to incorporate them. For example, while "incremental development" is referenced, this is a top-down technique for new products only. Even the literature and theory on software reuse via conversion, or design for portability is concerned primarily with new software and how to get it to work on a computer other than the one it was designed for.

There is, however, reason to believe that the analogy approach might play an important and unidentified role in the software life cycle in particular, and in the system engineering life cycle in general. The theory of analogy suggests that the managers and programmers approach new software (or engineering) efforts with what they already know. That is, when confronted with the need for a new system, for better or for worse, people will adapt analogous products (building blocks) to the extent possible in each phase of the life-cycle. In particular, this could include the transfer +/or adaptation of products ranging from old concept and requirement statements to old designs to actual lines of code as well as to many intermediate and related plans, procedures, and products commonly found in the systems environment (see Figure 2).

If this is the case, the classical life cycle model would need to be revised to account for the "fact" that

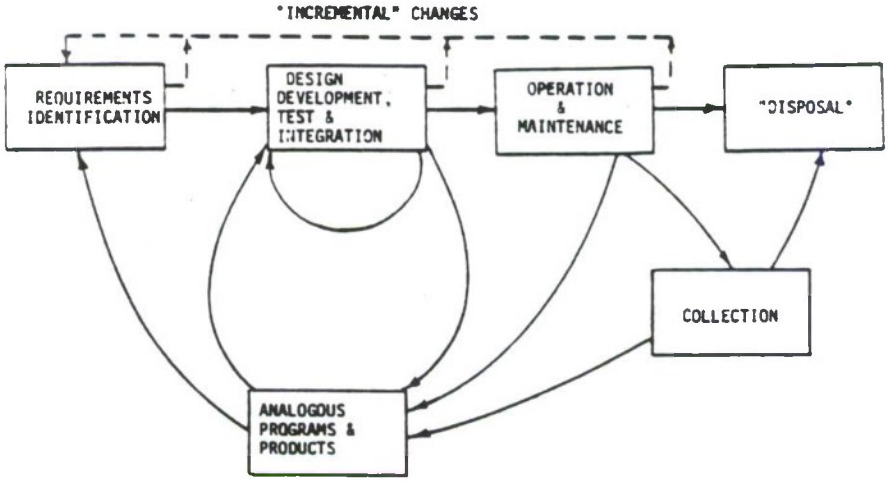


Figure 2 - The Analogical View of the Software Engineering Life Cycle

developers are cleverer at management and cost cutting than it suggests. In addition, this would suggest that research and development on potential tools and techniques for improving the process of utilizing analogous software (or system engineering) products should be increased. Study and understanding of how the life cycle works in practice holds the potential to lead to dramatic changes in both the way we view and teach (system) engineering and in the ultimate control of system cost.

EMPIRICAL FINDINGS

Over the past two years, the author has collected data from NASA and the Department of Defense in an effort to test the hypothesis that software programming is largely an analogical building block adaptation exercise [6-8]. Data collected include program statistics, software design documentation, and counts of lines of code reused. In addition, the author administered questionnaires to two workshops at NASA and personally interviewed several dozen software programmers and managers at NASA and in the military. The results are summarized in Table 1 and elaborated upon below.

NASA -- Size and reuse statistics were provided by NASA's Goddard Space Flight Center on sixteen software programs developed for satellite operations control centers. Of these software programs, twelve are based from 68 to 95% on old, pre-existing software, two are 50% revised, and two use an entirely new code. The author further investigated one of the "worst case" or zero re-use programs: this was the newest program and it was twice as large as any of the others. Examination of design documentation combined with interviews of designers led to the finding that in the original zero-reuse estimate the basic structure and most of the modules were predicated on an earlier similar computer program. In addition, the latest plans contradicted the earlier estimates (of Table 1) and showed that adapted, converted, or simply reused old lines of code actually accounted for 63% of the program and new code for only 37%. Due to the age of the other so-called "zero re-use" program, the author was unable to conduct a similar investigation.

Military Conflict Models -- Thirty-six military conflict computer models were selected from a wargame catalog in a search for a C^3I simulation capability. A half-dozen of these were selected at random for closer inspection, whereupon (1) the designers were contacted, (2) documentation was obtained, and (3) the history and geneology of the model was traced. Most of the models had evolved from a diverse set of building blocks that in turn could be traced back, in many cases, through an evolutionary chain beginning back in the early 1960s.

Joint Test Models -- Three current Joint Test Force sites (C3CM, IFFN, and JFAAD) were visited in an effort to assess the status of their approach, procedures, and standards concerning the use of computerized military conflict models. These are the first three

Table 1 - Building Block Reuse Statistics Of Existing Software Programs

		NASA	Dept. of Defense	
		Satellite Control Centers	Conflict Models	Joint Test Models
No. of programs examined		16	36	3
Program size (1000s of executable lines of code)	Average	83	20(a)	200
	Range	57 - 239	Unknown	50 - 500(a)
Reliance on building blocks		Yes	Yes	Yes
% code reused	Average	68%	Unknown	High
	Range	0 - 95%	Low - High	Low - High

(a) Best estimate

Joint Test Forces to utilize models to support pre-, during, and post-test activities, and thus, would appear to be prime candidates to develop new code. All three sites, however, are relying heavily upon the adaptation of building blocks -- i.e., prior conflict models available from the military community at large -- to cut costs and delay. One site may be considered a major software development effort. However, that Joint Test Force surveyed 200 potential building blocks and identified 32 models that could be incorporated to varying degrees into their architecture.

The point of all this data is simply that anyone who bothers to look rapidly discovers that the analogical building block version of the life cycle is representative of the practice a substantial amount of the time. Reuse is a fact of life. The more important findings, however, pertain to the adverse impacts created by the fact that management and organizations believe the myth that reuse and portability do not occur and that the traditional life cycle (Figure 1) describes the practice.

For example, the NASA questionnaire response and follow-up interviews revealed two unexpected paradoxes [9]:

PARADOX 1 -- Practitioners state they rely heavily upon the analogy technique and that the necessary organizational support for this approach is largely nonexistent. In real terms this means that practitioners are largely unable to locate information relevant to the reuse of building blocks because the corporate memory never accumulated it. Instead, potentially inferior, yet more readily available building blocks and analogs are used in an attempt to do the best with what one has.

PARADOX 2 -- Numerous research results have shown that humans, even experts, are highly susceptible to error in the treatment of uncertainty yet don't even know it [10]. The questionnaire results reveal a related situation: that human confidence and bias in the face of uncertainty is likely to lead analogy practitioners to use suboptimal analogical knowledge combined with inferior decision rules without loss of confidence in the results.

The point of these two apparent paradoxes is that the job is getting done. Yet it is not being done as cheaply, as efficiently, and as effectively as might be the case with a small one-time investment in job and information flow redesign. More on this shortly.

As another example, interviews at the three Joint Test sites revealed four principal drivers of modeling and simulation (MS) cost and effectiveness that are quite similar to the problems captured by the NASA paradoxes. That is, (1) a lack of central organizational support from project to project in terms of a pool of experts who can aid the reuse process; (2) a lack of

experience in the "best bet" building blocks of individuals assigned to the software project; (3) a lack of common procedures and standards that a given project can use to guide its software engineering effort (as practiced, not as taught); and (4) a lack of readily available information on the potential building blocks that can preclude each joint test having to repeat the learning curve obstacles for the same model building blocks.

WAYS TO IMPROVE EFFECTIVENESS

Space permits only a brief summary of three of the many inexpensive measures that managers can take to reap sizeable cost savings and productivity improvements (see Table 2). In brief, there is a need for a formal recognition and support of the building block approach in terms of procedures, services, and staff that provide the necessary "corporate building block memory" that each new project can tap into. The normal corporate memory does not automatically solve this problem as the findings cited above indicate. One service that appears particularly useful lies in the development of computerized decision support systems that can act as intelligent assistants who can translate user software or modeling requirements into building block architecture blueprints. This type of aid should not be confused with "automatic programming" or the "software factory" proposals for the 1990s and beyond. This type of aid is achievable today, is already being implemented, and is relatively inexpensive.

One question to close with is how could so many cost reduction/productivity improvement opportunities possibly have escaped the attention of so many managers? There are many possible answers, one of which pertains to the fact that the analogical paradigm is frowned upon by educators in western thought and, consequently, by 'rational' managers. It is now believed that the human brain generally processes logic, verbal, and analytical thoughts in the left hemisphere and intuition, visual, and analogical thought patterns in the right hemisphere. Philosophy, psychology, and education in the western world is largely oriented toward the left side of the brain. Engineers are taught deductive, analytical thought; managers are taught rational, well-defined approaches; and programmers are taught structured-logic methods. In fact, the computer is itself a mere extension of the left hemisphere of the brain. However, skill and expert performance are largely associated with intuition, analogy, and right-brained thought: e.g., see [10]. These methods are not taught, are not formally condoned, and hence, are overlooked. Yet, it is only by appreciating the right-hemisphere techniques that the opportunities described in this article can be realized.

TABLE 2 - SELECTED OPPORTUNITIES AND PROSPECTIVE BENEFITS

Area	Opportunities	Benefits
Approach and Procedures	Standardize and Distribute Procedures, Standards, and Manuals for the Reuse of Building Blocks	1) Save startup $s and effort 2) Reduce the Pitfall Potential 3) Enhance Commonalities/Reuse
Learning Curves	Provide Building Block Services	1) Save startup $s and effort 2) Reduce Learning Curve Obstacles 3) Enhance Commonalities/Reuse
Staff	1) Set Staff Training Standards	1) Eliminate "Novice" Mistakes
	2) Create a Small Building Block Support Center	1) Economies-of-scale 2) Global "Optimization" Across Projects 3) Reduce Learning Curve Obstacles 4) Enhance Commonalities/Reuse

BIBLIOGRAPHY

1. "Special Issue -- The DoD STARS Program (Software Technology for Adaptable, Reliable Systems)", Computer, 16, no. 11 (November 1983).

2. Yourdon, E. (ed.), Writings of the Revolution: Selected Readings on Software Engineering, Yourdon Press: New York, 1982.

3. Boehm, B. W., Software Engineering Economics, Englewood Cliffs, N.J.: Prentice-Hall, 1981.

4. Putnam, L. H., "SLIM System Description", McLean, Virginia: Quantitative Software Management, Inc., 1980.

5. Freeman, P., Wasserman, A. I., Tutorial: Software Design Techniques, New York: IEEE Computer Society (EHO 161-0), 1980.

6. Silverman, B. G., "The Use of Analogs In Systems Engineering: A Software Programming Case Study", The George Washington University, Engineering Administration Department (1983).*

7. _____, "A Study of Methods to Support Development of Mathematical Models and Standards for Joint Service Activities (Briefing)", prepared for DDTE/OSD by the GWU/EAD (July 1983).*

8. _____, "Modeling and Simulation Standards, Procedures, and Support Systems (Briefing)", prepared for DDTE/OSD by the GWU/EAD (December 1983).*

9. _____, "Why They Use Analogs", ch. 4 in Management of Innovation: A Look at the Role of Information and Artificial Intelligence Aids (forthcoming).*

10. _____, "Expert Intuition and Ill-Structured Problem Solving", in D. Lee (ed.), Management of Technological Innovation, Washington, D.C.: NSF, 1983 (to be reprinted in the IEEE Transactions on Engineering Management).*

ACKNOWLEDGEMENT

Research cited in this paper was in part supported by DDTE/OSD (ONR0014-83-C-0563) and NASA/GSFC (NGT-09-010-800 and NAS5-27200). Only space prohibits the naming of the many individuals who have contributed to this research.

* Photographic copy available from the author.

BIOGRAPHY

Barry Silverman received the B.S., M.S., and Ph.D from the University of Pennsylvania and has been at the George Washington University's Department of Engineering Administration for the past six years where he is now Associate Professor. During that time his research sponsors included numerous public and private sector organizations for whom he designed, developed, tested, or validated a variety of software and modeling products. Dr. Silverman's current research interests lie in the development of a human information processing model of the expert innovator that can be translated into aids for software and system innovation managers. Dr. Silverman is the author of numerous articles, book chapters, and a forthcoming book on these topics, and he is a member of ITEA, IEEE, and TIMS. ■

AUTHORS

Do you have or desire to prepare a paper for publication in the Journal of Test and Evaluation (JOTE)? JOTE offers you the opportunity to contribute to the record of progress and development in the test and evaluation (T&E) field. In addition, JOTE is read by over 3000 professionals who desire technology transfer and often seek coordination and dialogue with authors published in the Journal.

JOTE solicits papers on such subjects as techniques, process, fundamental concepts, unique facilities, simulations, specific programs, management of T&E, criteria, the 'ilities, and evaluation. JOTE also welcomes specialized papers on the specific topics of "exploratory, developmental, and operational T&E," or "research versus acquisition versus application T&E" involving government, industry, or consumer products.

If you indicate a desire to publish in JOTE, request an Author's Kit for final preparation of your paper along with the appropriate release form. The Author's Kit contains the materials necessary for layout of camera-ready copy and an Author's Guide to aid you in formatting your paper. Typical length of a paper should be six to ten single-spaced 8½" x 11" pages including illustrations yielding four to eight Author's Kit pages for 77 percent reduction.

Papers for publication or inquiries regarding publication should be addressed to the ITEA Operating Headquarters. - EDITOR

Journal of
test and evaluation
459



Volume V Number 1 January 1984

- **Editorial** ... 4
- **President's Corner** ... 6
- **The Software Engineering Paradox: Unexploited Cost Savings & Productivity Improvements—** Barry G. Silverman ... 11
- **A Decision Support System for Threat Simulator Acquisition Managers —** Jacqueline C. Hamilton ... 17
- **Policy Recommendations for Software Test and Evaluation: System Level Test Issues —** R. A. DeMillo, R. A. Gagliano, R. J. Martin and J. F. Passafiume ... 21
- **TECHNET: A Network for the Test and Evaluation Community —** Edward W. Page ... 29
- **Ten Years of Air Force OT&E —** Terence R. St. Louis ... 37
- **The Transition From Peacetime Simulation to Wartime Reality —** Hamlin A. Caldwell, Jr. ... 42
- **Improvements to and Validation of the TAC Disrupter Simulation for Joint Test Support —** D.L. Ismari and C. V. Rolli ... 47
- **Association News** ... 54
- **Chapter News** ... 57
- **New Members** ... 62
- **Featured Facility** ... 66

Official Publication of the International Test and Evaluation Association (ITEA)

A LISP MACHINE BASED SCENARIO GENERATOR FOR THE INNOVATOR EXPERT SYSTEM

A. Murray
K. Faggoa
M. Wainatein
B. G. Silverman
INSTITUTE FOR ARTIFICIAL INTELLIGENCE
THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, D.C. 20052

ABSTRACT

Thia paper describaa the capabilitiaa of an axpart ayatam being daveloped to aaaiat U.S. Military (Joint Taat) paraonnai in tha managamant of modaling and aimuiation aoftwara aaaats. Thia axpart ayatam, cailad INNOVATOR, ia daaignad to aupport both tha operationai manager engaged in joint taat davaiopmant and tha mid-ievai manager raaponaibia for performing program overaight. Tha compietad ayatam wiii ba an automatad aid that wiii ba amployad in conjunction with aatabiiahad written procadurea for aoftwara aaaat management, raauiting in increaaad efficiancy and raducad coat. Tha ayatam incorporataa a taxonomy of acana dapictiona accaaaad by a Scanario Generator that ia capabia of dafining modaling and aimuiation requiramanta. A hiararchicai modai attributa iattica and aaaociatad ruia baaa for matching aoftwara aaaats to acanario raquiramanta ara aiao daacribad. Finaiiy, tha incorporation of viaual anginaaring principiaa for providing an intaiiigant machina interfaca operating in conaonance with anaiogicai raaaoning modaa and proceaaaa wiii ba diacuaaad.

1. INTRODUCTION

Tha purpoaa of thia paper ia to praaant the raauits of work performad on a "acanario generator" that waa daveiopad aa an extanaion of the INNOVATOR expart ayatam project. INNOVATOR ia an expart ayatam tool being appiiad to the area of modaiing/aimuiation aaaat managaamt. At tha tima tha original project waa conceived, the U.S. Military Joint Taat (JT) comaunity waa faced with rapidiy increaaing coata in aii areaa of taat daaign, taat planning, taat axacution and poat-taat anaiyaia. A high-iavai military Joint Sciantific Advisory Board waa aatabiiahad to gain further inaight into thaaa problama and to bring racommandationa and proposad aoiutiona to tha proper ievai of attantion. Tha board identifiad a number of problam araaa inciuding:

1. Inadaquata definition of modaiing requiramanta for JT's.
2. Littia identification of common thraada acroaa differant JT's.
3. Raducad credibility of aaparata JT'a raauiting from failura to uaa a common acanario.
4. Minimai intaroparability among aaparata JT modaia.
5. Failura to racognizo modaling aa an intagrai component of the Taat and Evaiuation (T&E) procaaa.
6. Lack of documantation atandardization, inadaquata modai vaiidation procaduraa and abaanca of common Maaaura of Effactivanaaa (MOEa) [11].

