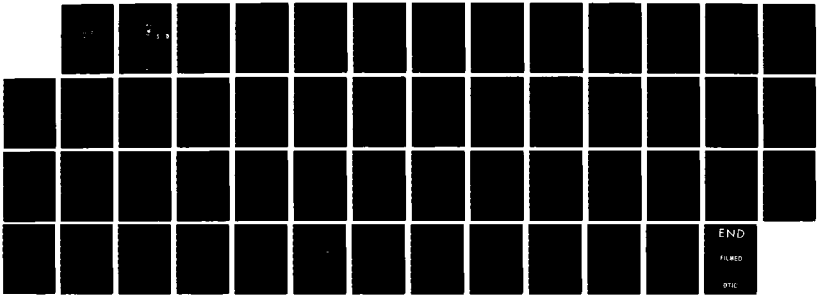
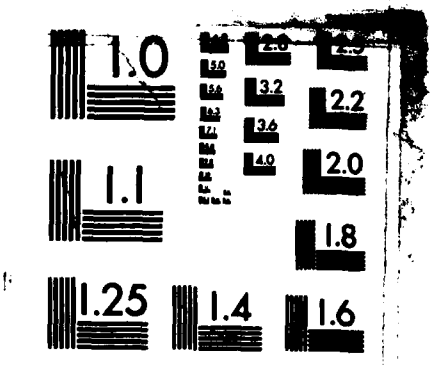
AD-A157 331 ON DISTRIBUTED WARGAMING IN OPERATIONAL C2 SYSTEMS USING OBJECT-ORIENTED PROGRAMMING LANGUAGES(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA R K HUBER ET AL. 1/1

UNCLASSIFIED JUN 85 NPS-74-85-001 F/G 9/2 NL





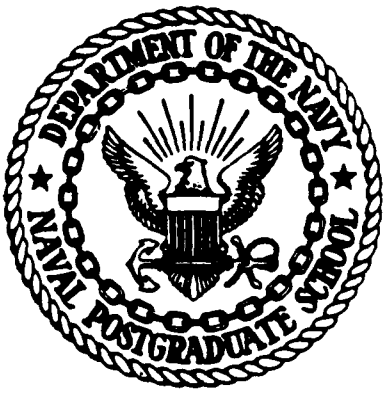

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

NPS-74-85-001

# NAVAL POSTGRADUATE SCHOOL

## Monterey, California








ON DISTRIBUTED WARGAMING
IN OPERATIONAL $C^2$-SYSTEMS USING
OBJECT-ORIENTED PROGRAMMING LANGUAGES

Reiner K. Huber

John M. Wozencraft

June 1985



Approved for public release; distribution unlimited.

Prepared for:
Defense Advanced Research Projects Agency
1400 Wilson Blvd
Arlington, Virginia 22209

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral R. H. Shumaker
Superintendent

D. A. Schrady
Provost

The research work reported herein was funded by Defense Advanced Research Projects Agency, 1400 Wilson Boulevard, Arlington, Virginia 22209, Project No. N0001484WR24187 for the Chief of Naval Research, Code 512, Arlington, VA 22217.



Prepared by:

Reiner K. Huber
Federal Armed Forces University Munich
D-8014 Neubiberg, West Germany

John M. Wozencraft
Professor

Reviewed by:

Michael G. Sovereign
Chairman, Command, Control
and Communications
Academic Group

Released by:

D. A. Schrady
Provost

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

## REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

| | |
|---|---|
| 1. REPORT NUMBER: NPS-74-85-001 | 2. GOVT ACCESSION NO.: AD-A157331 |
| 3. RECIPIENT'S CATALOG NUMBER: | |
| 4. TITLE (and Subtitle): On Distributed Wargaming in Operational $C^2$-Systems Using Object-Oriented Programming Languages | 5. TYPE OF REPORT & PERIOD COVERED: |
| | 6. PERFORMING ORG. REPORT NUMBER: |
| 7. AUTHOR(s): Reiner K. Huber<br>John M. Wozencraft | 8. CONTRACT OR GRANT NUMBER(s): |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School<br>Monterey, California 93943-5100 | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: N0001484WR24187 |
| 11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Advanced Research Projects Agency<br>1400 Wilson Boulevard<br>Arlington, Virginia 22209 | 12. REPORT DATE: June 1985<br>13. NUMBER OF PAGES: 44 |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): Chief of Naval Research<br>Code 512<br>Arlington, VA 22217 | 15. SECURITY CLASS. (of this report): Unclassified<br>15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Distributed War Gaming (DWG) is the interactive simulation of military systems employing their command and control ($C^2$) - systems and the data processing capabilities of their headquarters. Using land-warfare as an example, a conceptual framework for the development of DWG-systems is provided which is based on the utilization of object-oriented software systems. In contrast to traditional software, these systems invoke the transformation of data by messages between objects which specify the

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

transformation to be performed by the addressed object. When defining the system to be modelled, the analyst needs to be concerned with the reaction of individual objects to local messages only. The dynamics of complex systems are the results of the propagation of local messages to its individual elements.

Based on a brief description of the basic principles behind object-oriented languages, a classification of objects and their attributes in a land-wargame is discussed and a definition of object behaviors is illustrated. Eight basic types of messages for a DWG and their formats are proposed, as is an algorithm for the synchronization of distributed simulations in DWG.

# ON DISTRIBUTED WARGAMING IN OPERATIONAL $C^2$-SYSTEMS USING OBJECT-ORIENTED PROGRAMMING LANGUAGES

Reiner K. Huber
Federal Armed Forces University Munich
D-8014 Neubiberg, West Germany

John M. Wozencraft
Naval Postgraduate School
Monterey, California 93943

## ABSTRACT

Distributed War Gaming (DWG) is the interactive simulation of military defense systems employing their command and control ($C^2$) - systems and the data processing capabilities of their headquarters. Using land-warfare as an example, a conceptual framework for the development of DWG-information systems is provided which is based on the utilization of object-oriented software systems. In contrast to traditional software, these systems invoke the transformation of data by messages between objects which specify the transformation to be performed by the addressed object. When defining the system to be modelled, the analyst needs to be concerned with the reaction of individual objects to local messages only. The dynamics of complex systems are the results of the propagation of local messages to its individual elements.

Based on a brief descripton of the basic principles behind object-oriented languages, a classification of objects and their attributes in a land-wargame is discussed and a definition of object behaviors is illustrated. Eight basic types of messages for a DWG and their formats are proposed, as is an algorithm for the synchronizaton of distributed simulations in DWG.

Object-oriented software systems are judged to have potential for facilitating evolution in military software development. Their use in DWG could, in the long run, alleviate the problem of model acceptability. Thus, the development of prototype DWG-systems is proposed for feasibility testing and for the development of a strategy for the evolution of operational DWG-systems.

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT.)

## 1. DISTRIBUTED WAR GAMING (DWG)

We define DWG as the interactive simulation of military defense systems or parts thereof employing their command and control ($C^2$)-systems and the data processing capabilities of their headquarters. Thus, a DWG-system consists of two principal subsystems: (1) the $C^2$-system and (2) the war gaming system.