In raaponaa to thaaa concarna, a Modaiing and Simuiation Workahop waa convanad in tha apring of 1984 and waa attandad by raprasantativaa of OSD, thraa Joint Taat Directorataa, univaraitiaa and industry. During tha couraa of thia workahop, aavaral problam araaa affacting modaiing and aimuiation waro addraaaad, inciuding long davalopmant cyciaa, radundancy of afforta, maximizing uaa of axiating aoftwara, coordination among raiatad activitiaa, and uaa of laaaona iaarnad from pravioua afforta [2]. During the aama tima period, definitizad procadurea for tha managamant of modaiing/aimuiation-raiatad taaka wara daveiopad (Saa Figura 1) [3,4]. Thia waa a major atap toward undaratanding aoma of tha aignificant problama underlying tha modaiing/aimuiation procaaa. Ona problam in particular that waa addraaaad waa tha lack of attantion given to tha componanta of tha modaiing phaaa. Thaaa componanta inciuda: 1) buiiding a daacriptiva acanario; 2) praparing a atructurad acanario daacription; 3) identifying tha modaiing and aimuiation applicationa within the acanario; 4) formuiating tha modaiing approach; 5) dafining tha modai in tarma of functional and data requiramanta, acopa, purpoaa, vaiidation mathoda, atc. Aa ia ahown in Figura 2, tha modaiing

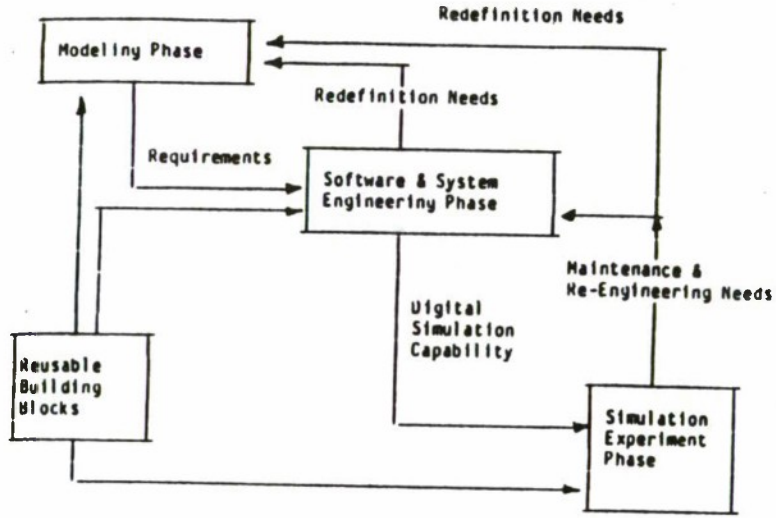
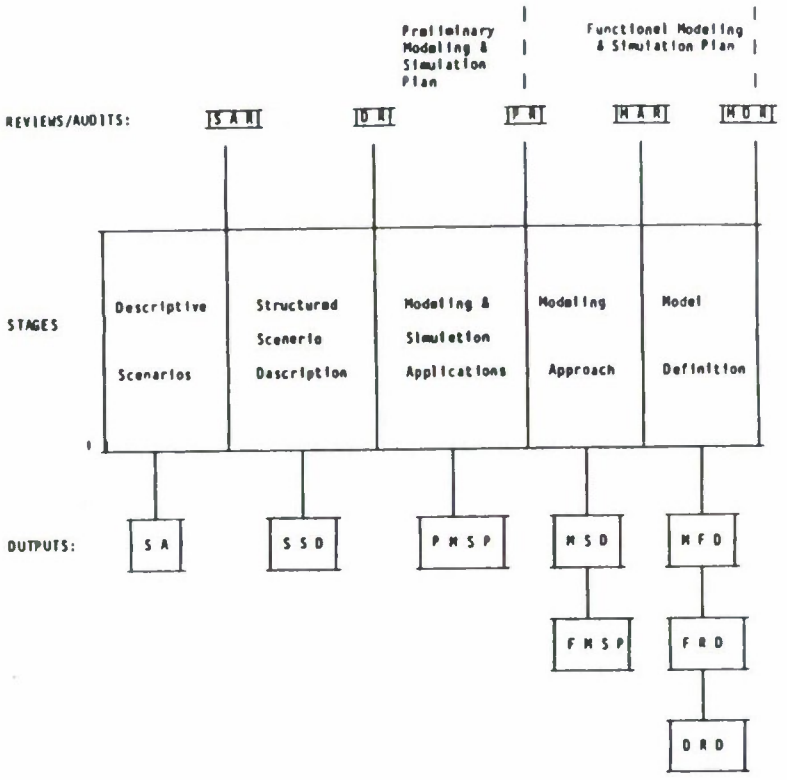


Figure 1. Modeling/Simulation Systems Development Process[4]



LEGEND

SA	- Scenario Analysis Review	DIDD	- Data Item Description Document
DR	- DID Review	PMSP	- Preliminary Modeling & Simulation Plan
PR	- PMSP Review	MSD	- Model Survey Document
MAR	- Modeling Approach Review	FMSP	- Functional Modeling & Simulation Plan
MDR	- Model Definition Review	DRD	- Data Requirements Document
SA	- Scenario Analysis	MFD	- Model Formulation Document
		FRD	- Functional Requirements Document

Figure 2. Outputs, Reviews and Decision Points for Each Stage of the Modeling Phase [4]

phase also involves several major decision milestones, participation of numerous organizations, and preparation of plans, reports, specifications and other documentation [3].

The INNOVATOR expert system kit provides an analogical expert systems approach to assist the test planner/manager in both the modeling phase and the software systems engineering phase. This expert system has an interactive consultation capability, a methodical taxonomy consistent with user cognitive style (progressive deepening), and heuristic search patterns to facilitate the manipulation of very large data bases. Most importantly, once the expertise is resident in the system, the performance of the expert system itself is not degraded as a result of personnel transfers. INNOVATOR was also structured to acquire additional expertise on its own, within the current limitations of machine learning and AI technology.

The modeling and simulation application of INNOVATOR, along with a knowledge base consisting of eighteen different models, was completed in the spring of 1985. This version allowed a planner/manager to specify, vaguely if necessary, broad modeling and simulation requirements. The expert system then presented the user with a compilation of existing models or model segments (modules), along with: 1) a rating of suitability to the proposed task, 2) citations of source material on the module and 3) the location and telephone number of a current expert for each of the selected modules.

INNOVATOR was found to be a robust and generic aid that provides assistance to a military-wide community of modeling/simulation users including the JT staff. The suitability rating was an especially significant feature in that it allowed the user to converge on an optimal solution depending upon the availability of definitized test requirements. Heuristic searches guided by user/computer consultations also contributed to this process. INNOVATOR was, however, a menu-driven system, and it was found that the selected queries/responses did not adequately support the manipulation of the very large and complex modelling/simulation problem domain. Furthermore, the human mind is highly visually-oriented and a graphics-oriented interface would accommodate random access user cognition much more readily, thereby improving efficiency and ease of use. This would potentially provide a substantial improvement over the textual interface, which was sequentially-oriented and therefore slower in nature. Based on these observations, it was determined that the use of computer visual engineering would be a beneficial enhancement to INNOVATOR, and this paper describes how this discipline was applied to the development of the scenario generator.

Figure 3 presents an overview of the INNOVATOR architecture as it currently exists, which is comprised of two main blocks: 1) the

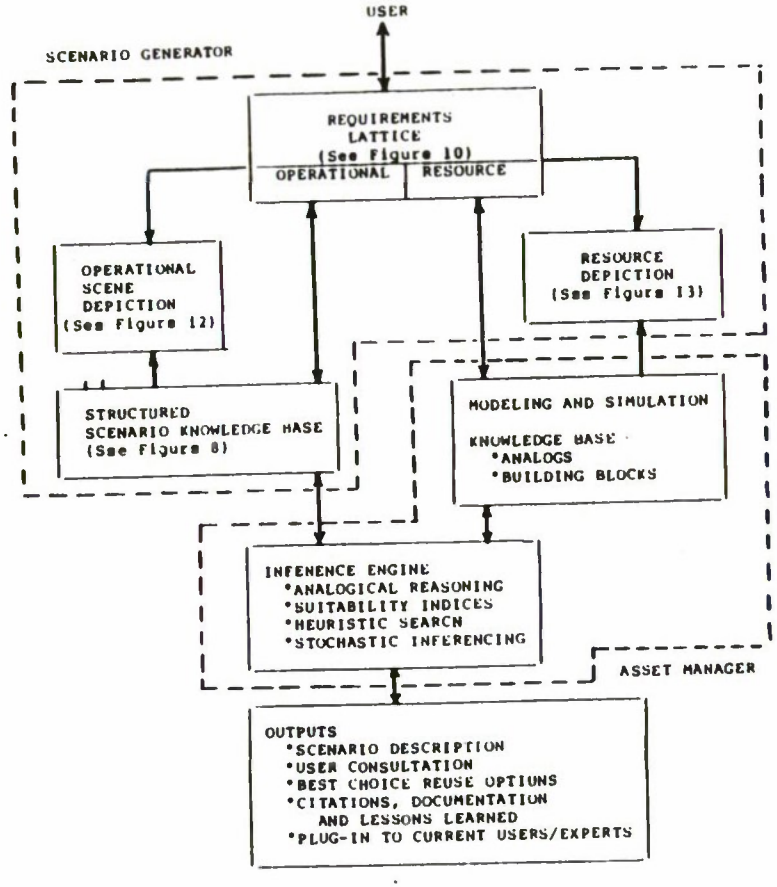


Figure 3. An Overview of the INNOVATOR Architecture

Scenario Generator and 2) the Asset Manager. This paper will discuss the scenario generator in depth; other articles [5, 4, 6, 7] address the asset manager portion. There are four systems resident within the scenario generator as shown. The requirements lattice is the primary interface driver, supported by two scene depiction windows that display icons representative of the key elements selected in an interactive user/system consultation. These modules interact with a global Structured Scenario Knowledge Base developed to support the operational aspects of the domain, and a Modeling and Simulation Knowledge Base designed to support the resource aspects. Each component within the Scenario Generator will be explained further in subsequent sections. However, a treatment of computer visual engineering in general is in order first.

2. COMPUTER VISUAL ENGINEERING

The advent of high-resolution bit-mapped graphics within the microcomputer environment has opened new possibilities into the graphical representation of knowledge, data and functions as applied to a specific problem domain [8, 9, 10, 11, 12, 13]. This same technology has proven to be effective in reducing fatigue and stress associated with working long hours in front of a Video Display Terminal (VDT) [14, 15]. The Lisp machine environment currently in place at the GMU/AI lab makes use of shape, size, density, depth, texture, direction, distance, inclination, etc., and provides invaluable assistance in representing knowledge of a multi-dimensional nature. These graphics aids are available to both the user and the programmer, and are tailored to respond to the unique problems of each.

One of the key ingredients in constructing a visually engineered interface is the icon. The icon, which as been a subject of study in psychology realms for many years [16, 17, 18, 19, 20, 21], has recently emerged into the computer field. To psychologists, the icon was determined to be the remnants, or "psychological phantom" of an image remaining in a subject's mind after the source image was removed [16, p22]. This "phantom" was found to have characteristics that were more simplistic, pertinent and integral to the overall understanding of the object under observation. Even eidetics (people with "photographic memory") were found to apply iconic principles subconsciously [19, p85; 21, p22].

Marcus [22] offers several important points to take into account in designing icons for graphics interfaces. These points include:

1. Simplicity, making use of hierarchies and representing only major components.
2. Clarity.
3. Familiarity (i.e., the user is quickly reminded of things/features already known).

4. Integrity.
5. Consistency (a feature recognized in one icon will represent the same characteristics in another),
6. Reliability.
7. Responsiveness, including speed, accuracy and pertinency.

Icons, by nature, can be troublesome if the above principles are not adhered to carefully. Hence, theories involving imagery, perception, and space and depth determination must be applied as subdisciplines to computer visual engineering. A simple icon such as that shown in Figure 4A [23, p451] may be obviously a rabbit to one viewer and a duck to another. Even more detailed depictions can be misleading as is the case with the classical wife/mother-in-law picture shown in Figure 4B [23, p451]. Above all, the iconic interface designer must be aware of the possibility of creating misleading assumptions on the part of the user [24, p31]. This can be overcome to some degree by presenting the user with explanations of the assumptions that the system will be conforming to. This will help ensure that the user will be operating in what Itteison calls the same "assumptive world" [24]. INNOVATOR complies with this caveat by employing a wide variety of prompt windows, warning messages and other techniques.



Figure 4A. The "Duck/Rabbit" Ambiguous Depiction



Figure 4B. The "Wife/Mother-in-Law" Ambiguous Depiction

INNOVATOR employs numerous object-oriented and access-oriented knowledge engineering packages that make extensive use of iconic representation. The INNOVATOR user environment provides an inheritance lattice structure interacting with iconic scene depictions, prompt windows, break windows, messages and hierarchical pop-up menus that allow the programmer/user to keep track of multiple interdependent elements simultaneously. These same aids are used in INNOVATOR to permit the user to easily present problems to the model asset manager expert system, as well as to modify the knowledge base or feed any problems encountered back to the developer. The underlying INTERLISP executive also provides such features as bit-map editing, display shaping, networking and pattern matching, as well as programming syntax aids such as CLISP (Conversational LISP) and DWIM ("Do What I Mean") [25].

Figure 5 illustrates an iconic type of visual programming aid known as a gauge. Gauges are useful in determining storage availability,

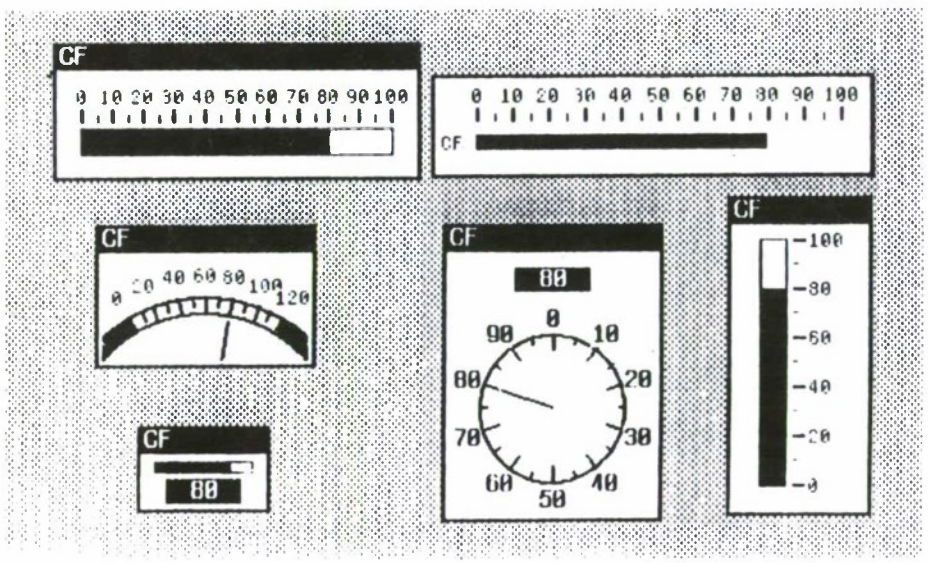


Figure 5. Sample Gauges

time remaining in a simulation, frequency of access, etc. One planned use of gauges in INNOVATOR is to continuously display the rating of suitability, which would permit the user to monitor the change in value as the solution space is traversed. Another planned enhancement is to display the level within the hierarchy that the user/expert system is operating. This would aid the user in understanding how coarse or how fine a level of detail is being examined at any given moment.

The creation and use of such images is based on the integration of a number of underlying software tools, starting with the display itself. The graphics display capabilities are based on "raster" display technology. A raster display screen consists of a number of individually addressable "pixels" (picture elements). Pixels are in one of two states at any given time: ON or OFF. The system represents the state of each pixel internally as a numeric value. A pixel that is ON is displayed on the screen in black color while a pixel that is OFF is displayed as a blank (white). The system has the ability to store the states of a collection of these pixels in a data structure known as bitmap. A bitmap editor is used to display a proportionate enlargement of a bitmap enabling the user to alter the display on a pixel-by-pixel basis, if necessary. Once a bitmap has been built, the user can use the mouse to perform a variety of operations on the bitmap, including inversion, rotation,

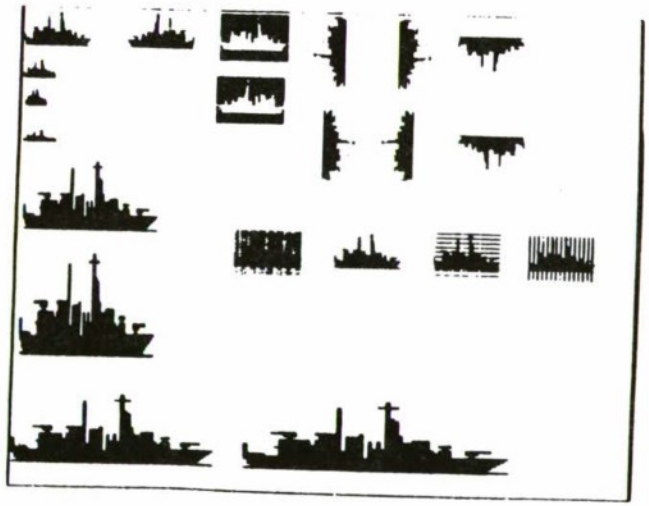


Figure 6. Bit-Mapped Representation Schemes

shrinking, etc. (see Figure 6). The final bitmap is usually stored for subsequent display as an icon.

Another system display tool is the "window", which represents an arbitrarily sized rectangular area on the screen. In fact, any output (textual or graphic) to the screen must be part of a window. Windows have gained popularity recently, especially in the personal computer field, since this technique allows segmenting the display into a more workstation-oriented configuration. Furthermore, windows can be "stacked" on top of one another in much the same way as papers, folders, etc, are arranged on an actual desk top. As will be shown later in the case of the scenario generator, several different windows can be used to build a composite display.

This section has served to acquaint the reader with a selection of theoretical topics and modes of implementation of the Computer Visual Engineering field. No attempt to be exhaustive was intented. The following two sections will examine the specific domain knowledge that INNOVATOR represents, and section 5 will explain how some of the principles of Computer Visual Engineering were applied to the problem domain as a means of constructing the scenario generator.

3. DETERMINING OPERATIONAL REQUIREMENTS

The goal of INNOVATOR is to assist the planner/manager in identifying existing modelling and simulation assets available to support a proposed program, thereby reducing the cost burden by making maximum use of existing software. Referring to Figure 1, INNOVATOR is being developed specifically to support the Modelling Phase and the first four stages of the Modal Development process, i.e., 1) System Concept Development, 2) System and Software Requirements Definition, 3) Software and Hardware Requirements Analysis, and 4) Preliminary System Design. It is intended that INNOVATOR will accept as input broadly specified requirements for a proposed development effort, will search the knowledge base for applicable software at the modular level, and will present to the user specific information including the organization and point-of-contact of where the software is being maintained. The basic approach, therefore, will be to match the user's requirements that are expected to be initially vague and somewhat uncertain against a more rigidly defined knowledge base of model performance attributes and descriptive characteristics.

It is essential that the requirements being entered by the user and the knowledge base being accessed by the system employ a common dialog. Furthermore, the components that make up this dialog must have operational credibility (i.e, realism and consistency), as well as traceability (i.e., an audit trail to the source policy, doctrinal, organizational and systems documentation). JCS Pubs, Army/Marine Corps Force Manuals, DoD Mission Area Analyses and unclassified DIA threat publications form the core of the INNOVATOR reference library.

Tha basis for defining tha operational raquiramenta will be the structured scenaro, as illustrated in Figure 7. Tha scenario providas a complete descriptlon of the initial conditions surrounding a given situation. Thosa conditions ara repraaantad in blocks comprising the stetus and makeup of both red and blue forces, i.e., a forca structure block, a material block, and a block containing tha missions, doctrines, tactics and objectives by which the represantstive forcas ara amployad. Within tha force structura block ara representations of command hierarchias with tha capability for granularity down to the individual unit level. Tha material block is essantially a hardwarm catalog that combines weapon systems and platforms with force organizational elements. The stete of

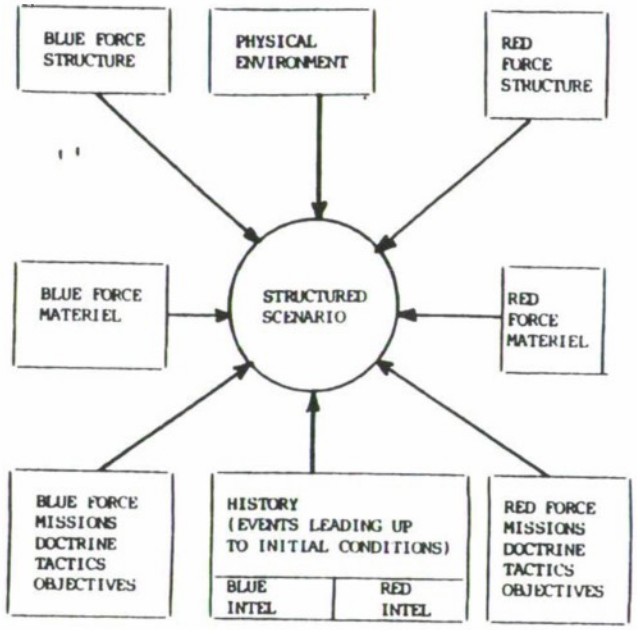


Figure 7. INNOVATOR Structured Scenario

reediness is also an important factor, and for this staga of development INNOVATOR will assume that a full state of raadinass exists at the time tha conflict is initiated. Subsequent varsions will allow the user to specify the dagrees of reediness, endurance and degradation among the various force alements.