The following discussions are concerned with the War Gaming System, in particular with the development of a conceptual framework for its evolution, assuming that an advanced $C^2$-system exists that provides sufficient data processing capabilities at each of its headquarters.[1)]

## 2. BASIC DEVELOPMENT REQUIREMENTS

In essence, a War Gaming System may be considered an information system for the dynamic representation of combat within the given $C^2$-structure. Thus, it comprises a software system which describes how the defense system and its parts react when given a mission, subject to tactics/doctrine, resource constraints and environmental conditions. Its design must be such that it is

a. easy to handle, in the sense that it does not require specific skills for its use beyond those necessary for the command and control of the real system;
b. not affected by any changes in the $C^2$-system hardware configuration;
c. capable of easy adaptation with regard to the introduction of new weapon systems and tactics, a change of operational principles and organizational structures, and the execution of new missions/tasks;
d. sufficiently realistic so that, in addition to providing a dynamic environment for training in staff functions and procedures, it offers a means to evaluate (on a relative scale) tactical and operational decision alternatives.

In order to provide for evolutionary development and flexible use, the software system must

e. consist of self-sufficient components, i.e., no component of the system may depend on the internal design details of any other component (modular design);
f. be based on a development and maintenance concept that permits one to test, implement and modify software modules without interrupting the

---

1) In the context of this paper a headquarters (HQ) is defined as an element in the military organization exercising command and control.

routine operation of the $C^2$-system and ongoing war gaming activities;

g. be developed "inward-out" so as to permit gradually enriching an initially austere but validated[2] operational gaming software. E.g., one may start the development with implementing models of the fundamental combat processes (attrition, suppression and movement of front line forces) first and subsequently proceed with combat support, combat service support and rear area functions;

h. provide for automatic documentation of the system states, modifications and extensions in order to facilitate effective configuration control and management capable of eliciting and then providing the user the system configuration he desires.

## 3. OBJECT-ORIENTED PROGRAMMING - THE SOFTWARE TOOL FOR DWG

The $C^2$-system as defined above may be considered as a hierarchy of HQ-computers linked by communication channels, as illustrated in Fig. 1 for a land force organization within a theater.

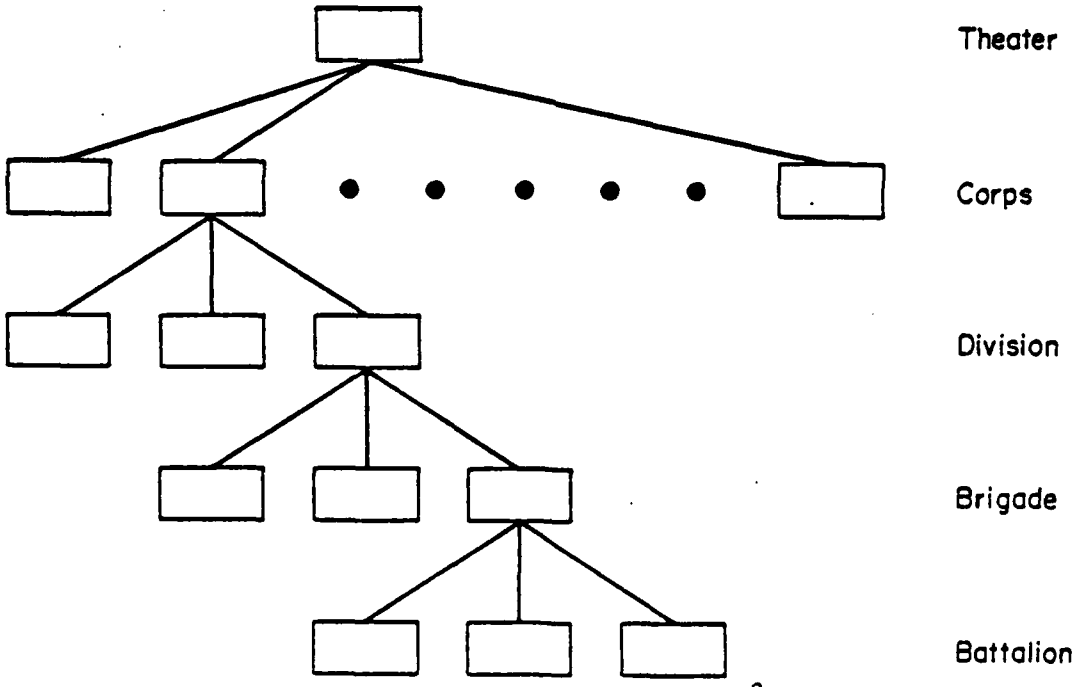



Fig. 1. Computer Hierarchy in a Field Army $C^2$-System.

[2] Validation does not imply that the transformations of the software modules are necessarily true in an empirical sense. Rather, it implies that the model of the respective combat process is internally consistent, i.e., derived by applying the formal rules of logic to the assumptions underlying the model and accepted by the user. (See Bergman [1]).

Such a system may be used for war gaming if the computers, at all levels, are given the capability to simulate the actions and behaviors of the respective unit and, in certain applications, their commanders consistent with the prevailing environment and the mission assigned or the orders given by the next higher level commander. In this case, which we shall assume throughout this paper, battle simulation may be considered to be essentially a result of interactions between the components of a hierarchical system where the components are the military commanders (war gamers) and the computer software. Thus, the interactions in the simulation represent, much as in real life combat, mesages that are passed between components.

This exactly is the perception underlying object-oriented programing languages such as SMALLTALK (Ingalls [2]) and ROSS (McArthur and Klahr [3]). In addition, object-oriented programing permits satisfying most of the basic design requirements for DWG discussed above. For example, the simulation objects (components) are defined so that they are self-sufficient. They and their behaviors are easily modified since one needs to be concerned only with how the objects react to local messages. Nevertheless, in the simulation the more distant effects of local behavior are revealed because "... each local message transmitted can trigger others, and these in turn can trigger still others ... (thereby modelling) the arbitrary propagation effects that characterize complex systems." ([3], p. 1).

### 3.1 Basic Principles Behind Object-Oriented Software Systems

Traditional software systems distinguish between data and procedures to transform those data. In contrast, object-oriented software systems have only one type of entity, the object, representing both data and procedures. The transformation of data is invoked by messages which specify the respective transformation to be performed (see, e.g., Robson [4]).

Thus, an object (e.g., a tank) is defined by its set of attributes (variables) and the set of transformations (methods) describing its behavior[3] (e.g., for a tank it might include activities like advance, observe, fire, retreat, etc.). In addition to the addressee (i.e., object to be manipulated, called the receiver), the message includes a symbolic name, the

[3] The transformation or method is a formal description of the sequence of actions to be performed by the processor.

selector, which indicates what the programmer wants to happen (e.g., move), i.e., which one of the addressee's transformations is to be performed. The message may also contain a parameter that specifies a location or a point (e.g., a tank may be ordered to move to a given point). The set of messages an object may respond to is called a message protocol or message template.

Most object-oriented systems distinguish between the description of an object and the object itself. The description of an object contains (1) the methods which describe the behavior and (2) the set of parameters which represent variables characteristic for a class as a whole. A specific object is called an instance of its class. Variables that are shared by all instances of a class are part of the class and called class variables. The other variables (the values of which differ for different instances of a class) are called instance variables. Thus, all instances of a class have the same number of instance variables. Also, all instances of a class use the same method to respond to a particular type of message (e.g., the movement-algorithm in case a tank is ordered to move). The difference in the response by two different instances is the result of their different instance variables (attributes).

Classes of objects may themselves be considered as objects which create new instance-objects when so instructed by an appropriate message. This feature makes an object-oriented language especially attractive to a user who may, in a war game, want to test quickly the pay-off from improved performance parameters and/or alternative tactics.