The physical environment block contains information ragarding the time of dey, climate, weather, geological features and man-mada obstacles that are important aspects of the structured scenario. A block for history is also included to permit an accounting of significant events leading up to the situation being depicted. This block includas intelligenca, which portrays what aach side perceives tha status and conditions of thm opposing side to be. As will be shown latar, each block within the atructurad scenario repreeents a block within the INNOVATOR modeling/simulation knowledge base.

For the purposas of this paper, a specific scenario based on thm US Marine Corps contingency entitled "Operation MOSSBACK" was salmcted [26]. This was to provide a cradibia dascription of a two-aided confict at tha division lavel, which miso coincides with tha lavel of granularity of a majority of tha modmis currently residing within the INNOVATOR knowledgm base.

4. DETERMINING RESOURCE REQUIREMENTS

Tha previous saction formed a basis for knowledge represantation of tha operational (i.e., military) aspects of tha requiramants, however, a acheme is needed to rapresant tha modaling/simulation resources that will be matchad against tha operational requirament. As tha structured scenario forms tha basis for datermining operational requiremants, the modaling/aimulation usar's working anvironment forms tha basis for datarmining thm resourca requiremants. Thm user's working anvironmant is defined as all hardwara, aoftwara and supporting facilities nacessary for tha model to perform its function. Hardware is defined aa tha host computer and associated parameters such as accessibility, spead, transportability and maintainability. Storaga media, capacity and physical volumm of the equipment are also included, along with networking and operational hardware such as workstations and remota tarminals, displays, printers and communications devicas. Software is dafinad as tha required operating system and support software needed to run thm simulation. Outputs/results from other models ara incluaad in this category, along with othar data baaes that need to be accassed. Langueges, communications protocols and axecutive software features/constraints also need to be documented. Facilitias requiremants include such items as: environmental conditioning; TEMPEST or other security requirements including data encryption; physical parameters such as weight, volume, transportability and mobility; other concerns appropriate to the specific modeling/aimulation requirement.

To date, documentation on over one hundred different models has been reviewed. In some cases, the only available documentation consisted of a summary one page or less in length. Other models were only slightly better documentad, while for some models, a complete source listing was evaliahle. It was determined at the completion of the investigation that an intermediate-level description, that is, a level between the one-page catalog description and the full documentation package, was necessary. For this purpose, a worksheet was developed that contained what was perceived to be the most relevant information needed for classifying model performance parameters. The worksheet is described in [5], along with completed survey forms for the eighteen models that were used in the initial INNOVATOR knowledge base. This worksheet, in turn, was used in developing the resource requirements lattice, which will be shown in the next section.

An inspection of the architectures of the models used in the study showed that most of the models were structured in a modularized fashion, whereby algorithms and routines were grouped into modules, and the modules formed the building blocks from which entire models were constructed. Furthermore, many of these modules were found to have evolved over periods of up to one or two decades. It is at this module level that the expert systems approach shows the greatest promise, since many users were found to be primarily concerned with reusing the modular and sub-modular building blocks of existing models [2, 5, 4].

5. SCENARIO GENERATOR

The scenario generator for INNOVATOR was constructed by applying Computer Visual Engineering techniques that permit flexible and efficient user entry/retrieval of the operational and performance requirements identified in the previous section. The challenge was to present a vast and complex body of information in a manner that would be tenable to both the user and the programmer. The result was a "low-level" AI environment consisting of an object-oriented lattice window, scene depiction and resource depiction shown in Figure 8. Each of these windows will be described in detail later.

5.1 Requirements Lattice

The construction of the requirements lattice shown in Figure 9 was driven by two major requirements as outlined earlier: 1) that the requirements definition and model description employ the same dialog, and 2) that the requirements lattice be in consonance with the building blocks in the structured scenario. In response to the first criterion, the same INNOVATOR requirements lattice is presented to the user for both entry of model attributes into the knowledge base and for constructing the modelling/simulation requirements of the proposed application. The first two branches represent the domains of operational requirements and resources requirements, as presented earlier.

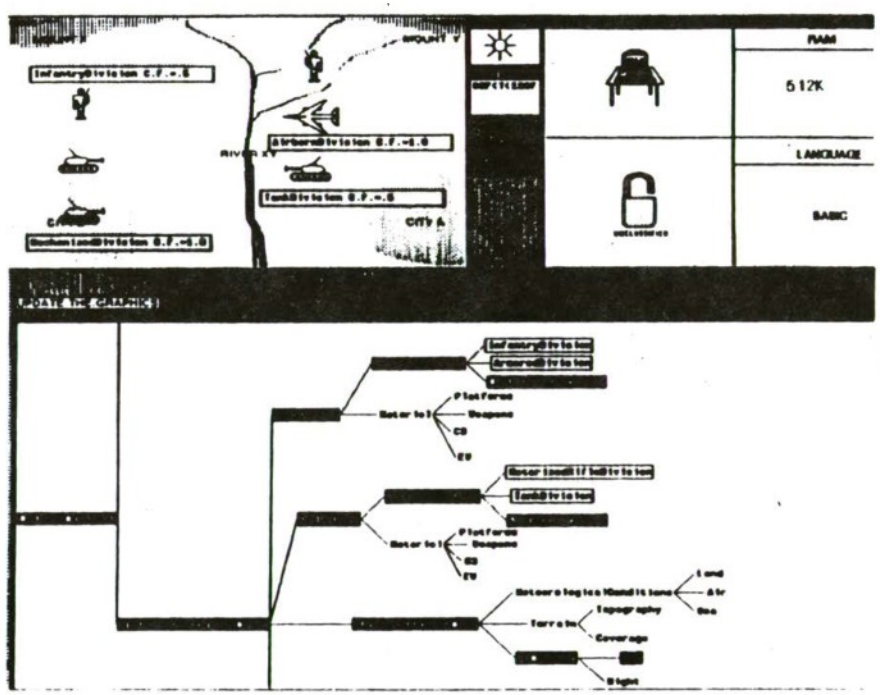


Figure 8. INNOVATOR Scenario Generator

give INNOVATOR a tri-state logic approach for dealing with uncertainty, rather than the more commonly used probabilistic approach. This resulted, in part, from work with earlier versions of INNOVATOR in which probabilistic assignments integrated over very large search spaces tended to lose significance. With a tri-state approach, the user is forced to select one of only three certainty factors. A value of 1.0 is selected when the requirement is fully satisfied, and a value of 0.0 is selected when the requirement is not satisfied. This forces the user to ascertain one of two possibilities: either the module will perform a given function or it will not. When the user is uncertain as to the expected performance of the module, then a 0.5 is selected. This precludes the user from making arbitrary choices such as, "I am sixty-six and two-thirds percent sure this will work".

The same principles apply when the lattice is being used to define Joint Test requirements. A 1.0 is assigned if the function is required, 0.0 if it is not required, and 0.5 if the requirement for that function is undetermined at that time. Note also in Figure 10 that INNOVATOR allows the certainty factors to be altered as new information becomes available.

5.2 Operational Scene Depiction

The purpose of the operational scene depiction window is to portray a graphic representation of the scenario being constructed. While the requirements lattice is the basic method for entering the data, the space available on the screen becomes a limitation. As the lattice becomes very large the user will begin to lose track of key elements in the scenario, especially if extensive scrolling is needed. The scene depiction, therefore, is a tool that is intended to assist the user in keeping track of the big picture, as well as any high-level impacts caused by entries made at more detailed levels lower in the hierarchy. The interface between the requirements lattice and the scene depiction window is essentially a preselection scheme whereby certain items within the lattice are passed to the scene depiction window as they are selected. The specific objects themselves are determined by the level of granularity that the user selects. For example, it may be desirable to simulate two mechanized divisions in combat. However, there are models that simulate one division vs. another, or three brigades vs. three brigades, or nine battalions vs. nine battalions. INNOVATOR needs to know the level of granularity the user desires in order to make the proper divisional-level combat simulation selection and scene presentation. For instance, division-level granularity would require less resources in terms of storage and processing than would a division combat simulation with granularity at the company or platoon level. However, the user must consider the fact that a simulation with finer levels of granularity is likely to be more accurate than one with higher levels.

In order to build as accurate a representation as possible, INNOVATOR incorporates knowledge regarding distances, terrain features, etc. In order to make the scene representation appropriate to the level of granularity selected. For instance, if division-level conflict granularity is chosen, the scale in the scene depiction window will indicate a distance of 150 kilometers between opposing divisions. If company-level conflict granularity is selected, then the scene would reflect a distance of five kilometers between the opposing forces.

Lastly, the scene depiction in the INNOVATOR prototype is made up of icons, which are limited in the amount of knowledge they can represent. For this reason, INNOVATOR has been designed to allow additional information regarding each icon to be displayed adjacent to the icon by placing the cursor on the icon and clicking the mouse button. In the example shown in Figure 11, two friendly mechanized divisions and one motorized rifle division are matched against one enemy mechanized division, one motorized rifle division, and one aircraft squadron. Clicking the mouse button over selected icons results in a pop-up display indicating that the selected icon represents a certain type of division with an assigned certainty factor.

Time of day is displayed in the upper right hand corner of the window along with the weather. In this example, a clear display indicates daylight hours and the sun symbol indicates clear weather. It should also be noted that INNOVATOR can readily be programmed to use standard military symbols in addition to or in place of the icons (this is a future development milestone).

5.3 Resource Depiction

The resource depiction is implemented by INNOVATOR in the same manner that was used for the scene depiction. An icon is used to quickly convey to the user the required scale of the ADP host system, ranging from manual to supercomputer operation. Figure 12 shows an IBM PC depiction as a sample host system icon. Adjacent to the icon are displays indicating the amount of RAM storage required, the programming language, and any other special features and interfaces. Since ADP security is an important concern especially with regard to the military world, the resource depiction window also has been provided with an icon showing an open or closed padlock to indicate whether the model requirement is either unclassified or classified, respectively. The specific level of security (i.e., unclassified, confidential, secret, etc.) is shown directly under the icon. Collectively, the resource depiction window and the operational scene depiction window provide a quick-look at the modeling requirement under investigation. The requirements lattice window, on the other hand, provides a means of browsing through the specific pieces of knowledge represented by the icons in the depiction at a much finer level of detail.

6. CONCLUSIONS

The INNOVATOR expert system tool has been upgraded significantly in order to make the vast body of knowledge associated with the modeling/simulation problem domain more tenable from the standpoint of both the user and the programmer. The INNOVATOR knowledge acquisition taxonomy has been developed in accordance with established military operational and software asset management guidelines, thereby allowing the system to more closely emulate a real "expert." The incorporation of icon-based Visual Computer Engineering principles has resulted in the creation of a more meaningful and less cumbersome user/machine environment. The next step will be to append the asset management knowledge base and inference engine portions of INNOVATOR to the system, and proceed toward the goal of developing a complete expert system package to support the modeling/simulation function.

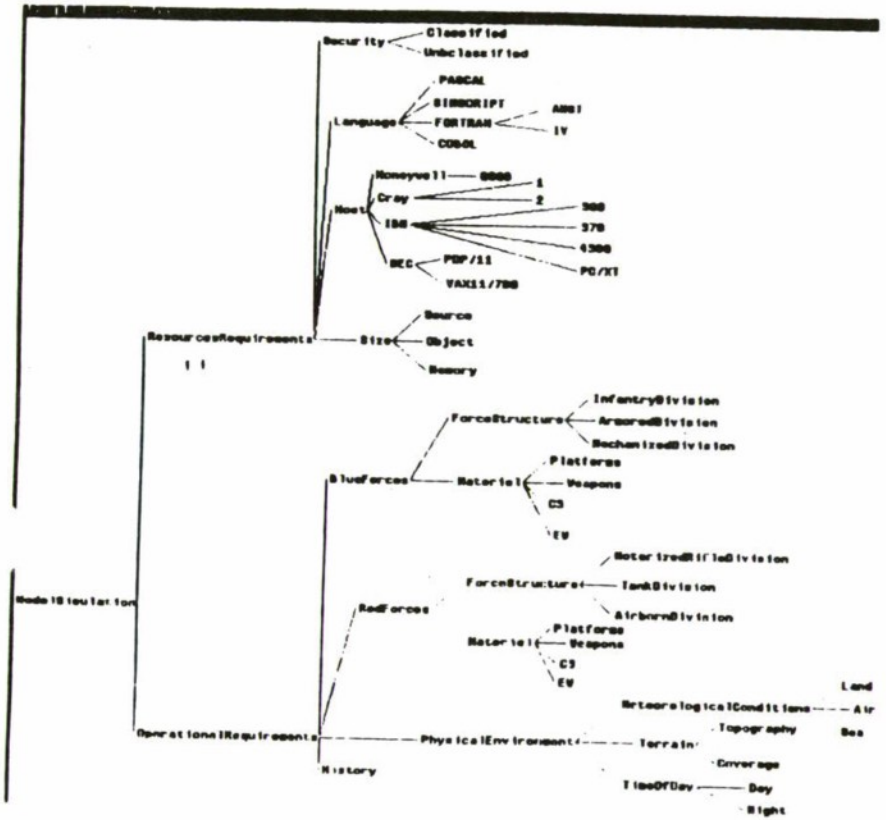


Figure 9. INNOVATOR Requirements Lattice

The lattice in the operational requirements branch is geared toward the elements of the structured scenario itself, i.e., blue and red force organization; material; mission; doctrines; tactics and objectives; readiness; physical conditions; history and intelligence. The lattice in the resource requirements branch is geared toward identifying programming languages, host characteristics, and storage and security requirements. The example shown is not intended to be a complete lattice, however the level of detail was sufficient to adequately address a hypothetical scenario involving two-sided airland combat at the divisional level.

Figure 10 illustrates some points regarding the selection process that occurs as the user traverses the requirements lattice. Say, for example, the user has a software module that needs to be entered into the knowledge base. The user starts at the top of the hierarchy and works through the lattice, selecting nodes that are appropriate to the module being entered. The user makes the selection by placing the cursor on the node and clicking the left mouse button. The user is then presented with a pop-up menu for selecting one of three certainty factors: 0.0, 0.5 or 1.0. The values were selected to

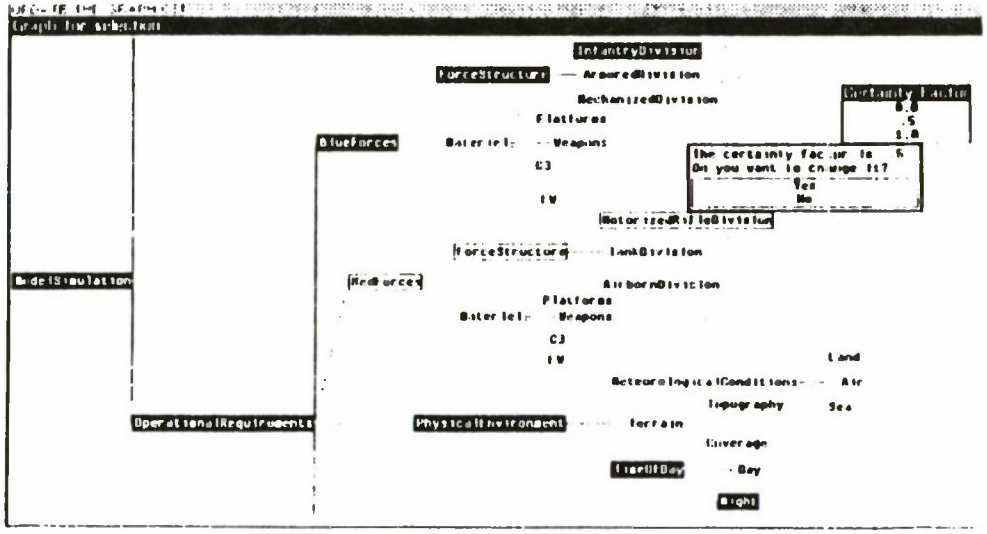


Figure 10. Certainty Factor Selection and Pop-Up Menus

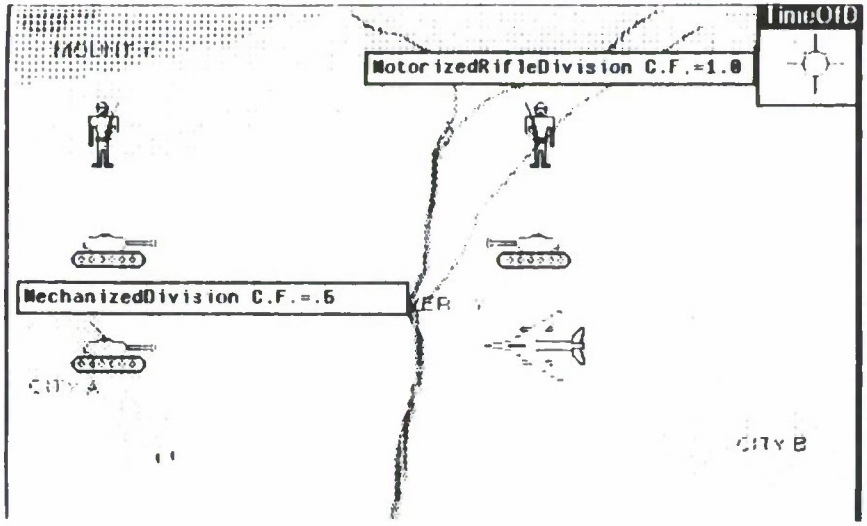


Figure 11. INNOVATOR Operational Scene Depiction

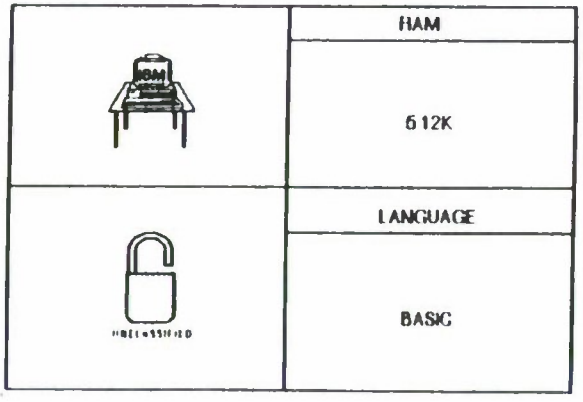


Figure12. INNOVATOR Resource Depiction

ACKNOWLEDGEMENT

This research is being supportd by OSD/DUSDT&E (subcontract under BDM prime contract F29601-84-C-0036). The information and opinions contained herein are provided by the authors and do not represent official government positions.

REFERENCES

[1] Joint Scientific Advisory Board, Minutes 3-4 February, 1983.