Object-oriented systems are further distinguished by a feature called inheritance, implying a parent-child relationship between objects or classes of objects. A child (subclass) inherits everything (variables and methods) from the parent (superclass). Thus, a class may be modified to create another class. The child-object, respectively subclass, may be considered to be simply an extension or a more detailed description of the parent-object, respectively superclass.[4] This feature appears to make object-oriented programming a natural technique for the development of combat model-hierarchies. In

[4] There are four different types of extensions: (1) addition of instance variables; (2) redefinition of methods (behaviors) of the superclass; (3) inclusion of methods for new messages (not understood by the superclass); (4) addition of new class variables (see [4], p. 86).

TABLE 1

Gaming Configurations

| Purpose of Gaming Exercise | Representation of Subord. $C^2$-Nodes | | | Representation of Subord. Gaming Level | | |
|---|---|---|---|---|---|---|
| | H | S | HS | E | A | EA |
| TRAINING<br>- Tac/Ops Procedures | | * | | * | | |
| - Tac/Ops Plans | * | | | * | | |
| TACTICS/OPERATIONS<br>- Tac Development | | | * | * | | |
| - Ops Concepts Tests | | | * | | | * |
| PLANNING SUPPORT<br>- Weapon Sys Plng | | * | | * | | |
| - Force Structure Plng | | * | | | * | |
| - Exercise Plng | | * | | | | * |

H = Human Decision-Maker (HDM)

S = Formal Surrogate of HDM

HS = HDM only for "critical" objects (e.g., objects situated within area of enemy main thrust)

E = Explicit representation of objects and their interactions

A = Aggregated representation of objects and their interactions

EA = E only for test/exercise objects

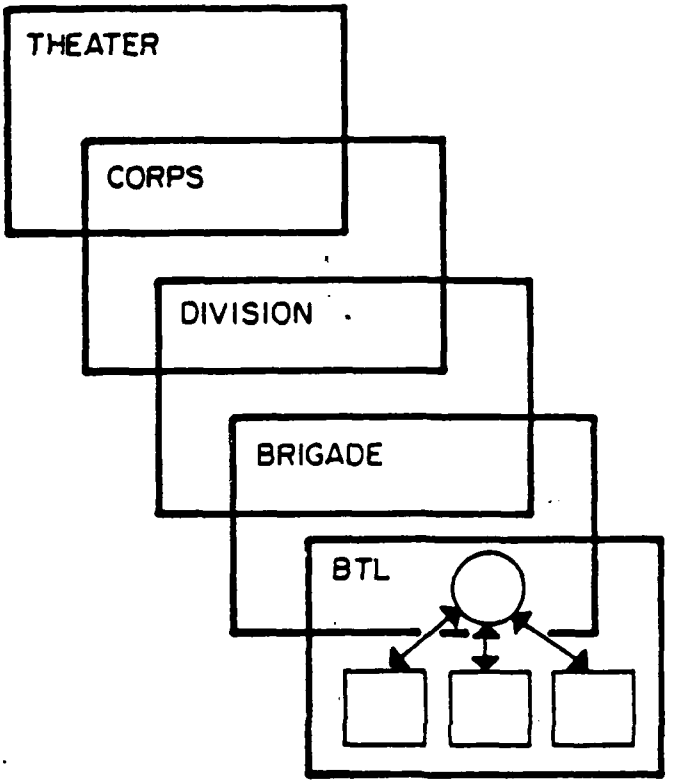



Figure 8. HIERARCHY OF MODELS.

Which game configuration one may want to implement will depend on the purpose of the gaming exercise and on the available $C^2$-resources, i.e., to what extent the existing $C^2$-system (including its human decision resources) may become physically involved. Table 1 provides a summary view on game configurations as they appear to be required by a number of gaming purposes.

4.2 Attributes of Basic Objects/Actors

In the context of this paper, basic objects or actors correspond to real world objects or actors within the four main functional categories defined

The $C^2$-node hierarchy is identical to the computer hierarchy depicted in Fig. 1, i.e., each $C^2$-node has access to an appropriate computer. In a war gaming exercise, all of the item-nodes are simulated on an appropriate computer of the hierarchy. The simulation of $C^2$-nodes (i.e., through replacing the respective commander and his staff by decision logic) might be optional.

$C^2$-nodes would most naturally be implemented on their "own" computer, as would the subordinate item-nodes if they represent maneuver units. In case they represent support units they may, depending upon the gaming configuration, also be simulated on the computer of the $C^2$-node controlling the supported unit.

In case a $C^2$-node is being automated, one might consider aggregating that $C^2$-node and its subordinate item-nodes into one aggregated item-node. If that is done, starting at battalion level up through the hierarchy, one obtains a hierarchy of partially overlapping models (see Fig. 8), i.e., at each level, the resolution of the model is identical to the aggregation of the next lower level model.

A game configuration in which all $C^2$-nodes are automated, but none are aggregated, would correspond to what has become called a "Nested Model" (see Huber [5]).

With a computer configuration in the $C^2$-system as postulated above, one could implement any game configuration between a fully automated hierarchy (where combat on each level would be simulated without human intervention in a stand-alone mode) and a fully interactive nested model (where the commanders and their staffs participate in the simulation of the respective level and all levels below). This includes versions which may, at one given level, retain some interactive $C^2$-nodes with the rest being automated (i.e., modelled by decision logic) and/or aggregated together with their subordinate item nodes. Fully and partially automated configurations would permit different degrees of aggregation.[8] Also, one may retain the interactive $C^2$-nodes at a high, say division, level controlling an aggregated division model. Such an aggregated interactive configuration would permit games that involve only one $C^2$-level.

---

[8] Such configurations would essentially resemble versions of variable-resolution combat model architectures proposed by Parry (see [7]).

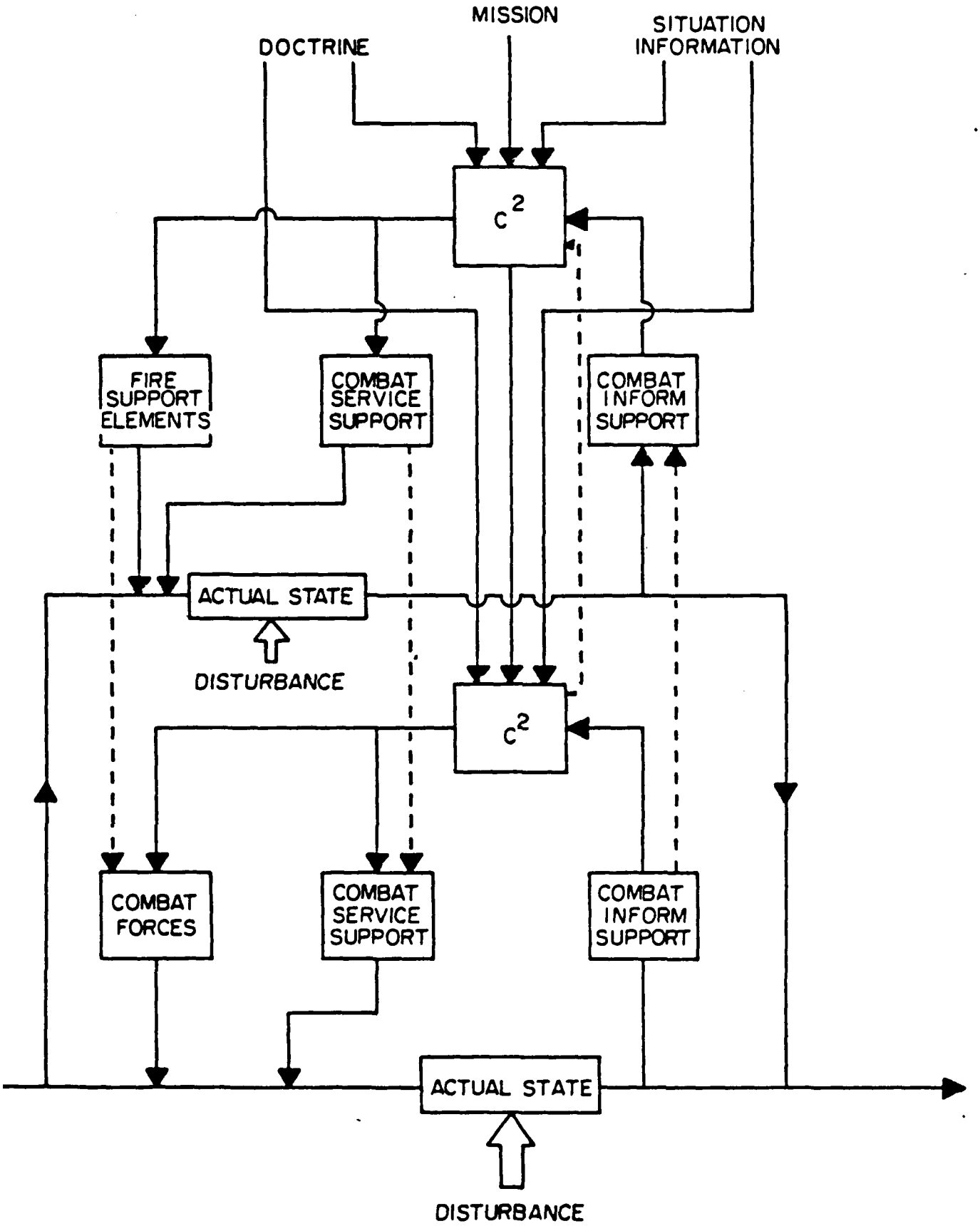



Figure 6. BASIC HIERARCHICAL INTERACTIONS IN A MILITARY SYSTEM

higher levels of $C^2$, the subordinate organic unit may be a commander and his staff controlling their organic units, which again may be a subordinate commander and his staff until the level is reached at which combat takes place physically, i.e., where the opposing maneuver forces interact. Thus, if we interpret the "disturbance from enemy actions" as a result of maneuver force interaction (rather than a result of, e.g., attrition or suppression effects caused by interdiction), Fig. 3 may be considered to represent the lowest relevant $C^2$-level (e.g., Btl). With regard to maneuver forces, the next higher $C^2$-level should indeed control the Btl HQ rather than the maneuver companies. This leads us to extend the functional hierarchy by one additional level, as depicted in Fig. 6.

At each level, the $C^2$-authority responds to state changes in its environment (as perceived from the information provided by the combat information support and the intelligence and surveillance systems) in such a way as it deems appropriate to accomplish the assigned mission.

The mission statement can be considered to be an element of contingency plans and/or operational plans of the superior level $C^2$-authority. It is, among others, an expression of the current operational concepts and includes in addition to the objective to be accomplished, information on the available inorganic combat and support resources, on issues related to boundaries and coordination with neighbors, and on certain restrictions that may have to be observed.

Military doctrine and tactical principles include a further set of rules or restrictions controlling the response of the $C^2$-authority and the behavior of its subordinate actors.

#### 4.1.3 Hierarchical Aspects of Game Design

Generalizing the structural relations depicted in Fig. 3 and 6, as they are relevant to a field army, we arrive at a tree with two types of nodes, $C^2$-nodes and item-nodes. A $C^2$-node symbolizes a commander and his staff, an item-node represents generic military units and installations. Fig. 7 shows such a tree, which corresponds to the organizational hierarchy of a field army, from theater down to battalion-level. Only at battalion-level do the item nodes represent maneuver units. At the higher levels, item-nodes represent support units which may, upon request by one of the lower level $C^2$-nodes, be allocated to any one of the $C^2$ and/or item-nodes subordinate to the requesting $C^2$-node.

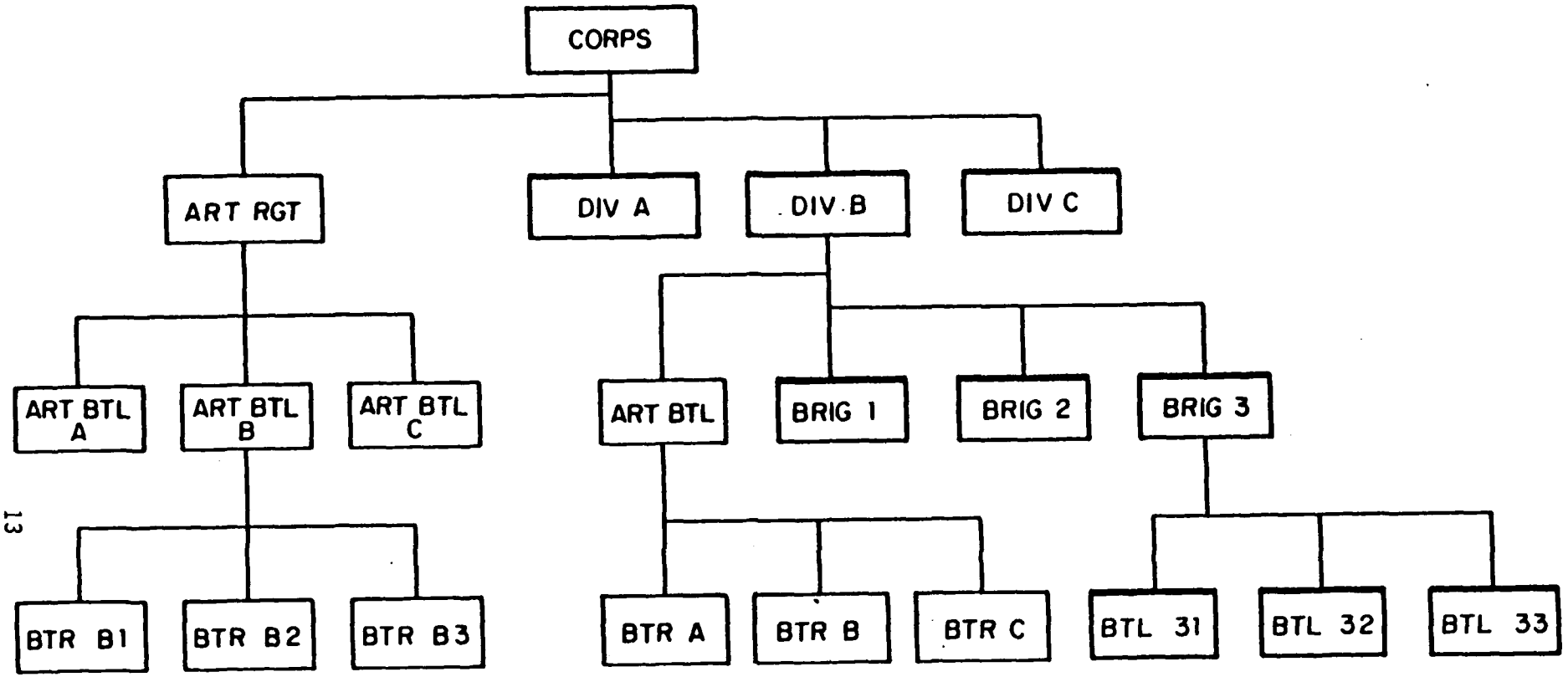



Figure 5. HIERARCHIES OF MANEUVER AND FIRE SUPPORT UNIT WITHIN A CORPS.