[2] The George Washington University Institute for Artificial Inteiligence, Joint Test Modeling Workshop Proceedings, Spring 1984.

[3] Staff, Joint Test and Evaluation (JT&E) Procedures Manual Modeling Modeling and Simulation Appendix, DoD/DUSD T&E, November 1985 (DRAFT).

[4] Silverman, B.G., "The Software Engineering Paradox: Unexploited Cost Savings and Productivity Improvements," Journal of Test and Evaluation, vol 5, no 1, January 1984, pp 11-16.

[5] Murray, A. J., An Analysis of DoD Models and Attribute Representation Techniques to Support the Development of a KBS for Model Design, October, 1984.

[6] Silverman, B. G., "An Expert System for the Management of the Modeling and Simulation Process," Proceedings of the 1984 National Test and Evaluation Conference, ITEA: Lexington Park, Md.

[7] Silverman, B. G., Mouratkis, V.S., "INNOVATOR: Representations and Heuristics of Invantor/Engineers," In B. G. Silverman, ed., Expert Systems and AI in Management, Addison-Wesley (forthcoming).

[8] Brown, Gratchan P., etc. al., "Program Visualization: Graphical Support for Software Development," Computer, August 1985. pp 27-35.

[9] Grafton, Robert B, "Visual Programming ," Computer, August 1985, pp. 6-9.

[10] Jacob, Rovert J. K., "A State Transition Diegram Language for Visual Programming," Computer, August 1985, pp. 51-59.

[11] London, Ralph L., and Robert A. Duisberg, "Animating Programs Using Smalltalk," Computer, August 1985, pp. 61-71.

[12] Melamed, B. and R. J. T. Morris, "Visual Simulation: The Performance Analysis Workstation," Computer, August 1985, pp. 87-94.

[13] Raedar, George, "A Survay of Currant Graphical Programming Techniques, Computer, August 1985, pp. 11-25.

[14] Foley, James D., Victor L. Wallace and Peggy Chan, "The Human Factors of Computers Graphics Interaction Techniques," IEEE Computer Graphics and Applications, November 1984, pp. 13-43.

[15] Smith, Michael J., "The Physical, Mental, and Emotional Stress Effects of VDT Work," IEEE Computer Graphics and Applications, April 1984, pp. 23-26.

[16] Block, Hed, Imagery, The MIT Press, Cambridge, Massechusetts, 1981.
[17] Hegen, Margarat A., The Perception of Pictures, Volume II, Academic Press, New York, 1980.
[18] Lodding, Kenneth N., "Iconic Interfacing, "IEEE Computer Graphics and Applicetions, March/April 1983, pp. 11-20.
[19] Morris, Peter E. and Peter J. Hampson, Imagery and Consciouanese, Acedemic Press, New York, 1983.
[20] Richardson, John T.E., Mental Imagery end Human Memory, St. Martin's Press, New York, 1980.
[21] Sheikh, Anees A., ed., Imagery: Current Theory, Research and Application, John Wiley and Sons, New York, 1983.
[22] Marcus, Aeron, "Corporate Identity for Iconic Interface Design: The Graphic Design Perspective," IEEE Computer Graphics end Applicetions, December 1984, pp. 24-32.
[23] Anderson, John R. and Gordon H. Bower, Human Associative Memory, V. H. Wineton and Sons, Washington, D.C. 1973.
[24] Ittelson, William H., Visual Space Perception, Springer Publishing Company, Inc., New York, 1960.
[25] Xerox Corporation, INTERLISP Refarance Manual, October 1983.
[26] USMC Command end Staff College, Landing Force Amphibious Operations Planning--Operetion MOSSBACK, 1981.

JAMS: An Expert System for RF Data Link Vulnerability Assessment

by

B.G. SILVERMAN J. SIMKOL
A. KAMRAN
C. DUBLIN

Institute for Artificial Intelligence
George Washington University
Washington, D.C. 20052
December, 1985

ABSTRACT

This paper presents initial results and future plans for development of an expert system for identifying and quantifying the anti-jam (AJ) capabilities and limitations (jamming vulnerabilities) of radio frequency (RF) data links when operating in a hostile electronic countermeasures (ECM) environment.

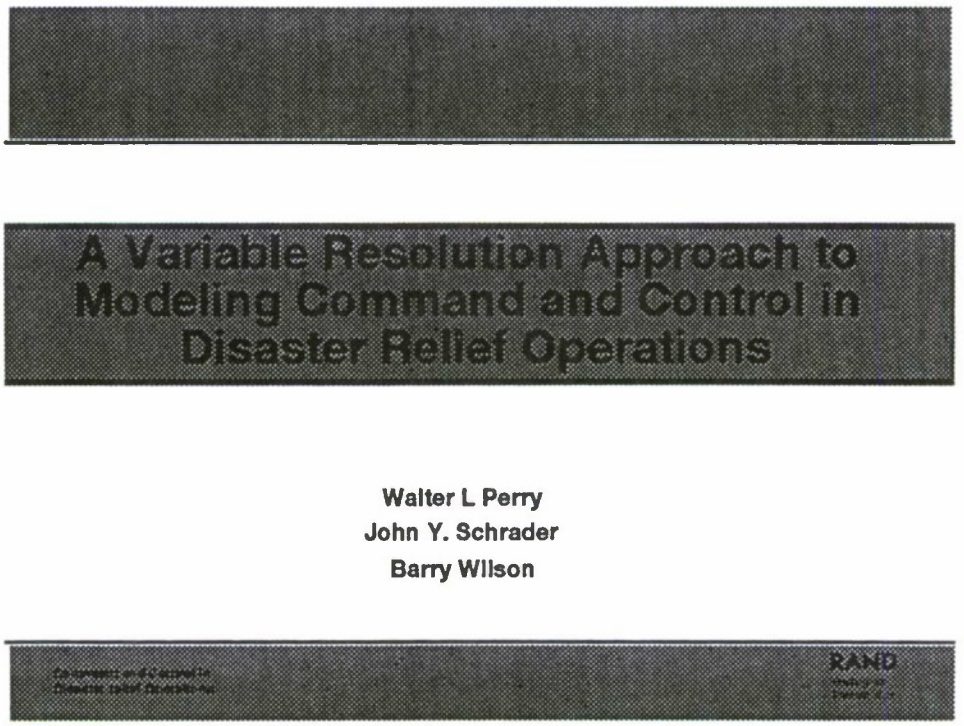


Chart 1

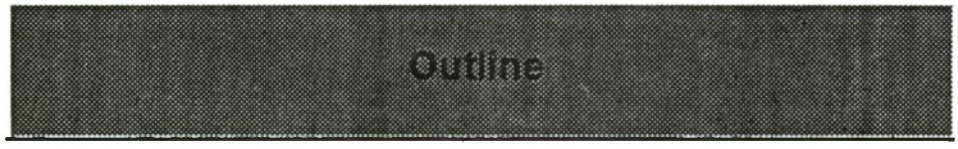
INTRODUCTION

This annotated briefing describes the use of variable resolution modeling techniques to understand the functioning of a proposed new system for reporting damage from national disasters and in directing subsequent relief operations. The modeling technique illustrated is an example of a more general process of using an existing modeling testbed to rapidly generate and evaluate alternative concepts of operation for the command and control of critical processes. A detailed discussion of the methodology is presented in the forthcoming RAND publication, *Decision and Control In Combat Operations*, W.Perry, 1992.

The work focuses on the construction and modification of command and control (C^2) models designed to assist in the development of concepts and procedures to support disaster relief decisions and to control their execution. The objective is to suggest a process to evaluate alternative operational concepts against a standard measure of information quality. This includes such elements as completeness, relevance, timeliness, accuracy, etc.

The construction process follows a top-down approach starting with a crude representation of the essential C^2 network, operational procedures and operational processes. Subsequent modifications consist of recursively increasing the level of resolution at critical network facilities and information processing activities. The objective is to obtain a clearer understanding of which system components are degrading the quality of information. Modifications can then be made by adding and/or deleting information processing facilities, alter the relationships among the facilities, modify the operating procedures within the facilities, and/or increase the information gathering process.

This approach allows for the analysis of C^2 systems to examine processes at differing levels of resolution while maintaining a fixed level on the information being processed. The methodology is top-down in that aggregate processes are examined first with detail being added only as necessary and only at those nodes and facilities where detail is needed to understand the causes of the degradation in performance.



- C2 in Disaster Relief Operations
- Operational Relationships
- Operational Procedures
- Information Requirements
  - Input Data
  - Processing Rules
- Results



Chart 2

OUTLINE

We begin with a brief discussion of emergency disaster relief operations. The focus is on the resources available to federal, state,and local governments to direct disaster

relief operations. This includes the several agencies and organizations generally involved in these operations. These agencies are clearly non-homogeneous, thus complicating the command and control effort. As a result, use of a modeling technique which allows for the rapid assessment of alternative operating concepts at varying levels of resolution, extremely useful in simulating disaster relief operations.

Next, we present a general discussion of the elements of command and control. These consist of the operational relationships among the various C^2 facilities; the operational procedures employed within each facility; and the information requirements. This is followed by a more detailed presentation of the disaster relief C^2 system and the application of the elements of command and control to this system.

Finally, we describe the model used to evaluate operational concepts for the command and control of emergency disaster relief operations. The model is simple and not intended to represent actual procedures in every detail, however, at the lowest level of resolution, it is reasonably accurate.

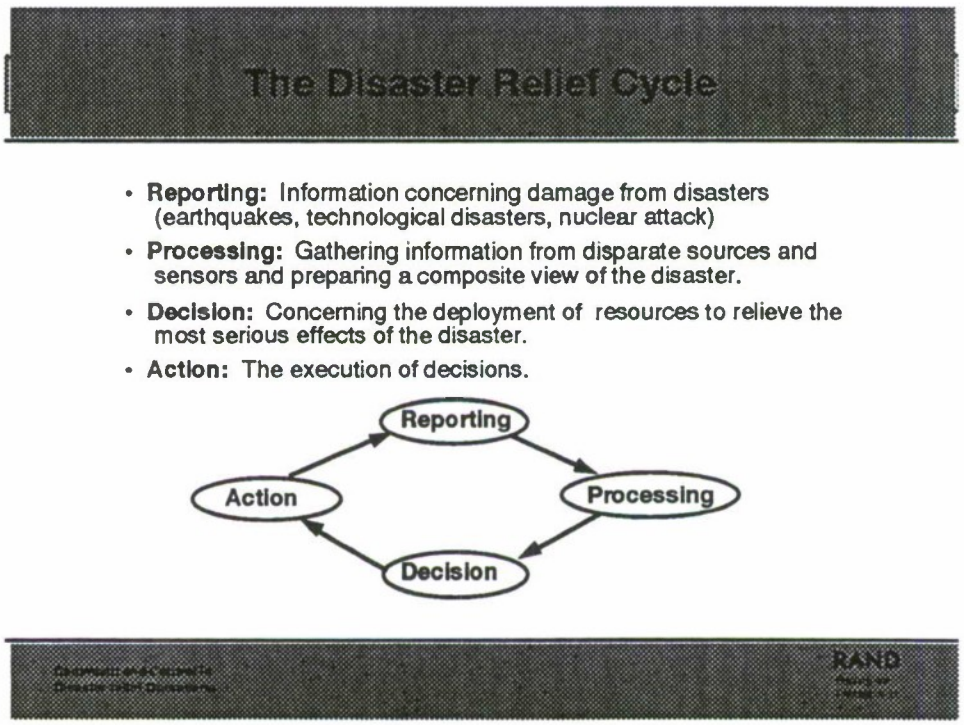


Chart 3

DISASTER RELIEF

The Disaster Relief process is much like a military operation. It includes planning prior to an event, situation assessment, mobilization and deployment of relief assets and

organizations, the execution of relief operations, and the subsequent recovery of deployed assets. Central to relief operations is the cycle of operations depicted in Chart 3. Information is gathered from sources close to the disaster. This information is processed and fused with other information available to the processing facility and a composite view of the disaster develops. Based on this view of the disaster, decisions are made to deploy resources to those sites most urgently requiring relief. Actions are taken to execute these decisions and their effects are subsequently observed by the sources and sensors and the process regenerates.

The real issue centers on the architecture associated with this simplistic representation of the operational cycle. Furthermore, even if the architecture were settled, the question of operating relationships and procedures needs to be examined. That is, the issues associated with the command and control of disaster relief operations. Chart 4 states the problem succinctly and the discussion presents some of the current activity within the federal state and local governments to address these issues.



Develop an **operational concept** for the command and control of disaster relief operations which effectively utilizes the capabilities of Federal, State and Local agencies to provide quality (timely and accurate) **information** at all decisionmaking levels.



Chart 4

PROBLEM

Chart 4 states the central issue associated with the command and control of emergency disaster relief efforts. The problem is one of defining a workable architecture,

defining a suitable operational concept for directing relief activities, and the definition of essential elements of information required to inform decisions. The following discussion focuses on the current status of these activities.

National Damage Reporting

Many resources are available to observe the effects of damage from major disasters. Satellites, aircraft, and ground-based observers can all provide elements of information that can be used to construct a comprehensive picture. Planning for disaster response has been fragmented with integrated national planning for nuclear attack and distributed planning for floods, earthquakes, and other natural and technological disasters. Recent studies sponsored by the Defense Nuclear Agency (DNA) and the Federal Emergency Management Agency (FEMA) have examined the expected performance of existing plans and procedures. Discussions with emergency planners at the local, state, and national level have led to the formulation of a new "seamless" approach to reporting damage effects. The new architecture requires integration of information from military and civil sources and increased sharing of information. Since information sharing requires links (hardware, software, and procedures) that may not exist, the value of nodes and links, in terms of the quality of information provided, needs to be understood.

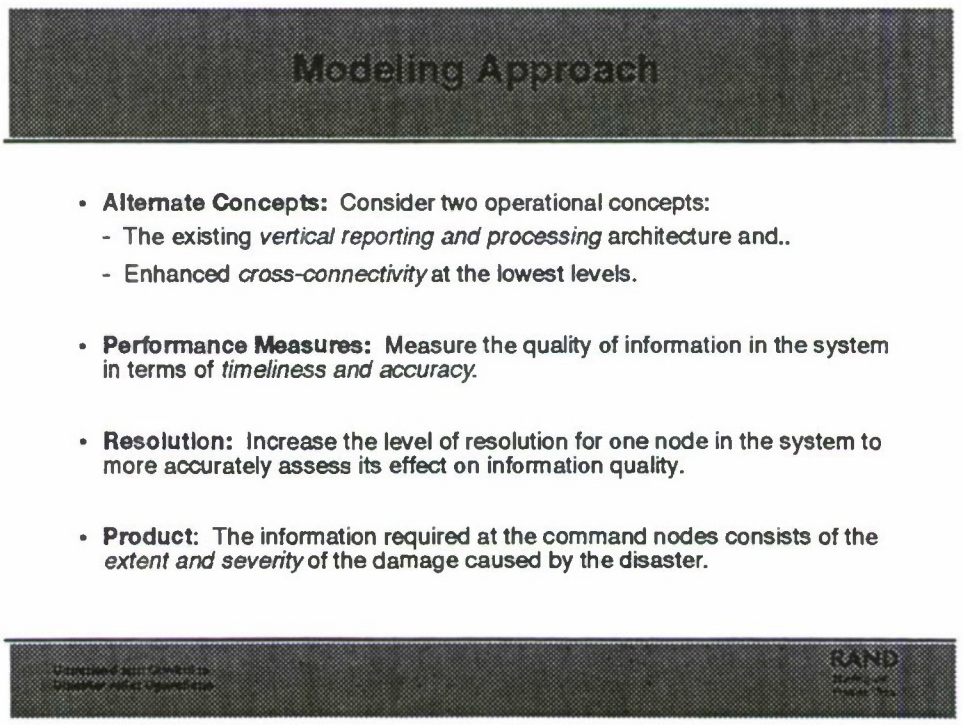


Chart 5

MODELING

The Alternate Concepts

Three parallel vertical reporting systems currently exist: state, federal civil, and military. Each contain data fusion nodes that can perform the functions of synthesis and consolidation of information or they can simply function as relay nodes (See Chart 9). As events occur, they are observed by sensors that report within their principal hierarchy, unless specific interconnections (paths) exist. Consequently, an alternative to the current concept is to add connectivity and accompanying operating procedures at the lowest levels in the reporting and information processing facilities.[1]

Performance

System performance is related to the accuracy and timeliness of reports available at the top of hierarchies (Governors and the President). However, the quality of information at nodes other than the top are also critical since the entire network represents interdependencies which cannot be separated. Consequently performance is also measured at intermediate nodes. This provides the criterion for increasing the level of resolution.

Level of Resolution

Increased resolution in this context, is not achieved through the introduction of more detailed data. In fact, the resolution of the data is considered fixed. In assessing the performance of alternative C^2 operating concepts, it is more desirable to vary the representation of the operating procedures in the model. For example, we might find that representing a fusion facility as a single node with a variable processing time and accuracy expressed in terms of a probability distribution, is not sufficient. Increasing the level of resolution might consist of anything from decomposing the process into several sub-processes to representing the facility as a network embedded in the larger C^2 system.

The Product

The product is the focus of the reports being forwarded to the decisionmaker. In this application it consists of the extent and severity of the damage being reported. These are in effect, the essential elements of information required to inform the decisionmaking process at the Local, State and Federal levels.

---

[1]As previously noted, this characterization is simplified. In practice there are existing interconnections at various levels. In any case the methodology can be applied to more complex systems where it can be used to understand the contribution of specific interconnections.

The Robustness of this Approach

This approach helps in system definition and provides analytical support to facilitate interdepartmental cooperation, if improvements in command and control procedures can be shown to lead to better quality decision support information. The variable resolution process is included in a comprehensive framework for analyzing C^2 issues and its application to situation assessment, decision and action. The objective is to provide a methodology for developing intelligent control mechanisms which are aimed at producing decisions and actions which favorably effect operations. In this context, we focus on a methodology designed to address two basic questions: *How does an employed command and control system effect overall operations?* and *How can we design and build a more effective command and control system?* These questions spawn a set of more analytically useful operational questions such as: *What effect do we want the command and control system to have on operations? How do we measure the effect?* and *What is meant by a more effective command and control system?* The emphasis in all of these questions is on intelligence, decisions and actions and the communications connectivity used to move information. The analysis of these issues proceeds at variable levels of resolution.

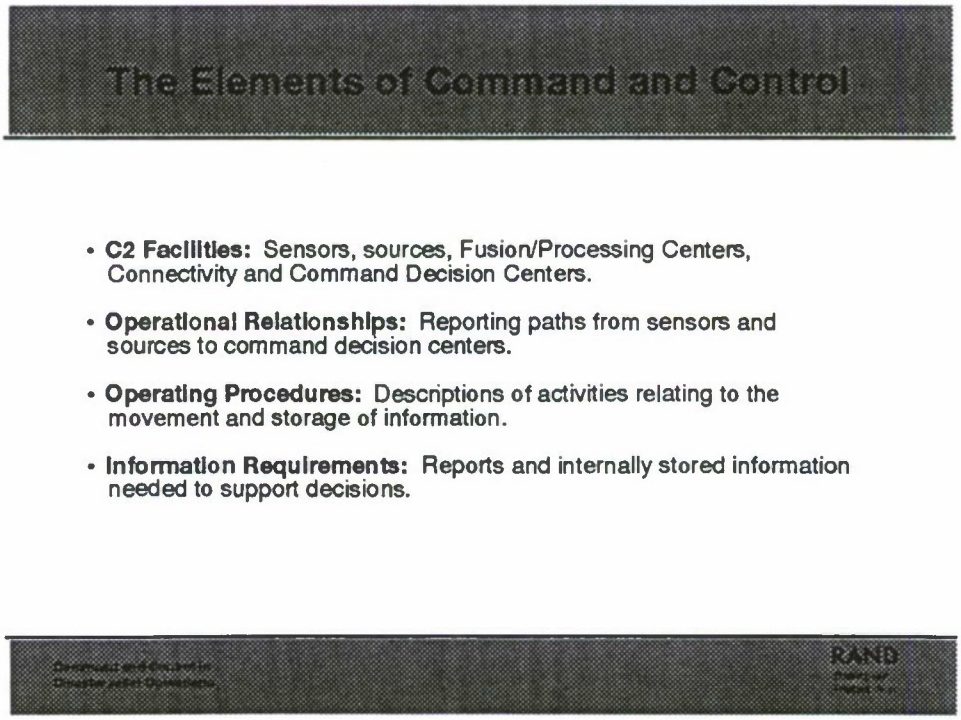


Chart 6

THE ELEMENTS OF COMMAND AND CONTROL

We now focus on the command and control process more generally. The emphasis in the next few charts will be on the fundamental elements of command and control, and the process of analyzing command and control. This is essential to understanding the model developed to analyze alternate operational concepts for the command and control of disaster relief.

Command and Control

Command and control is presented in terms of the functions it is designed to perform. The purpose is to motivate the development of a modeling paradigm for studying C^2 issues and not one that focuses solely on connectivity and information flow. Command and control proceeds from sensory observations of the environment to the fusion of sensory outputs and the classification of the fused representation of the situation to the eventual command decision. The ability of the modeling process to examine the operation of processes and relationships at varying levels of resolution is essential to the development of an effective operational concept.

The C^2 system is viewed as having three basic components: facilities such as sensors, communication equipment and command posts; structure, consisting of a physical network; and procedures which define the functions the system is to perform, the information requirements of the system and the operational procedures for processing and moving information in the system. From these components, we abstract the elements of command and control listed inChart 6.

In addition to the C^2 system, a C^2 concept of operations exists to support decisions and actions which must be made in support of the overall mission. The C^2 concept of operations addresses the principal tasks required of the C^2 system to support the functional mission (in this case, disaster relief). It consists of four basic elements which address both the structure and procedures:

- **C^2 Facilities:** These are the physical nodes in the network. They consist of the sensors, fusion/processing facilities communications facilities and the decisionmaking elements of the system.

- **Operational Relationships:** The nodes (or facilities) within the C^2 system are logically connected to reflect the reporting requirements needed to support the

functional mission. This connectivity defines the operational (not necessarily command) relationships.

- **Operational Procedures:** Operational procedures are the rules which are applied to the information arriving at a node in the C^2 system network to support both intermediate decisions and to update the status of state variables in the system.

- **Information Requirements:** Each facility in the C^2 system network receives information from either the environment or other nodes in the network. The character of the information may vary, but it is basically of two types: (1) information directly associated with the generation of a report; and (2) cueing information indicating that the sensors associated with the receiving node should engage in active sensing.

A critical element in conducting analyses of command and control is clearly the nature of these C^2 elements. With the mission, the nature of the disaster, the disaster relief resources and the required decisions assumed to be constant, the performance of the C^2 system can be evaluated by varying all or some of these.

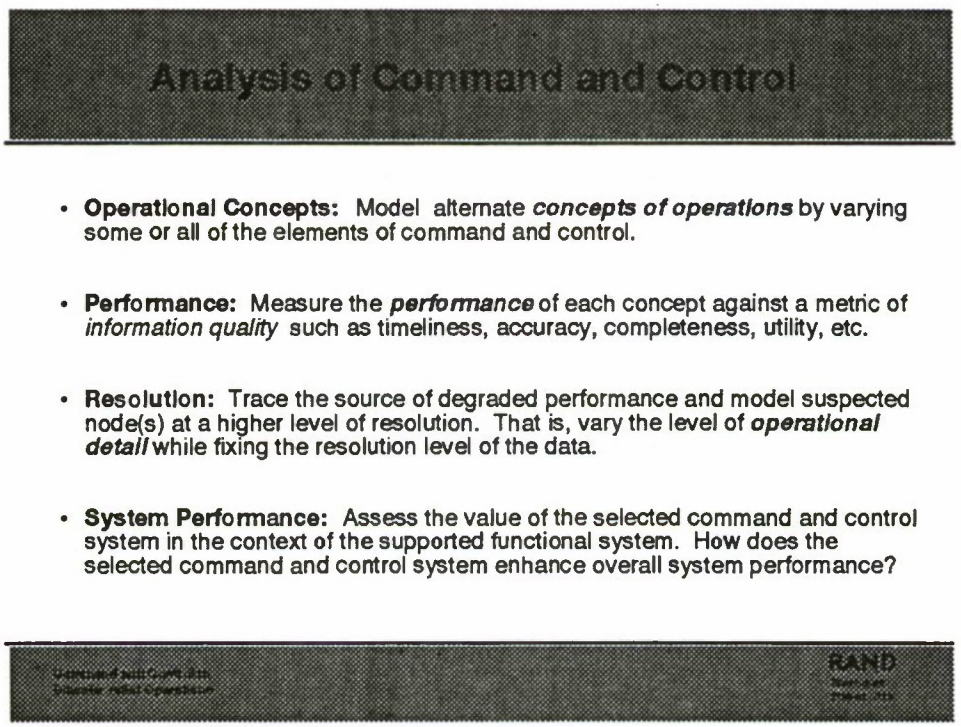


Chart 7

THE ANALYSIS OF COMMAND AND CONTROL

The operation of the C^2 system is central to the analysis of command and control. Typically, we can expect that the facilities available to support the C^2 system and the functions it must perform are given and the analytic objectives are to develop an operational concept which maximizes mission effectiveness in some way and which also maximizes the performance of the C^2 system.

A useful form of analysis is to evaluate alternative concepts of operations. The mission, the environment, the disaster relief resources and the C^2 system facilities are assumed to be fixed and elements of the system structure and procedures are allowed to vary, that is, the remaining three elements of command and control depicted in Chart 6:

- **Operational Relationships:** varying operational relationships is equivalent to varying reporting channels. The impact of skipping an echelon on the accuracy of information or the impact on timeliness of adding additional reporting layers can be assessed. This may also impact on the physical and logical connectivity of the C^2 system network.

- **Operational Procedures:** Procedures used to produce reports to support the system functions, and the conditions required for their transmission to other nodes in the network can be varied. The effect on system performance can be to vary the quality of information and system reliability.

- **Information Requirements:** Varying information requirements can vary the requirements placed on the sensor suite and on the processing at the nodes in the network. Typically more stringent requirements for information increases accuracy at the expense of timeliness.

APPLICATION TO EMERGENCY DISASTER RELIEF

As a first step in applying this methodology to emergency disaster relief, we discuss the command and control elements discussed in Chart 6 in terms of the emergency disaster relief process. These include a brief discussion of the C^2 facilities, a description of the operational relationships among these facilities, the operational procedures followed within each facility as it processes and moves information, and a definition of the information elements visible to the sensors.

Next we need to define the structure of the reports to be generated and transmitted. The facilities responsible for their generation and transmission and the recipients of these reports are defined in the operational procedures. Since the quality of information is to be measured in terms of the timeliness and accuracy of reports (Chart 4), we must also provide estimates for the time required to produce reports and a suitable measure of information accuracy. The former is usually stated as a probability distribution, and the latter is generally a function of the degree to which the reported information differs from actual ground truth. In order to treat accuracy in the model, we must also specify the reliability of sensors and processing facilities.

Finally, certain assumptions are made about the functional mission, the availability of the resources to provide relief to victims of the disaster, and the nature of the disaster itself. In addition, we establish an initial level of resolution for each of the C^2 facilities included in the C^2 system. For example, one sensor in the system is the police. Once the number of police nodes has been established, the procedures followed by each must be established. Questions concerning what the node can "see", how reliable his observations are, and the time required to make an observation and produce a report must be addressed in some detail. This sets the initial level of resolution. Adding detail

involves expanding on the procedures followed by the police node and not by varying the detail associated with what it sees.

The next few charts outline these elements for the simple emergency disaster relief problem.

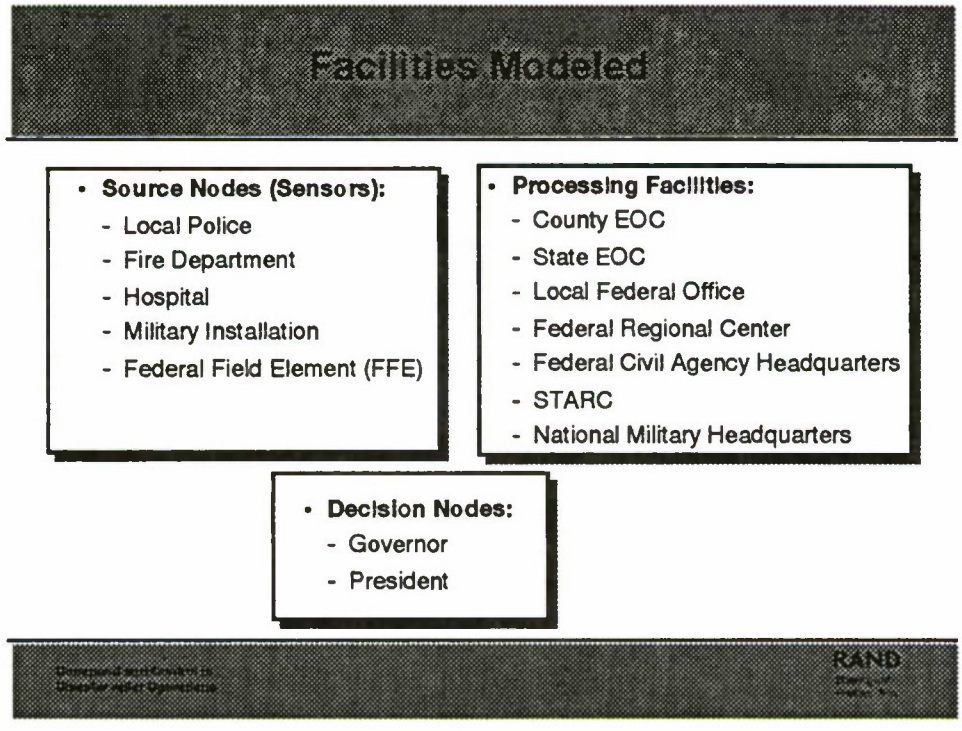


Chart 8

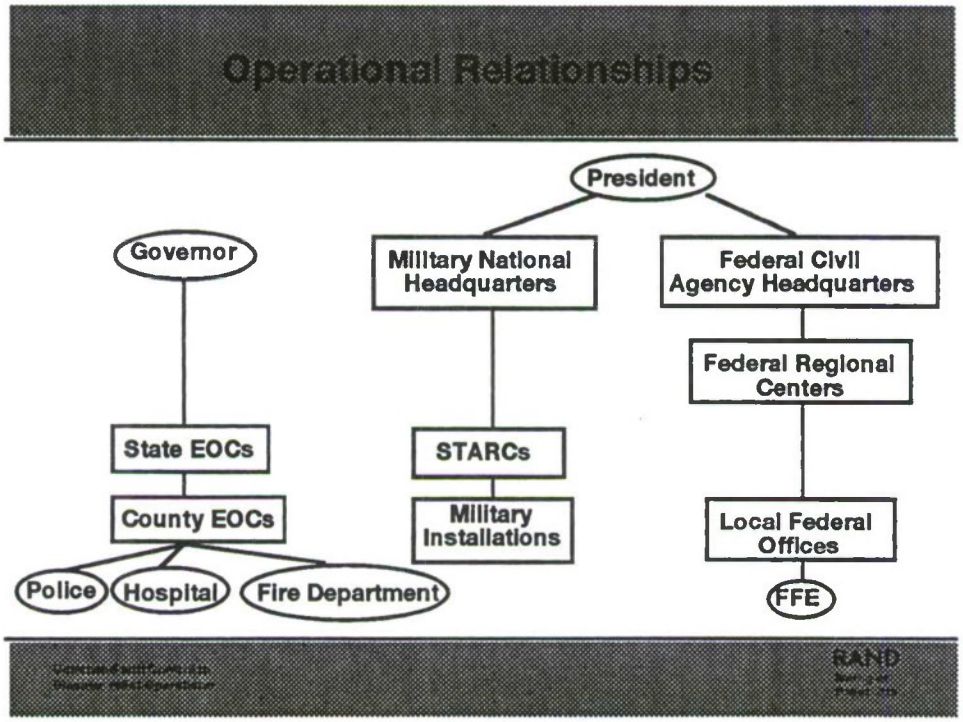


Chart 9

OPERATIONAL RELATIONSHIPS AMONG DISASTER RELIEF FACILITIES

Chart 8 lists the disaster relief sensor, processing and command facilities used in the model and Chart 9 describes the operational relationships among these facilities. The following is a description of the facilities and reporting requirements imposed on the system.

Disaster Relief Reporting Facilities

Information flows in disaster response begin with observers at or near the scene of a disaster. For this simple model we will consider local police forces, local hospitals including their emergency service vehicles, local fire departments, military installations with their ambulances and other emergency vehicles, and federal field elements such as FBI offices or Department of Agriculture county agents. They perform other functions beyond reporting damage but we will only consider procedures for reporting the effects of a disaster as they acquire information. The quality of information reported will depend on the reporters ability to accurately observe damage and to communicate their observations to another node.

Local police can be expected to observe all aspects of a disaster: likely cause, severity, number of people affected, and resources needed to mitigate the disaster's effects.

In our model the local police (actually all the vehicles and personnel assigned to a police station or dispatcher) forward all information on a disaster to their county Emergency Operations Center (EOC). The police station can also be expected to respond to queries from the county EOC for more data.

Local hospitals have a narrower focus than the police. They are expected only to contribute information on casualties. The police would also be reporting on casualties but we would expect reports from hospitals to be more accurate than those of the police regarding physical consequences of disaster. On the other hand police reports may be more accurate in estimating the total number of people affected. Like the police we assume that hospitals will make casualty reports to the county EOCs. Hospitals will also respond to queries for additional data.

Fire departments are the third and final element class in our model that reports damage directly to the county EOC. Reports are expected on fire damage and projected fire spread. They also respond to queries.

Military installations near the site of a disaster may be working with local police, medical, and fire fighting personnel but their reporting structure is independent. As military units observe damage they will make reports to their superiors. There may be more than one command that receives reports from military installations but we will only address the primary superior command with responsibility for military support to civil authorities. These units are called State Area Commands (STARCs) and are staffed with reserve component personnel.

Non-military federal departments and agencies have many local personnel throughout the country. These are generically called Federal Field Elements (FFEs). They report through local offices to Federal Regional Centers (FRCs) operated by the Federal Emergency Management Agency (FEMA). Collectively they can be expected to report on all aspects of a disaster, although any individual participant will have a specific set of skills that will influence the nature and quality of reported effects.

Processing Nodes

Our model has three processing nodes to perform the function of data fusion for reports from one or more source nodes. The first is the county EOC where reports from police, hospital, and fire department source nodes are first integrated. There are many rule sets that can be constructed for handling conflicting, contradictory, or partial information. A county EOC has a clear set of source nodes that submit reports. This is the first location where an integrated disaster status file is maintained. In addition to collecting reports and performing the data fusion function, the county EOC also prepares

an Initial Disaster Report for submission to the State EOC and then provides updates on a regular basis. In all cases the county EOC receives reports from the police, hospital, and fire units in its area of responsibility.

The state EOC functions as a processing node in a manner analogous to the county EOC operations. Data (reports) are received from the component counties and they are fused to form the Governor's Disaster Summary Report. Like other source and processing nodes it responds to requests for additional information from higher level nodes.

The STARC is a processing node in the military reporting hierarchy. It processes reports from all military installations in the state. It prepares a summary disaster report based on the individual installation reports. Its summary reports are submitted to the national military headquarters.

The federal civil reporting hierarchy processing nodes are the Federal Regional Centers that receive reports from local federal offices. Many departments and agencies with information on the disaster submit data to FRCs for fusion to form a Summary Federal Disaster Report for forwarding to national headquarters . The final civil information integration takes place at federal civil agency headquarters (in practice FEMA operates the national Emergency Information Coordination Center with representatives from all federal departments and agencies).

Command Nodes

Command nodes are the recipients of data from processing nodes. Our model shows information to support governors and the President. Governors rely on State EOCs to provide a basis for deciding which state resources to apply to the problem and deciding when to request support from outside. The President gets reports from the military and from federal civil agencies. Again the information is used to determine whether state resources can meet needs or whether federal support is warranted. Decisions generated in the presidential node include federal disaster declarations and orders to military forces to provide support to affected states.

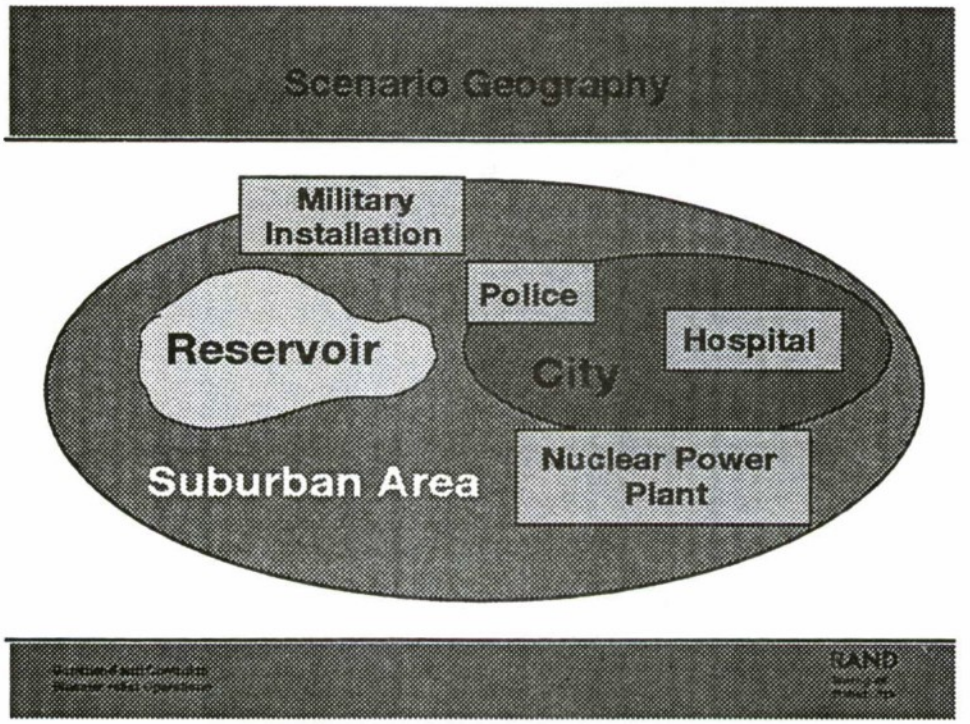


Chart 10

INFORMATION REQUIREMENTS

Next, we discuss the information requirements of the emergence disaster relief system. Chart 10 depicts a stylized geographic entity within a county. The military, the police and the hospitals serve both as sources of information about the disaster, and the means for providing relief to disaster victims. In this analysis, we proceed only to the point at which a decision is made. The action taken as a result of the decision is not modeled.