MANEUVER UNIT

FIRE SUPPORT UNIT

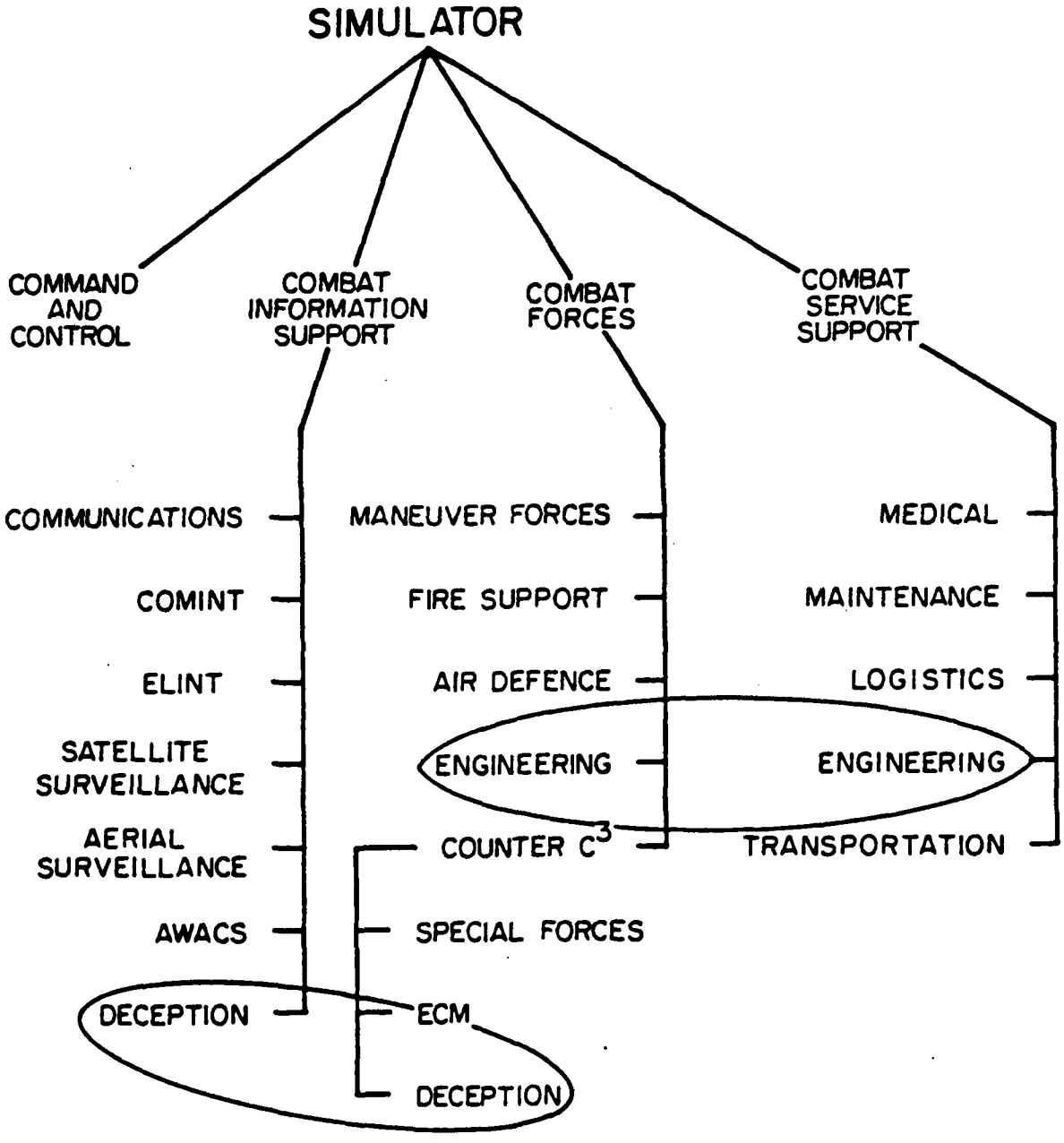



Figure 4. PRINCIPAL SUB-HIERARCHIES IN A LAND WARFARE SIMULATION.

- COMINT
- ELINT
- Satellite Surveillance
- Aerial Recce
- AWACS
- Deception

Some of the functions (see Fig. 4) are present in more than one main category (e.g., Engineering and Deception). If the corresponding objects exhibit similar attributes and behaviors in all categories, their organizational allocation within the object hierarchy should be such as to take advantage of ROSS's inheritance hierarchy (see [6], pp. 7-8).

Whether or not all sub-hierarchies will have to be modelled explicitly depends on how the operational or tactical control over their respective units is organized and/or practiced. However, be it as it may, a sub-hierarchy always represents an organic entity at some (usually higher) level of $C^2$-authority. This is illustrated in Fig. 5 for fire support and maneuver hierarchies as they may exist within a Corps. It shows that one functional area (in this case, Fire Support) may comprise more than one sub-hierarchy, at least when considering the organizational lines of command and control.

Taking into account the informal communication channels in an organization, it may well be that the behavior of the artillery units in either of the two hierarchies might depend, perhaps significantly, on the messages passed through such channels. This would justify eventually amalgamating the two sub-hierarchies into one. However, in order to do so one must know not only what informal communication takes place, but also how it affects behavior. Since both may only be found out experimentally, it is proposed that an initial version of a distributed gaming system be designed considering only the formal lines of communication and then applied to find out, among other questions, what informal communication takes place and how it affects the response of actors.

#### 4.1.2 Command and Control

In accordance with Fig. 3, command and control (as we understand it) does, at each level, represent the quintessential actor as defined in ROSS. In reality, $C^2$ corresponds to a commander and his staff who plan, direct and supervise the operations of their subordinate organic units. At the

structure must, for the purpose of object-oriented programming, include generic elements that can create temporary instance objects when ordered to do do.

Inorganic forces or systems are those which the particular level $C^2$-authority may (1) be awarded temporarily by a higher level $C^2$-authority upon request or (2) have access to continuously without a special request. The former comprise combat support and combat service support elements (e.g., Corps artillery fire support for a maneuver battalion, Division-level transportation support to provide consumables (ammo, Pol, food) to maneuver companies), and some combat information support (e.g., target information provided to a divisional artillery regiment by air recce sorties authorized through a TOC). The latter includes mainly the battlefield surveillance and intelligence organization and its assets.

In the context of land operations we may consider a military system to consist of four fundamental hierarchies:

- Command and Control ($C^2$)
- Combat Forces
- Combat Service Support
- Combat Information Support

Within the three classes of physical entities we may distinguish functional subhierarchies such as, for the

Combat Forces
- Maneuver Forces
- Fire Support
- Air Defense
- Counter $C^3$ (ECM, Deception, Special Forces)
- Engineering,

Combat Service Support
- Transportation (ground, air, pipeline)
- Logistics
- Maintenance
- Engineering
- Medical,

Combat Information Support[7)]
- Communications

[7)] It is assumed that ECCM is an organic electronic self-defense capability associated with each of the elements within the Combat Information Support System.

situation assessment, operational planning, specification and issuance of orders, and supervision of their execution.

Fig. 3 shows, in a simplified manner, the interaction of the four fundamental elements of military systems at one given level of $C^2$.