Our model uses five measures of disaster intensity. They are radiation level, water quality, infrastructure damage, casualties, and residual military capability. Each of these ground truth elements can be measured but the accuracy of the reported measurement is a function of the training of the observers, the sensor used to measure the effect, and the location where the measurement was taken. The model is intended to show the effect of alternative C^2 systems so it is necessary to know the actual value over time of each of the measures. More measures could be used but we have chosen this set as manageable and representative of the types of data required.

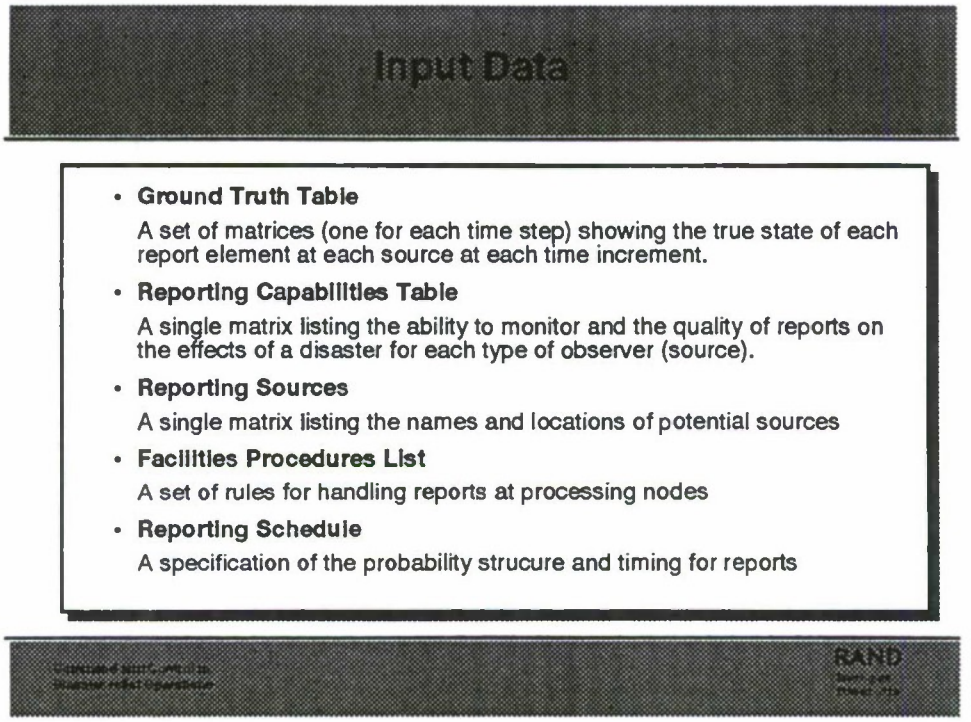


Chart 11

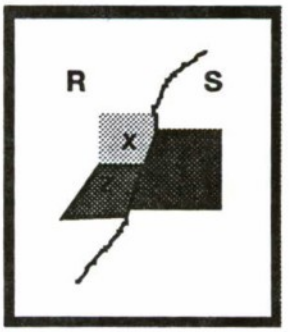
INPUT DATA

There are five basic sets of data required as input. With the exception of the Ground Truth table, each is associated with either the operational relationships, or the operational procedures. At any time during a run, the simulation may be interrupted and modifications may be made to any or all of the inputs associated with procedures or relationships. In addition to the inputs depicted here, additional information is required which is associated with the simulation itself. These include probability distributions for processing times, transmission times, and reliabilities.

Ground Truth Table

Final Ground Truth Values

	County X	County Y	County Z
	State R	State S	State R
Radiation	Major	Minor	None
Water Quality	50	10	2
Infrastructure Damage	30%	5%	0%
Casualties	20K	0.5K	0
Residual Military Capability	50%	100%	100%



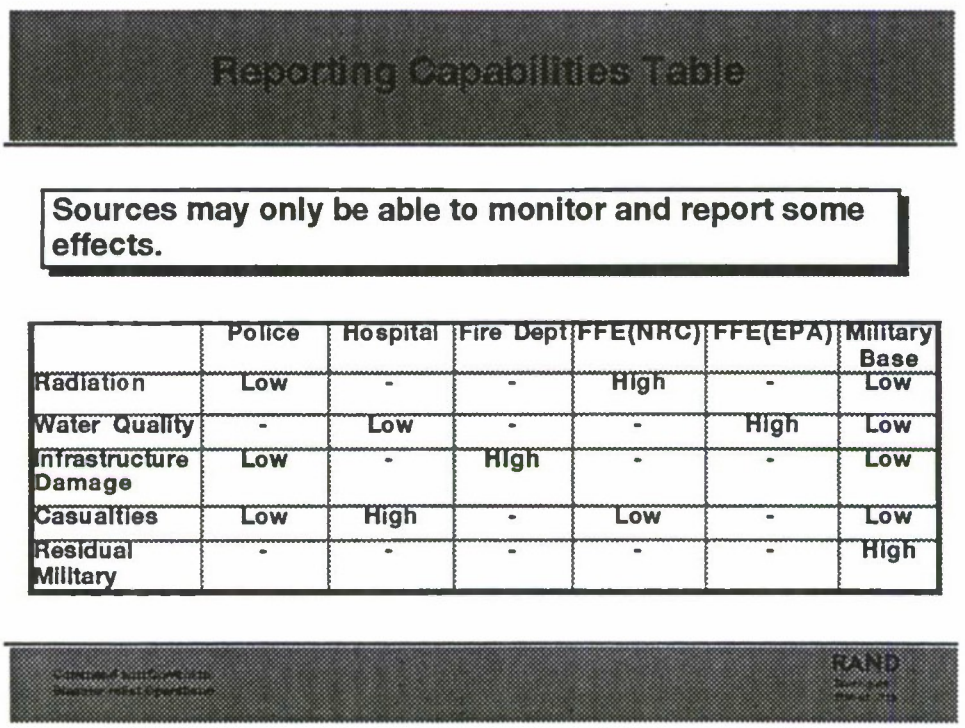
The situation deteriorates linearly to this state in one-hour increments for 24 hours after the earthquake occurs.

RAND

Chart 12

GROUND TRUTH

The first set of data needed to operate our model is a ground truth table for a region of interest. Our simple case includes areas of two adjacent states (R and S) with two affected counties in state R and one affected county in state S. Most damage occurs in county X but some effects are felt in the other counties. The table shows that some observations are qualitative (radiation) and others are quantitative. Actual radiation reading are numbers from measuring instruments but fusion of those readings are assumed to take place at the county level and a single qualitative value is reported. Experience levels or defective equipment may result in a reported value different from the ground truth value. This table of ground truth values is necessary to measure the accuracy of the reports available at other nodes.



	Police	Hospital	Fire Dept	FFE(NRC)	FFE(EPA)	Military Base
Radiation	Low	-	-	High	-	Low
Water Quality	-	Low	-	-	High	Low
Infrastructure Damage	Low	-	High	-	-	Low
Casualties	Low	High	-	Low	-	Low
Residual Military	-	-	-	-	-	High

Chart 13

Information reported from source nodes may be ground truth or it may only be the best estimate of the reporting agency. The Reporting Capabilities Table (Chart 13) illustrates that we must consider both the ability to report and understand the likely quality of reports transmitted. For example, in our table police are not expected to be able to report on water quality. The police are assumed to have some equipment to monitor radiation but the quality of their radiation reports is not expected to be as good as reports from a federal field activity of the Nuclear Regulatory Commission. In most cases the effect reported on is not part of the primary mission of the reporting agency.

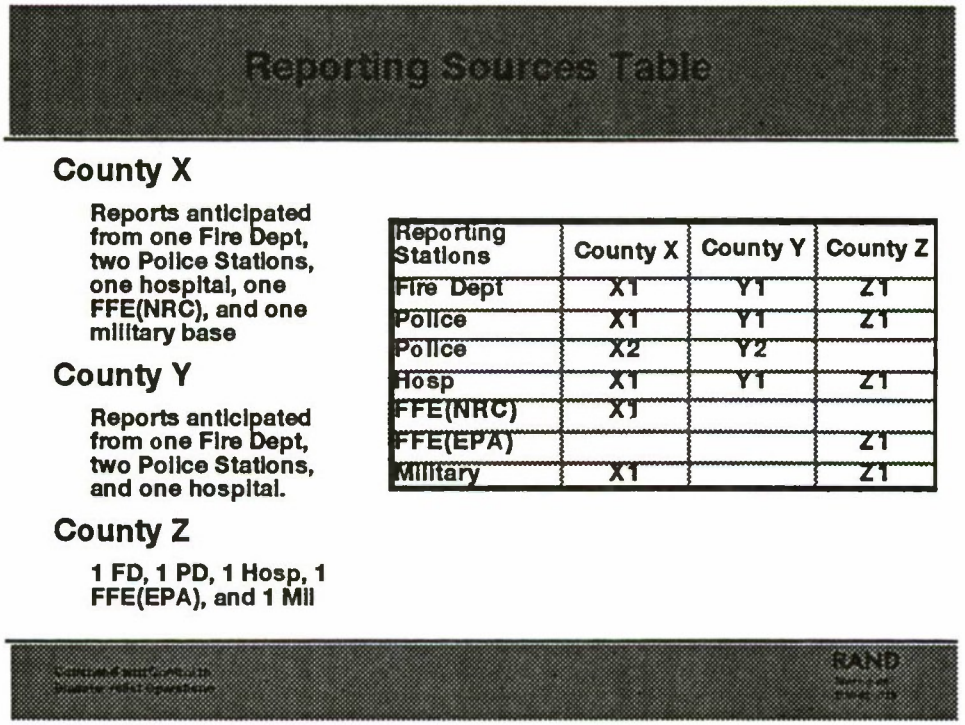


Chart 14

In addition to understanding the expected quality and types of report, it is necessary to know where reporting stations are located. Most counties will have fire and police departments that can submit reports but only some counties have military bases or offices of specific federal departments and agencies. Chart 14 describes the status of reporting facilities in the affected counties.

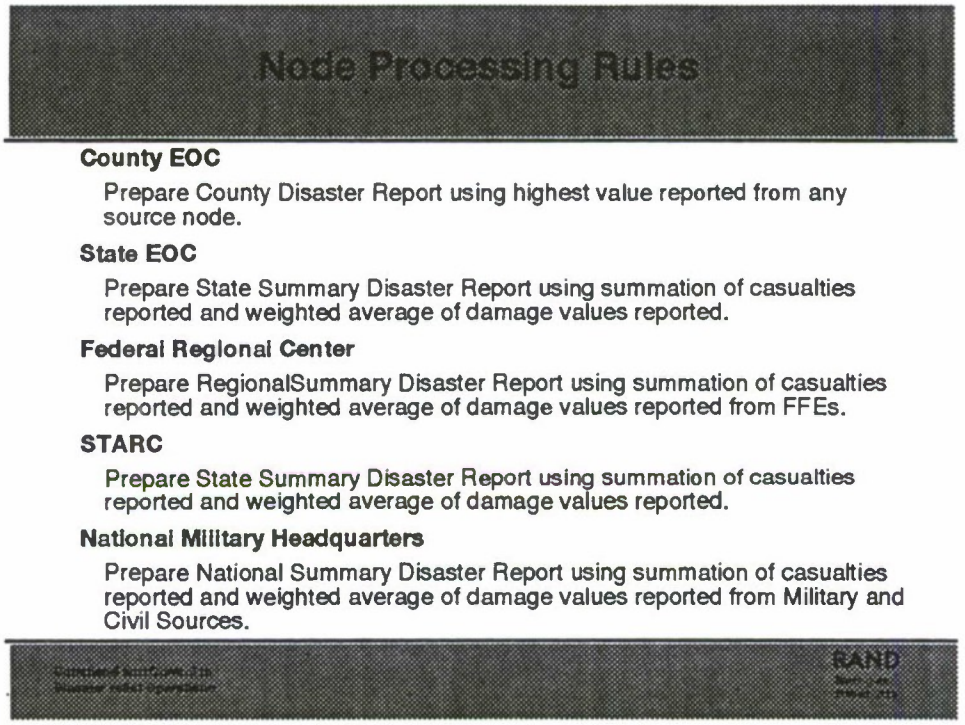


Chart 15

OPERATIONAL PROCEDURES

The procedures to be followed at each processing facility to generate reports from incoming information are specified in the form of *rule sets*. The rules specify such things as how disparate information is to be combined (fusion), how conflicting observations are to be resolved, and what criteria is to be used to determine when a report must be generated and forwarded. The output from this process is either a forwarded report or an updated internal status file. This latter file is used to produce a report once *sufficient*, reliable information is available or when some time requirement has been exceeded. These conditions are generally expressed as thresholds which must be attained.

Where uncertainties exist about timing or component reliability, random variables are used to assign values. Consequently, the results from the operation of the model are expressed in statistical terms after several replications of the model's operation. This allows us to establish confidence levels about a specific operational concept with regard to performance objectives. To obtain statistically significant results, several replications of the model operation on a specific case are necessary.

The following is a simplified description of the procedures followed by each of the facilities listed in Chart 9. It also describes the major reports generated and the reporting

relationships among the facilities. The rules used to described these procedures in the model are written in RAND ABEL™, an expert system programming language.[2]

Sources

Sources report on observations they make based on their capabilities (see Capabilities Table, Chart 13). For example, the police in County X may report on the number of casualties based on what they have currently observed. The procedures associated with transforming the observation into a report is trivial: what is seen is reported. The police, hospitals and fire departments forward reports to the County EOCs and respond to EOC inquiries. The Federal Field Elements (FFEs) forward their reports to the Federal Field Office (FFO) and the Military Installation forwards its reports to the local STARC. The reliability of the reports is determined by the probability distributions consistent with the ability of the observer to accurately report on a phenomenon as depicted in the Capabilities Table.

County EOCs

The County EOC's are the primary *processing nodes* within the state. They prepare the *Initial Disaster Report* and forwards it to the State EOC. This report consists of the most current information on each of the ground truth elements available at the time of the report. The value of each element is taken to be the most severe condition reported by the source nodes. Every hour, the center submits a follow-up. In addition, the center maintains a *Disaster Status File* which is used to update reports and to provide answers to queries from the State EOC. The County EOC's receive reports from police, hospitals and fire departments.

State EOCs

The State EOC's are intermediate *processing nodes* at the State level. They receive the Initial Disaster Reports from the county EOC's within the state and all follow-on updates. In addition, they may query the county EOC's for additional detail on specific information elements. The state EOC prepares the *Governor's Disaster Summary Report* and forwards it to the Governor's office once it is complete. This report consists of combined assessments of the information received from the counties. In this model, we

---

[2] RAND-ABEL™ is an expert system language developed at RAND as part of the RAND Strategy Assessment System (RSAS). The language compiles into C code and is used to write the war plan rule sets for the RSAS. A version of the language and accompanying software tools which can be used in other domains is also available under the name RAND-ABEL Modeling Platform (RAMP). For a more detailed description of the language, see N. Shapiro, et al *The RAND-ABEL Programming Language: Reference Manual*, RAND, N-2367-1-NA, 1989.

continue the simplistic paradigm of selecting the most severe status on the reported elements discounted by the reliability of the report.

The Governor

The Governor is the single *command facility* in the state. He receives the Governor's Summary Disaster Report from the State EOC and submits requests for additional information. Based on the reports he receives, he assesses the general extent and severity of the disaster and may decide to do nothing, or he may declare a state of emergency in the effected counties. This is issued as a *State Disaster Proclamation.* In addition, he may also submit a request for federal assistance to the President.

The extent and severity of the damage used as the basis of his decision is expressed in terms of the percentage of the state area damaged in some way, and the percentage of the reported facilities considered to be damaged. However, the rule used to activate this decision is not strictly binary. That is, the decision to request federal assistance, for example, is not always made just when certain thresholds are exceeded. Other unquantifiable factors may influence the decision. In the model, these are represented by probability distributions on the Governor's likelihood to act based on a perception of his aversion to risk.

Federal Field Offices (FFEs)

The FFE is the primary *source node* for the federal government in the state. Its members observe the damage directly and the facility prepares the *Initial Federal Disaster Report*. This is accompanied by a follow-up every hour the emergency persists. All reports are submitted to the Federal Regional Center in the state. In addition, the FFE's respond to specific inquiries from the state FRC.

As with the county source nodes, the report consists of observation made by the federal officials concerning the elements listed in the Capabilities Table (Chart 13). The reliability of the reports is also determined by the probability distributions consistent with the ability of the observer to accurately report on a phenomenon.

Federal Regional Centers (FRCs)

The Federal Regional Center is the primary *processing facility* for the federal government in the state. It receives the Initial Federal Disaster Report from the FFE's in the state and combines the reports to produce the *Summary Federal Disaster Report*. This report is forwarded to the Federal Civilian Agency Headquarters. The FRC submits inquiries to the FFEs, and responds to inquiries from the Federal Civilian Agency Headquarters. Like the state EOCs, Governor's Disaster Summary Report, the Summary

Federal Disaster Report consists of the most severe assessments of the data element reported.

Federal Civilian Agency Headquarters (FCAH)

The Federal Civilian Agency Headquarters is the federal level *processing facility* of disaster information submitted by the state FRCs. The Agency receives the Summary Federal Disaster Reports from the state FRCs and uses the information provided to prepare and transmit the *President's Disaster Summary Report*. In addition, the Agency submits queries for additional information to the FRCs and responds to inquiries from the President. As with the other reports, the "most severe status" paradigm is used to assess the status of reported elements.

The President

The President is the federal *command node*. He receives the President's Disaster Summary Report from the FCAH, and the President's Military Disaster Report from the National Military Headquarters. He may also receive requests for federal assistance from the Governor. In this simple model, the President is the only command or processing node where information from both military and civil sources merge. Clearly, a more interesting concept is one in which information sharing occurred at much lower levels.

The President may issue a state of emergency for the state in the form of a Disaster Proclamation. In addition, he may submit inquiries to the FCAH and the National Military Headquarters for additional information. The criteria used to make the proclamation is similar to the Governor's. That is, it is based on assessments about the severity and extent of the damage in the State and the degree to which the President is risk averse. The decision to provide federal assistance to the state is always made in response to the Governor's request.

Military Installation

Military installations located in the state also monitor the severity and extent of the damage. These facilities are the military *source nodes* for collecting information on the disaster. The military installations prepare the *Initial Army Disaster Report* which is submitted to the State Area Command (STARC). Follow-up reports are submitted hourly as the disaster persists, and the installation responds to inquiries from the STARC for additional information.