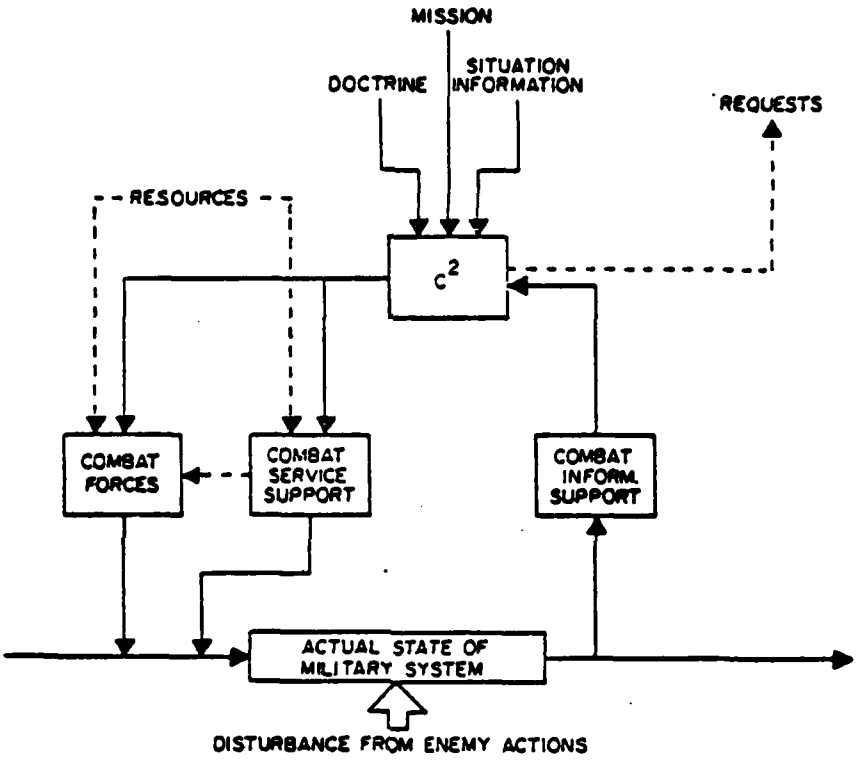



Figure 3. PRINCIPAL ELEMENTS OF MILITARY SYSTEMS IN A COMBAT ENVIRONMENT.

At any one level, the particular elements resemble organic forces or systems of the respective $C^2$-authority. Depending on the $C^2$-level, the individual elements may be more or less in evidence. For example, on a maneuver company level, the combat service support element is almost non-existent. Organic combat information support is frequently provided by ad-hoc systems temporarily created to collect information on the enemy (reconnaissance patrols). Thus, in addition to the organizational elements (i.e., objects defined in the organization chart), the generic organizational

to apply the ROSS principles, an auxiliary object was introduced when the first prototype ROSS-based combat simulation was designed (see Klahr et al [6]). There, three such objects have been included, the Scheduler, the Physicist, and the Mathematician.

The Scheduler may be considered as an "ominiscient god-like being" which, given current information, anticipates the up-coming non-intentional events and informs the appropriate actors of their occurrence.

The Physicist accounts for physical phenomena such as terrain and weapon effects, ECM, smoke, etc.

The Mathematician performs all complex mathematical computations (e.g., determines position coordinates, intervisibility, hit probabilities). Thus, all mathematical detail can be removed from the behaviors of objects making the code rather readable.

## 4. OBJECTS IN A LANDWARFARE GAME

In order eventually to be able to develop a war gaming system within a given $C^2$-system and based on an object-oriented software system, it is necessary to specify

a. the object hierarchies to be considered,
b. the attributes of the objects,
c. their behaviors.

### 4.1 Object Hierarchies[6)]

#### 4.1.1 On the Nature of Hierarchies

Disregarding, at this time, auxiliary objects that may eventually have to be created for the wargaming simulations, the objects/actors in a military force fall into one of four main categories, namely (1) Command and Control, (2) Combat Forces, (3) Combat Information Support, and (4) Combat Service Support. While the latter three comprise physical entities, the first may be thought of as a conceptual entity performing the (conceptual) tasks of

---

6)Examples used to illustrate hierarchies and/or allocations of forces and assets do not necessarily reflect present or past policies of any NATO army. They merely serve the purpose of explaining the nature of some fundamental relationships relevant for the design of a simulation system in an object-oriented programming language.

Behaviors are defined in terms of a sequence of messages which the actor, when receiving a message, issues on to other actors or to itself. The other actors also have such behaviors which allow them, when receiving the messages, to respond.

Example: The commander of a tank company orders his platoons to stand by for a certain mission. When receiving that order, the platoon commander's response might be to tell the tank commanders to assume tactical formation and to tell himself to start observing the mission area. This behavior might look, schematically, like this:

(ask PLT when receiving (CP orders mission standby)
    (tell MBT assume tactical formation)
    (tell PLT observe mission area))
(ask MBT when receiving (PLT orders tactical formation)
    (tell driver start engine)
    (tell MBT to move to position)

Each of the statements in parentheses resembles a ROSS command. The syntax is

(<rword> <object> <message>),

where

<rword> is one of {ask, tell}

<object> is an actor

<message> denotes the message sent to that actor.

In simulation, behaviors get invoked by matching the pattern of the incoming message (e.g., (CP orders mission standby)) and its associated actions against the patterns of each of the pattern-action pairs in the set of behaviors of the respective actor or object.

As is the case for the actors' attributes, behaviors may be inherited. Thus, when an actor receives a message, he first checks its own behaviors for a response. If he cannot find one, he searches the behaviors of his parents, and then of its ancestors up the line, until a matching behavior is found.

### 3.2.2 Auxiliary Objects

When applying ROSS to wargaming, there are numerous events that do not necessarily correspond to a real-world message transmission (e.g., an approaching attacker formation does not notify the defenders that it has entered their visibility range and should be seen). In order to still be able

c. MBT:

| | |
|---|---|
| gun mount | revolving turret |
| fire control | fully stabilized |
| . | |
| . | |
| . | |
| parents | BT. |

The first example implies an instance-actor, the second a class actor or generic object with regard to the first example and an instance-actor with regard to the third example. The attributes associated with the generic actors are true for each of the respective instance actors. Each individual MBTA inherits all properties from its generic parent MBTA. Similarly, the class of MBTA's inherits all properties from a generic parent MBT, etc. Thus, the notion of instance and class relates, in a "technical" hierarchy, to the objects of two neighboring hierarchical levels.

In an organizational hierarchy, the notion of instance and class may not always be relevant, at least not in a purely hierarchical sense. For example, organizationally the individual battle tank is an element of the next higher entity, the platoon. However, from this it does not follow that the individual tank would or could inherit any attribute from the platoon. But an individual platoon, being an instance of a generic platoon, inherits the latter's attributes and behaviors. Thus, in an organizational hierarchy, the instance-class relationship is rather a lateral one. One may say that the organizational structure defines a hierarchy of generic objects and the implementation of a structure resembles a hierarchy of instance-objects. In this sense, scenario generation involves, among others, the creation of an instance hierarchy from a generic hierarchy.