As with other source nodes, the military installations make observations of those elements listed in the Capabilities Table (Chart 13) consistent with their ability to accurately report on the phenomenon.

State Area Commands (STARCs)

The State Area Command is the military *processing facility* in the state. It receives the Initial Army Disaster Report from the military installations in the state and prepares the *Summary Army Disaster Report*. This report is transmitted to the National Military Headquarters. In addition, the STARC submits inquiries to the military installation for additional information and responds to inquiries from the National Military Headquarters. The "most severe status" paradigm is again used to summarize the status of reported elements.

National Military Headquarters

The National Military Headquarters is the national level military processing facility for disaster reports. It receives the Summary Army Disaster Report from the STARC, and uses it along with amplifications obtained through information requests to the STARC, to prepare the *President's Military Disaster Report*. The Headquarters also responds to inquiries from the President for additional information. The "most severe status" paradigm is again used to assess the status of reported elements and to estimate the severity and extent of damage in the state.

THE MODEL

The C^2 model used to describe the disaster relief operations discussed in this briefing consists of three basic elements: A Ground Truth Monitor, a Model Generator and a Simulator. The interactions among these three elements is depicted in Chart 16 and a brief description of each is presented below. A more detailed discussion of this modeling technique will be available in the forthcoming RAND publication, *Decision and Control In Combat Operations*, W.Perry, 1992.

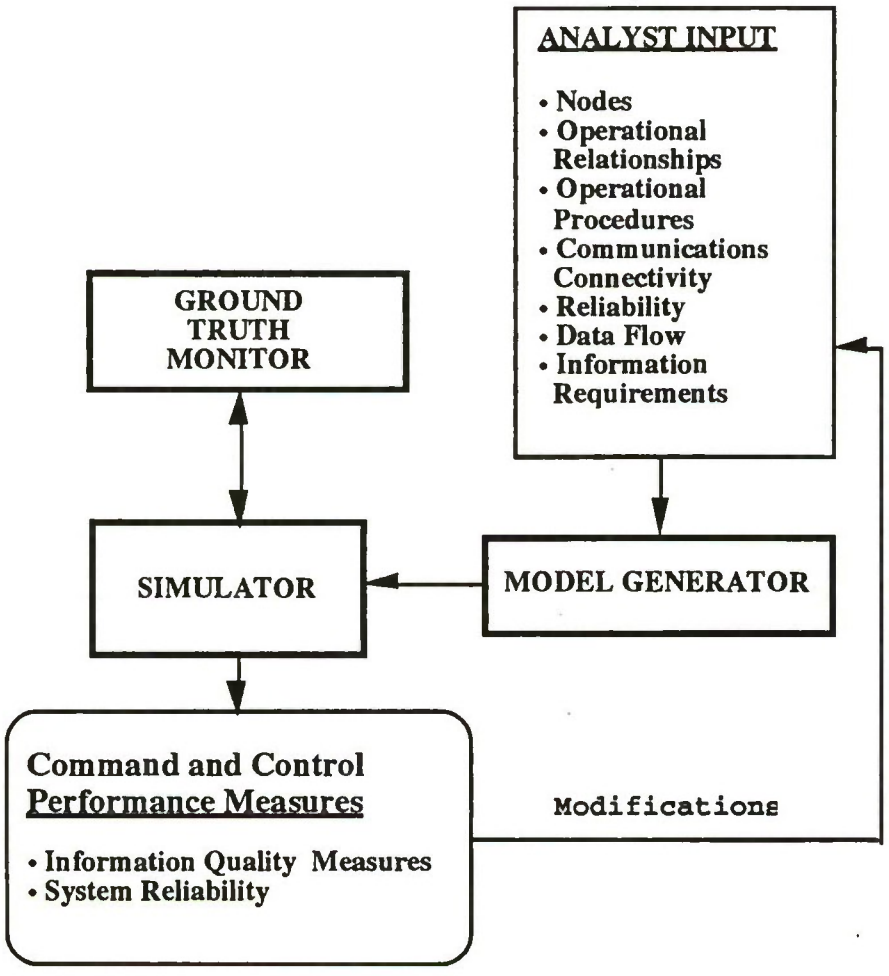


Chart 16

The Model Generator

This module performs the tasks of transforming a command and control system into a customized version of the Simulator. RAND-ABEL™ is used to represent both the physical aspects of the C^2 system and the decision rules associated with the operational concept. The module is designed to sequentially read a series of input files, some of which represent the rules associated with the operation of one or more system nodes, and one is used to define the data structures associated with the reports and internally stored data. Within these files, the analyst must specify descriptions of the elements of the system discussed above. These are more generally referred to as: the communications system, nodes or facilities, data flows, information content and reliability assessments.

The Simulator

The Simulator is a customized description of the command and control system and the rules established to process information through the system. It is described as customized because it consists of code which is produced in the Model Generator from the input description. Thus each different command and control system description input will produce a unique Simulator, tailored to support evaluation of the system described. The operation of the model means the execution of the Simulator.

The Simulator responds to stimulation from the Ground Truth Manager. The stimulations are essentially sensor suite detections which result from detectable events associated with the disaster. This starts the simulation process as the C^2 system processes the information. Processing consists of the application of the node functional rules and the assessment of timing delays associated with the processing and the transmission of reports over the system communications networks.

In addition to viewing the status of the C^2 system, the Simulator also allows the user to interact with the components of the system. The objective is to provide the user with the tools required to alter the operational concept in order to produce a modified C^2 model or to add more resolution at specified points. This is a crucial aspect of the process in that it provides a mechanism to rapidly assess competing operational concepts and system structures.

The Ground Truth Manager

The Ground Truth Manager simulates real world events. If the C^2 model were to be operated in conjunction with the simulation of real world disaster, the Ground Truth Manager would act as the interface to the external simulation model. The Ground Truth Manager generates the stimuli to which the disaster relief command and control system must react and the objects upon which the results of command decisions take effect.

The model can run in *batch* mode or *interruptible* mode. The batch mode is a Monte-Carlo run used to generate statistics which describe the performance of the C^2 system. In the interruptible mode, the analyst is able to freeze the action at any time and examine the status of the information flow, node processing, component status, etc. In addition, the analyst may also alter any aspect of the simulation, i.e., any of the elements constructed by the Model Generator.

AN EXAMPLE

A simplified version of the model was run using nominal data concerning the timing and reliability of processing facilities and based on the rule sets and operational

relationships described above. The following is a summary of the results of executing the simulation in the interrupt mode first, and then in the batch mode.

Interrupt Mode

The purpose of operating in this mode is to allow the analyst to examine the detailed operation of the system with the candidate operating concept. The objective is to modify certain components thus creating either alternate operating concepts or adding more resolution to certain activities.

Charts 17 to 19 depict the interrupt mode terminal interface. The display is an adaptation of the RAND Strategy Assessment System The window in the upper left is used to control the operation of the model. The smaller window within the control panel provides a running chronology of events as they occur. In this mode, the model may be run one time step at a time or a fixed number of time steps. The window at the lower left is the hierarchy tool. This displays the major node types in the network along with the reporting relationships among them.

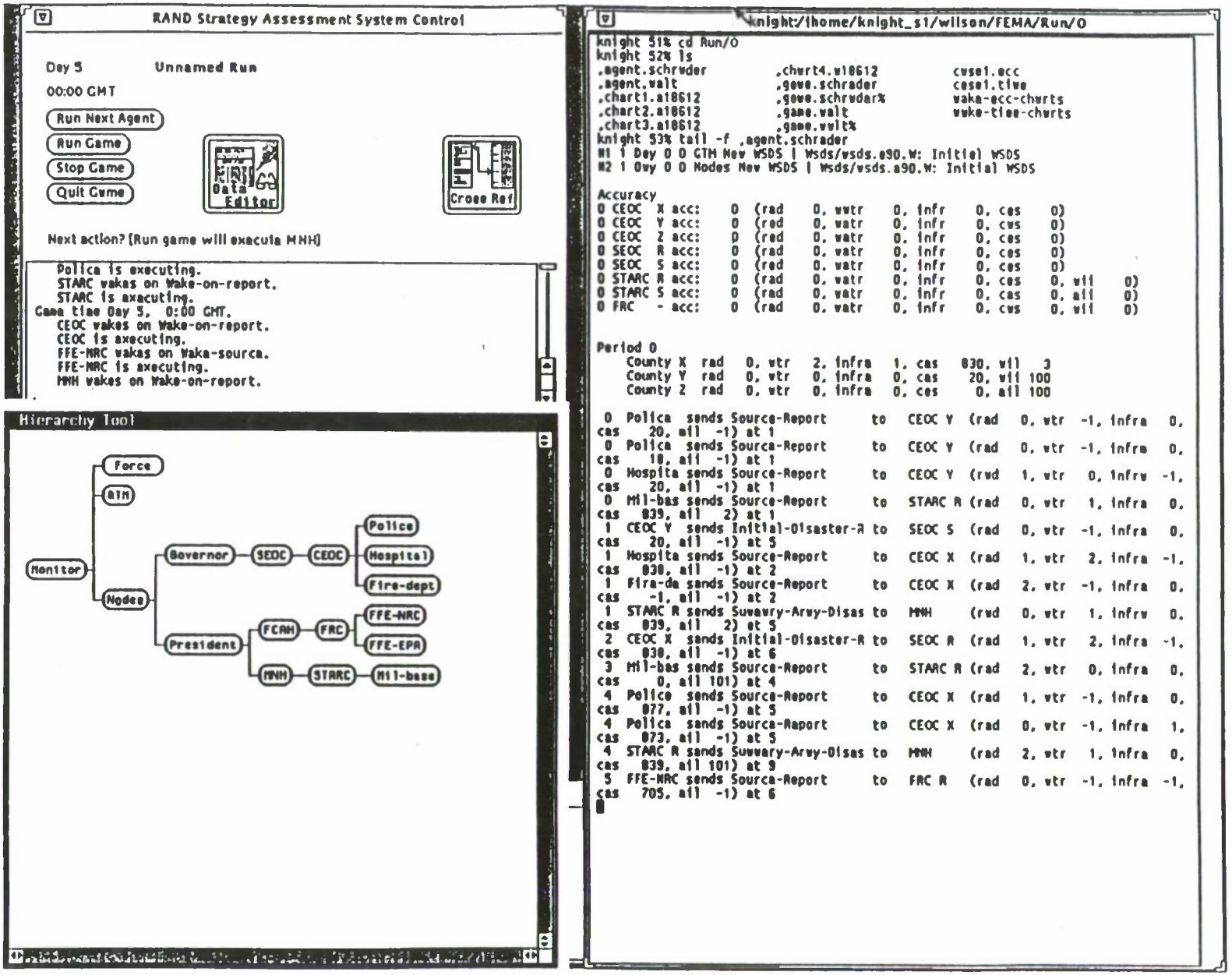

RAND Strategy Assessment System Control
Day 5
Unnamed Run
00:00 GMT
Run Next Agent
Run Game
Stop Game
Quit Game
Data Editor
Cross Ref
Next action? [Run game will execute MNH]
Police is executing.
STARC wakes on Wake-on-report.
STARC is executing.
Game time Day 5, 0:00 GMT.
CEOC wakes on Wake-on-report.
CEOC is executing.
FFE-NRC wakes on Wake-source.
FFE-NRC is executing.
MNH wakes on Wake-on-report.
Hierarchy Tool
Force
BTM
Monitor
Governor
SEOC
CEOC
Police
Hospital
Fire-dept
Nodes
President
FCAH
FRC
FFE-NRC
FFE-EPA
MNH
STARC
Mil-base
knight:/home/knight_s1/wilson/FEMA/Run/0
knight 51% cd Run/0
knight 52% ls
.agent.schrader .chart4.a18612 case1.acc
.agent.walt .game.schrader case1.tive
.chart1.a18612 .game.schrader% wake-ecc-charts
.chart2.a18612 .game.walt wake-tiee-charts
.chart3.a18612 .game.walt%
knight 53% tail -f .agent.schrader
#1 1 Day 0 0 GTM New WSDS | Wsds/wsds.a90.W: Initial WSDS
#2 1 Day 0 0 Nodes New WSDS | Wsds/wsds.a90.W: Initial WSDS
Accuracy
0 CEOC X acc: 0 (rad 0, wtr 0, infr 0, cas 0)
0 CEOC Y acc: 0 (rad 0, wtr 0, infr 0, cws 0)
0 CEOC Z acc: 0 (rad 0, wtr 0, infr 0, cas 0)
0 SEOC R acc: 0 (rad 0, wtr 0, infr 0, cas 0)
0 SEOC S acc: 0 (rad 0, wtr 0, infr 0, cas 0)
0 STARC R acc: 0 (rad 0, wtr 0, infr 0, cas 0, wil 0)
0 STARC S acc: 0 (rad 0, wtr 0, infr 0, cas 0, ail 0)
0 FRC - acc: 0 (rad 0, wtr 0, infr 0, cws 0, wil 0)
Period 0
County X rad 0, wtr 2, infra 1, cas 830, wil 3
County Y rad 0, wtr 0, infra 0, cas 20, wil 100
County Z rad 0, wtr 0, infra 0, ces 0, ail 100
0 Police sends Source-Report to CEOC Y (rad 0, wtr -1, infra 0, cas 20, wil -1) at 1
0 Police sends Source-Report to CEOC Y (rad 0, wtr -1, infra 0, cas 10, ail -1) at 1
0 Hospita sends Source-Report to CEOC Y (rad 1, wtr 0, infra -1, cas 20, ail -1) at 1
0 Mil-bas sends Source-Report to STARC R (rad 0, wtr 1, infra 0, cas 839, ail 2) at 1
1 CEOC Y sends Initial-Disaster-R to SEOC S (rad 0, wtr -1, infra 0, cas 20, ail -1) at 5
1 Hospita sends Source-Report to CEOC X (rad 1, wtr 2, infra -1, cas 830, ail -1) at 2
1 Fire-da sends Source-Report to CEOC X (rad 2, wtr -1, infra 0, cas -1, ail -1) at 2
1 STARC R sends Suwary-Army-Disas to MNH (rad 0, wtr 1, infra 0, cas 839, ail 2) at 5
2 CEOC X sends Initial-Disaster-R to SEOC R (rad 1, wtr 2, infra -1, cas 830, ail -1) at 6
3 Mil-bas sends Source-Report to STARC R (rad 2, wtr 0, infra 0, cas 0, ail 101) at 4
4 Police sends Source-Report to CEOC X (rad 1, wtr -1, infra 0, cas 877, ail -1) at 5
4 Police sends Source-Report to CEOC X (rad 0, wtr -1, infra 1, cas 873, ail -1) at 5
4 STARC R sends Suwary-Army-Disas to MNH (rad 2, wtr 1, infra 0, cas 839, ail 101) at 9
5 FFE-NRC sends Source-Report to FRC R (rad 0, wtr -1, infra -1, cas 705, ail -1) at 6

Chart 17

The node labeled "Force" generates the simulation time step. The system monitor selects the node to play at any given time. Initially, force initiates a time step, followed by an update of ground truth by the Ground Truth Monitor (GTM). The nodes are activated next with the source nodes acting first to detect the latest update of ground truth. The procedures rule sets and the operational relationships now determine which of the nodes will activate next. The simulation may be interrupted at any of the nodes by selecting that node. Once selected, the simulation stops the next time the node is activated. At this point, a number of things can be viewed. Examples are presented in Chart 18.

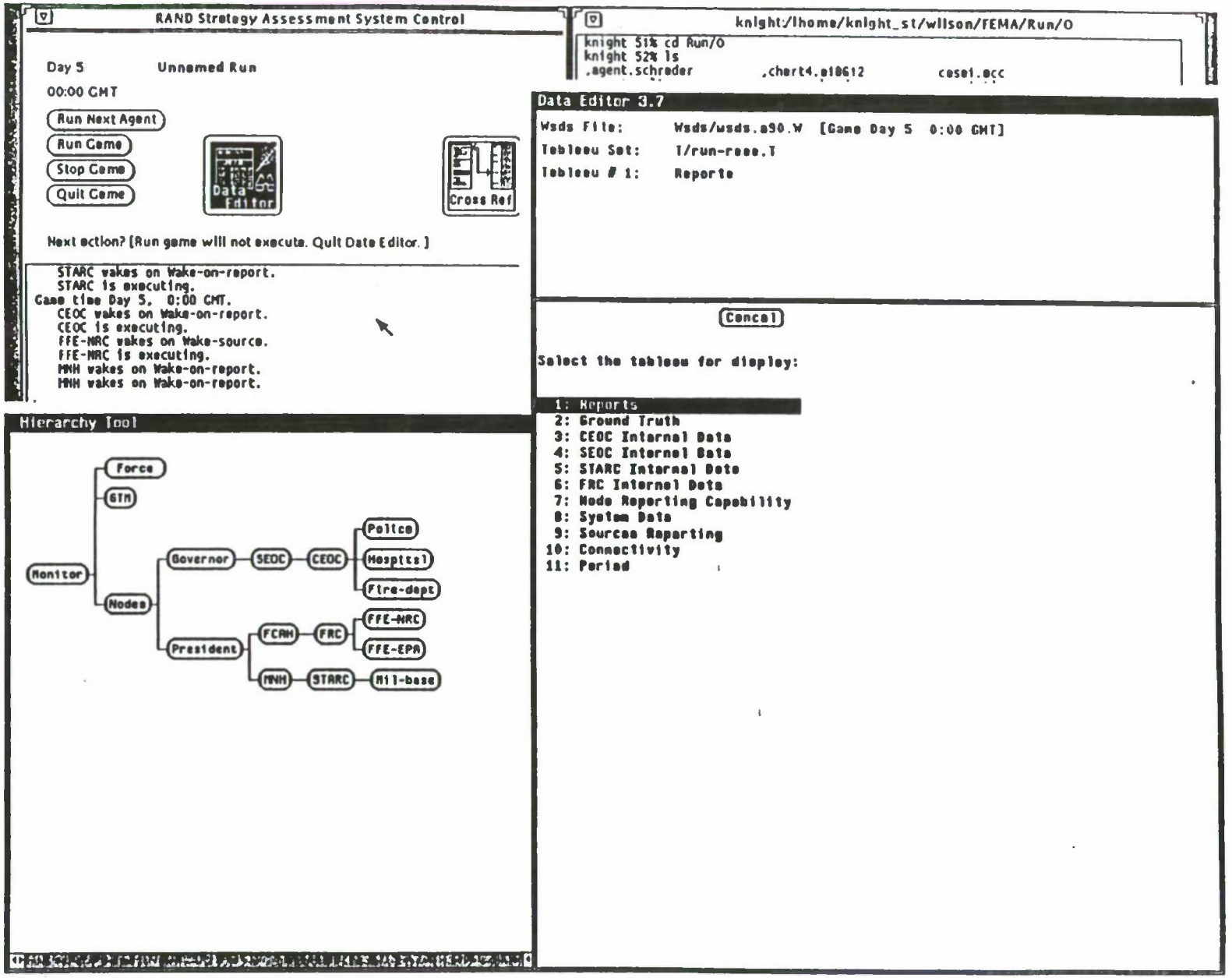

RAND Strategy Assessment System Control
Day 5
Unnamed Run
00:00 GMT
Run Next Agent
Run Game
Stop Game
Quit Game
Data Editor
Cross Ref
Next action? [Run game will not execute. Quit Data Editor.]
STARC wakes on Wake-on-report.
STARC is executing.
Game time Day 5, 0:00 GMT.
CEOC wakes on Wake-on-report.
CEOC is executing.
FFE-NRC wakes on Wake-source.
FFE-NRC is executing.
MNH wakes on Wake-on-report.
MNH wakes on Wake-on-report.
knight:/lhome/knight_st/wilson/FEMA/Run/0
knight 51% cd Run/0
knight 52% ls
.agent.schrader
.chart4.p18612
case1.ecc
Data Editor 3.7
Wsds File: Wsds/wsds.a90.W [Game Day 5 0:00 GMT]
Tableau Set: T/run-rese.T
Tableau # 1: Reports
Cancel
Select the tableau for display:
1: Reports
2: Ground Truth
3: CEOC Internal Data
4: SEOC Internal Data
5: STARC Internal Data
6: FRC Internal Data
7: Node Reporting Capability
8: System Data
9: Sources Reporting
10: Connectivity
11: Period
Hierarchy Tool
Force
GTN
Monitor
Governor
SEOC
CEOC
Police
Hospital
Fire-dept
Nodes
President
FCAH
FRC
FFE-NRC
FFE-EPA
MNH
STARC
Mil-base

Chart 18

The window at the right in Chart 17, logs selected details of the run. The analyst is able to select the detail required by modifying the code during the simulation or before the simulation has begun. The example in Chart 19 depicts some of the information requested at time periods 0 through 4. The first set of data depicts the accuracy with which each of the facilities "knows" ground truth. This is assumed to be a reading before any detections are made. Therefore, the accuracy measured as the "distance"[1] from ground truth is 0. That is, perfect knowledge is assumed. The damage currently existing within the counties and states is also initialized to 0.

[1] More formally, accuracy in knowledge about i at time t is defined to be $A_i = d(o_i g_i)$ where d is a distance metric, o_i is the sensor's perceptionof the true value of i and gi is the ground truth value.

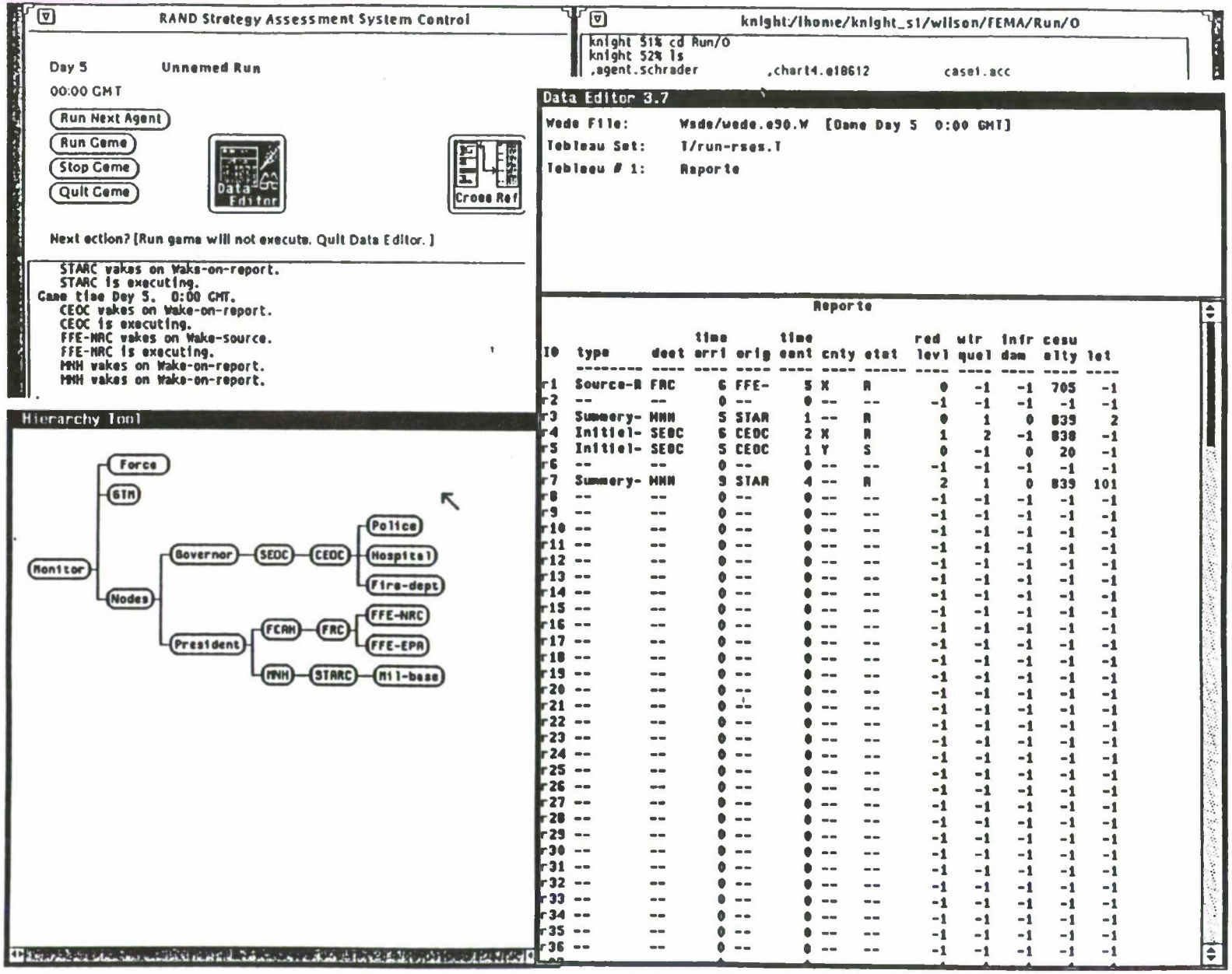

RAND Strategy Assessment System Control
Day 5 Unnamed Run
00:00 GMT
Run Next Agent
Run Game
Stop Game
Quit Game
Data Editor
Cross Ref
Next action? (Run game will not execute. Quit Data Editor.)
STARC wakes on Wake-on-report.
STARC is executing.
Game time Day 5. 0:00 GMT.
CEOC wakes on Wake-on-report.
CEOC is executing.
FFE-NRC wakes on Wake-source.
FFE-NRC is executing.
MNH wakes on Wake-on-report.
MNH wakes on Wake-on-report.
knight:/home/knight_s1/wilson/FEMA/Run/0
knight 51% cd Run/0
knight 52% ls
.agent.schrader .chart4.e18612 case1.acc
Data Editor 3.7
Wede File: Wede/wede.e90.W [Game Day 5 0:00 GMT]
Tableau Set: T/run-rses.T
Tableau # 1: Reporte
Hierarchy Tool
Force
GTN
Monitor
Governor
SEOC
CEOC
Police
Hospital
Fire-dept
Nodes
President
FCAH
FRC
FFE-NRC
FFE-EPA
MNH
STARC
Mil-base
Reporte
ID type dest time arri orig time sent cnty stat red levl wtr qual infr dam casu alty let
r1 Source-R FRC 6 FFE- 5 X R 0 -1 -1 705 -1
r2 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r3 Summary- MNH 5 STAR 1 -- R 0 1 0 839 2
r4 Initiel- SEOC 6 CEOC 2 X R 1 2 -1 838 -1
r5 Initiel- SEOC 5 CEOC 1 Y S 0 -1 0 20 -1
r6 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r7 Summary- MNH 9 STAR 4 -- R 2 1 0 839 101
r8 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r9 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r10 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r11 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r12 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r13 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r14 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r15 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r16 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r17 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r18 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r19 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r20 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r21 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r22 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r23 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r24 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r25 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r26 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r27 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r28 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r29 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r30 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r31 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r32 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r33 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r34 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r35 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1
r36 -- -- 0 -- 0 -- -- -1 -1 -1 -1 -1

Chart 19

The second set of data is the ground truth update at time 0. Note that County X has experienced level 2 damage (expressed as a percentage of total resources available) to water resources and level 1 to infrastructure. In addition, there were 830 casualties and only 3% of the military facilities are still operational. Based on ground truth, we should conclude that maximum severity and extent of damage occurred in County X.

The next set of log statements consist of a chronological sequence of report activities. For example, at time 0, the police in County Y forward a report to the County EOC that the radiation level is 0 and that an estimate of 20 casualties. The -1 entries means that no information on these items was rendered in the report. Finally, the report is scheduled to arrive at time period 1.

Charts 18 and 19 illustrate the interrupt process. In Chart 18, the user selects the *Data Editor* button in the control panel and the options list on the right appears. Each option presented to the user, records the current status of certain elements of the system. Within each of these, modifications, additions and deletions may be made to most of the data. In this case, we have selected the reports status which is displayed in Chart 19. At time 0, 7 reports (r1 through r7) have been generated. A -1 entry simply means that no information was provided on this data element.

Batch Mode

A total of 100 runs of the simulation were made and data on accuracy were accumulated at each of the nodes. The summary histograms in Charts 20 and 21 record the frequency with which certain accuracy levels were achieved. In each case, the vertical axis is the frequency of occurrence and the horizontal axis represents accuracy. Accuracy is measured in terms of relative distance from ground truth. Consequently, the accuracy of the reports generated by the County EOCs (Chart 20) in this operational concept is not very good. Less than 5% of the observations were completely accurate and most (47%) were highly inaccurate (70% inaccuracy). For the reports generated by the National Military Command Center, there were no completely accurate reports, but most were only 7% inaccurate (Chart 21).

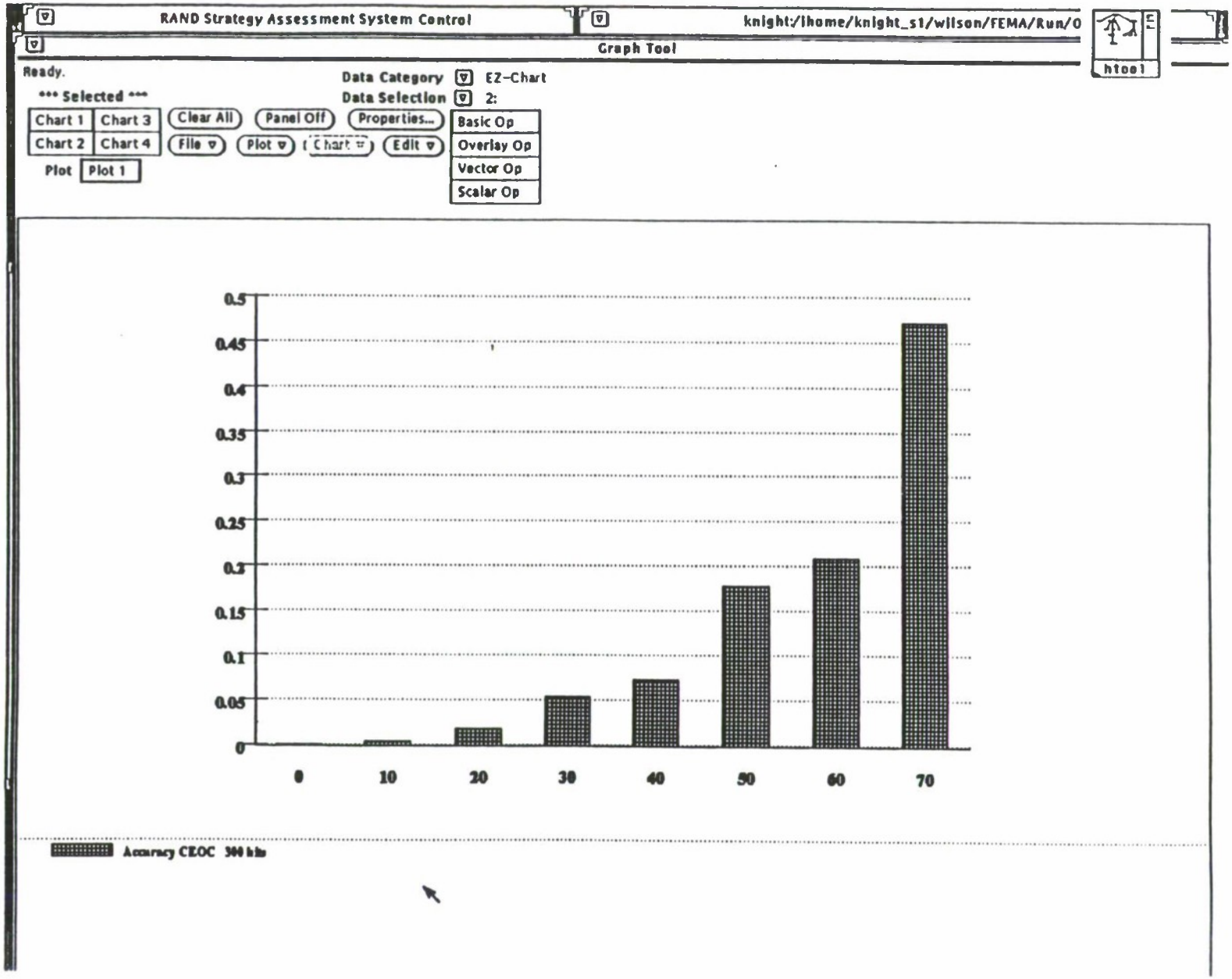

RAND Strategy Assessment System Control
knight:/home/knight_s1/wilson/FEMA/Run/0
Graph Tool
htool
Ready.
*** Selected ***
Chart 1
Chart 3
Chart 2
Chart 4
Clear All
Panel Off
Properties...
File
Plot
Chart
Edit
Plot
Plot 1
Data Category
EZ-Chart
Data Selection
2:
Basic Op
Overlay Op
Vector Op
Scalar Op
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
10
20
30
40
50
60
70
Accuracy CEOC 300 hits

Chart 20

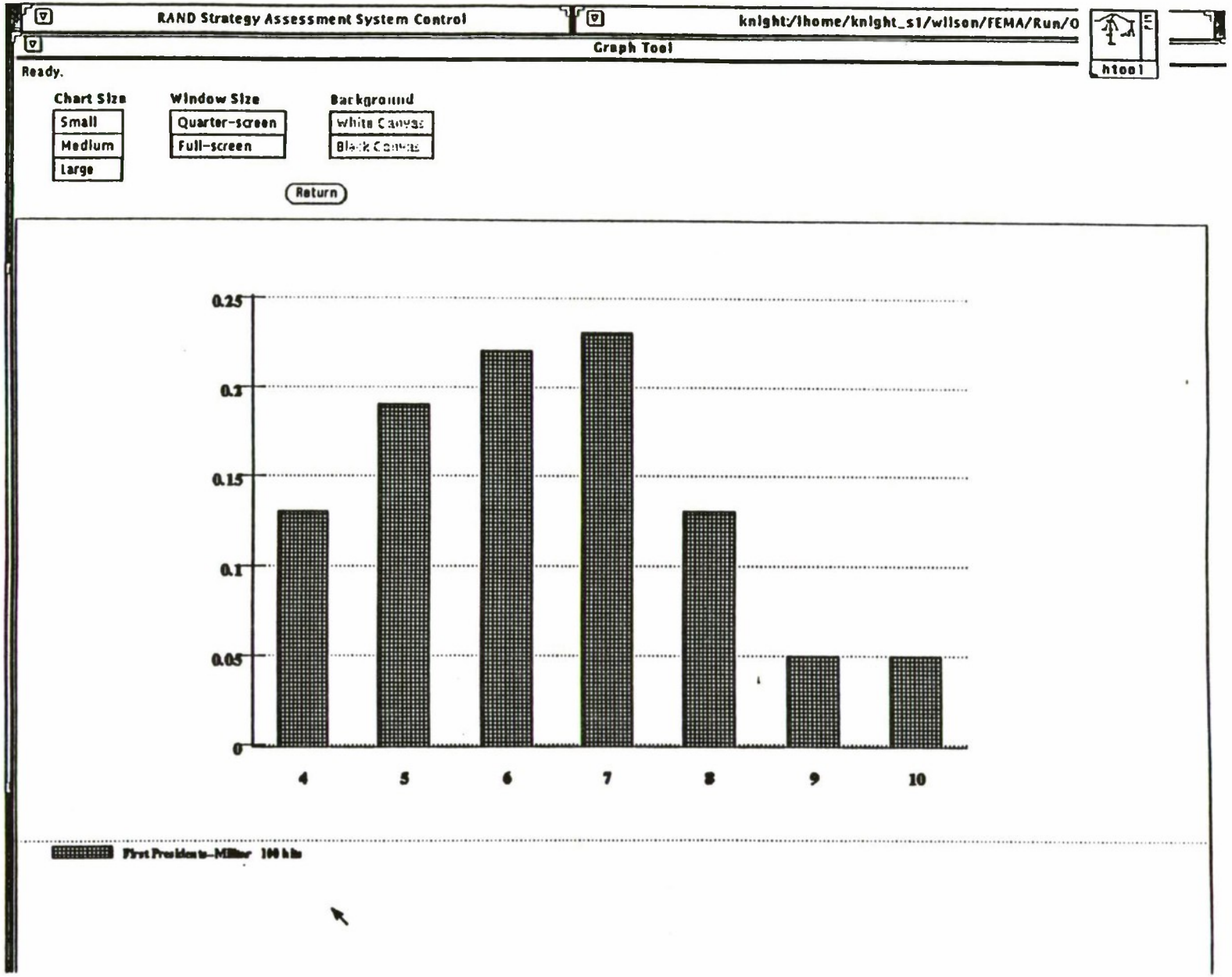

RAND Strategy Assessment System Control
knight:/home/knight_s1/wilson/FEMA/Run/0
Graph Tool
htool
Ready.
Chart Size
Small
Medium
Large
Window Size
Quarter-screen
Full-screen
Background
White Canvas
Return
0.25
0.2
0.15
0.1
0.05
0
4
5
6
7
8
9
10

Chart 21

CONCLUSION

The notion of using a testbed to develop C^2 models allows the analyst to focus on those aspects of command and control which are generally neglected in existing models, namely the C^2 system and the operational concept. The testbed also allows us to generate alternative operating concepts and systems whose performance can be compared. Typically, the analyst begins with a rather well defined base case, and proceeds to alter it in meaningful ways. In this way an optimum is approached through an iterative process. Central to the analysis is the ability to represent processes at variable levels of resolution.

Ideally, the physical aspects of the C^2 system should be modeled by the external simulation (the particular disaster being monitored). This leaves the processing rules, information flow and uncertainty models to be altered through the testbed. This limits the variability of the C^2 model and also allows for the command and control system to be effected by the same dynamics as other elements in the environment. This latter effect is something which is generally lacking in current C^2 models.

Finally, the application of this methodology moves the debate about C^2 from a question of connectivity to one of decision and the exercise of control. In addition, it provides the research community with a mechanism to test new fusion and classification methods to support command decisions and actions.