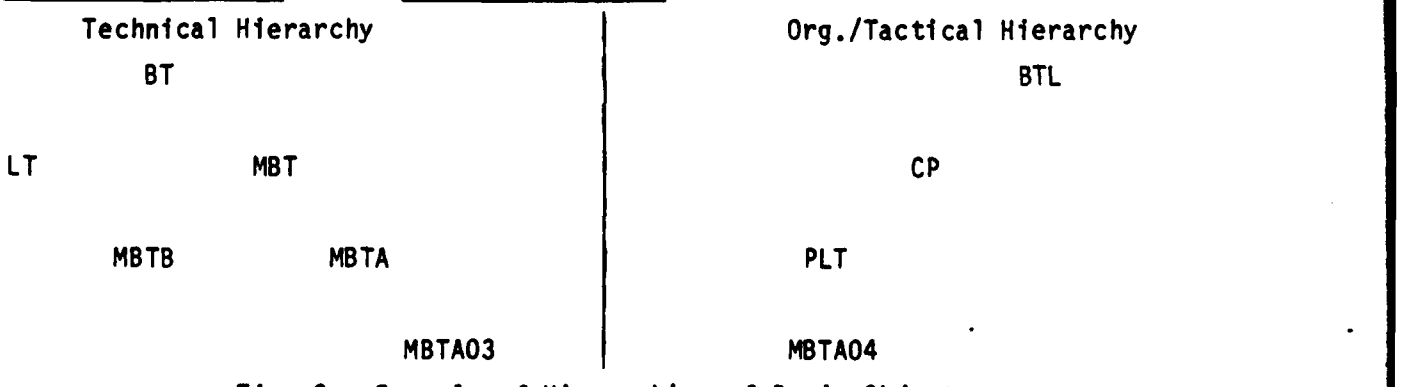



Fig. 2. Example of Hierarchies of Basic Objects.

particular, it should help solve rather elegantly the aggregation problems that exist with such model hierarchies (see Huber [5]).

### 3.2 Example of an Object-Oriented Language: ROSS[5)]

ROSS is an object-oriented programming language developed by the RAND Corporation. Its "message-passing" style of programming models a system in terms of a set of actors (objects) and their behaviors (rules for actor interaction). The actors interact by sending messages invoking a behavior.

#### 3.2.1 Actors and Messages

Consistent with the notion of instance and class in object oriented programming, ROSS distinguishes two distinct actor types: (1) instance-actors, representing individuals or individual objects and (2) class (generic) actors, representing sets of individuals or objects (see [3], p. 2).

Examples:

a. MBTA03:

| | |
|---|---|
| position | coordinates |
| status | standby |
| rounds left | 15 |
| . | |
| . | |
| . | |
| parents | MBTA. |

b. MBTA:

| | |
|---|---|
| average speed | |
| - terrain A | 50 km/H |
| - terrain B | 35 Km/h |
| gun caliber | 120 mm |
| . | |
| . | |
| . | |
| parents | MBTA. |

---

[5)] The explanations of this paragraph are adapted from [4] and draw on some examples discussed in [6].

in the previous section. They represent the simulation entities[9] within the particular gaming configuration. Thus, with regard to basic objects belonging to one of the three "physical" categories (combat forces, combat information support, combat service support), their definition and their attributes depend on what gaming configuration one wants to implement. For example, in a Corps-level game of the "nested model" variety the basic objects may be individual combat platforms. In an aggregated Corps game they might be armor brigades, battalions or companies (see, e.g., IABG's Corps/Division-level KORA model [8]). However, irrespective of the aggregation level of the game and the functional category of the basic object, we may distinguish attribute classes that are common to all basic objects resembling item-nodes as defined in section 4.1.3. These physical classes relate to

- geometry
- performance
- strength
- position/location
- posture
- supply level
- morale
- parents

With regard to the basic $C^2$-objects (commanders and their staff at all command levels), the conceptual attribute classes include

- policy
- status
- plans/requests
- parents

4.2.1 Physical Attribute Classes

The geometry class subsumes all parameters describing the spatial distribution of the basic object as a function of its status, the prevailing terrain and weather conditions, and the threat.

---

[9] A simulation entity defines the inner bounds of a system to be simulated. It can be thought of as a black box that responds to external stimuli. Its internal structure is of no concern in the simulation.

- Performance attributes describe the basic object's mobility, vulnerability, and immediate mission-related output (e.g., firing rate, maintenance manhours, transportation capability [tokm/hr] etc.). They are, among others, a function of the object's actual strength and posture, terrain and weather conditions, supply levels, morale, and threat.

The strength attributes include the TOE-strength (100%), the actual strength and an indication of which inorganic units are allocated to the respective basic object. Inorganic units are themselves basic objects subject to description.

The position/location attributes specify the instantaneous geographical coordinates of the basic object or its center of gravity as well as its spatial orientation.

Posture attributes provide a description of the instantaneous activity of the basic object (e.g., maintenance, resupply, movement to staging area, stand-by, advance to contact, attack, defense, withdrawal, etc.).

Supply level parameters permit determining the respective object's endurance, i.e., for what (further) length of time the various activities of the basic object involving consumption of supplies may be continued.

Morale descriptions classify levels of morale (fuzzy sets!) from which the relative capability of the basic object to implement its potential as described by the (physical/tactical) performance parameters may be determined via set membership functions. This relative capability can be thought of as the complement of the probability that the object is suppressed.

For item-node-type objects the attribute class parents refers to the immediate operational ancestor and to the generic parent. The operational parent identifies the superordinate $C^2$-node (e.g., the Btl commander and his staff, in case the basic object is a tank company). From the operational parent the basic object inherits its mission/task as well as the rules and conditions to be considered when executing the mission/task (see following section 4.2.2). The generic parent is the class object that represents the set to which the respective instance object belongs (e.g., Cp X is an instance of the generic class "tank company"). From the generic object the basic object inherits all attributes and behaviors typical for the respective generic class (see [3], p. 15).

### 4.2.2 Conceptual Attribute Classes

The attribute classes of the basic $C^2$-objects may be considered to describe the inputs to and outputs of the respective command decision process (see Fig. 9).

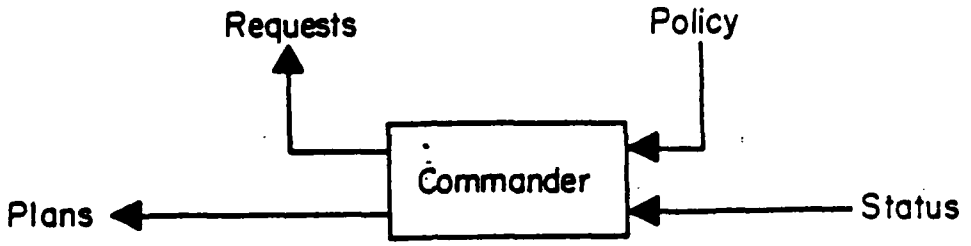



Fig. 9 Command Decision Input/Output

The policy class includes all attributes given to the commander by higher $C^2$-authorities and/or related to the set of "invariables" determined, among others, by the rules of engagement, doctrine, and international conventions. The attributes specified by higher $C^2$-authorities specify the mission or task of a military unit and organizational and tactical conditions that have to be met (e.g., boundaries and coordination with neighbors).

The status attributes specify the commander's perceptual data base on the state of the world (order and state of battle). In addition they may include all aggregated information items that a commander (or his surrogate) may require for his decisions (e.g., trend data on consumptions, availability and losses, effectiveness summaries, etc.).

Plans/requests specify missions, tasks, operational concepts, contingency plans, and operational orders for subordinate units, and requests for inorganic support to superordinate $C^2$-authorities.

The attribute parents names the operational parent of the respective $C^2$-actor as defined by the operational (instance) hierarchy of the field army as well as its generic parent.

### 4.3 Behavior of Objects/Actors

The dynamics of complex systems are the manifestation of the interaction between their elements. Object-oriented software systems assume that the interactions between pairs of elements (objects/actors) basically are brought about by or consist of messages which cause the addressed element to respond by directing messages to other objects or to itself. The response that a particular message induces in an object is one of the object's behaviors. The set of behaviors (i.e., all message-action pairs) of an object reside, together with its list of attributes, with that object.[10)]

With regard to hierarchically organized systems, a top-down analysis offers a natural approach to defining the behaviors of objects. Starting, e.g., with the theatre commander at the top and proceeding down the hierarchy as illustrated in Fig. 7, the following questions need to be answered for each object or actor:

(1) What is (are) the mission(s)/task(s)?
(2) Which activities or actions are necessary in order to accomplish the task(s)?
(3) Which organic and/or inorganic objects/actors are involved in carrying out the activities?
(4) How do these objects/actors coordinate and implement their actions; what are the alternatives?
(5) What conditions (state of the world, object attributes) must prevail for the alternatives to be implemented?
(6) What messages need to be passed among the involved objects/actors to implement the alternatives?

By answering these questions, progressively denser graphs might be derived in which the nodes denote objects/actors and the (directed) links represent messages to be passed among the objects to accomplish the assigned mission or task. Fig. 10 illustrates, in a simplified manner, some of the real-world message flows that might take place throughout the preparation for a counter-attack by a maneuver brigade. In essence, such graphs capture the knowledge of

---

10) In object-oriented languages, behavior and attributes are treated alike. "Structurally, there is nothing to separate ordinary object properties form behaviors." ([3], p. 4)

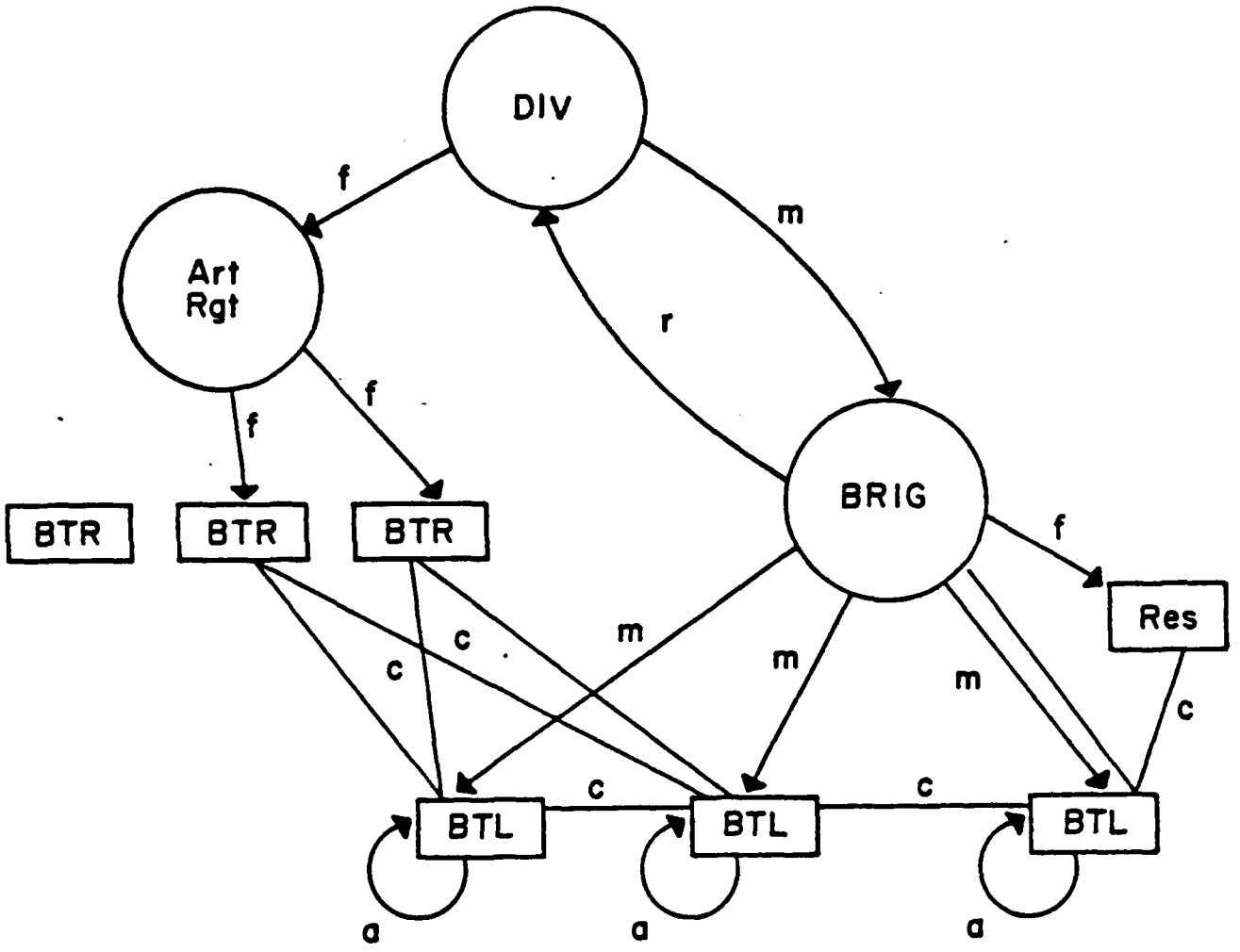
"experts" on how they go about their business. Thus, they might be considered as cognitive maps representing the perceptions and beliefs of the respective individuals about their subjective world. In the 1982 paper on "Cognitive Maps of Decision-Makers in a Complex Game" [9], Klein and Cooper report on a series of experiments to arrive at cognitive maps of military commanders in defined tactical situations. From their results it appears that cognitive mapping through gaming experiments should, in some instances, be a valuable tool to help in describing the behavior of actors/objects and in structuring the knowledge bases in tactical expert systems (see also Huber [10]).

In addition to the real-world message flows, as illustrated in Fig. 10, war gaming within an existing $C^2$-structure involves what might be called simulation messages. These are messages directed at or sent by an auxiliary object such as one requesting a "Mathematician" to determine the time of arrival of a maneuver battalion at its jump-off position. In order to do this, the mathematician may, depending on the weather conditions, request a "Physicist" to provide the coefficients for movement under rainy conditions. The "Scheduler" would tell the maneuver battalion that an enemy reconnaissance party is within its visibility range if that battalion is marching with a look-out (the scheduler keeps track of all objects' positions). The battalion, in turn, might order itself to engage the enemy, requesting again a mathematician to determine the results of the firefight. This partial example illustrates how friend-foe interactions are handled through message passing between basic and auxiliary objects/actors (see also 4.4).

Since their definition involves two different kinds of experts, we need to distinguish between the behavior of real-world (basic) objects/actors and auxiliary objects/actors. The former is the concern of the military expert, the latter of the systems analyst. While the behavior of real-world objects/actors is described as a sequence of messages to other real-world objects/actors or to itself, subject to certain (perceived) conditions prevailing, the behavior of auxiliary objects is defined in terms of messages both to other (real-world and auxiliary) objects/actors and to formal models of real-world processes being simulated by the respective object.

### 4.4 Basic Interactions of Objects/Actors

Fig. 11 illustrates, in a simplified manner, the basic interactions between objects/actors in a game. Subject to a given objective or mission (as specified in the plans/request-class of attributes; see section 4.4.2), the




f Fire Support Allocation

m Order to Move to Attack Position

r Request for Fire Support

a Generation of Recce Parties

c Coordination

Fig. 10 Example of Some Message Flows During Preparation for Brig Counter-Attack.

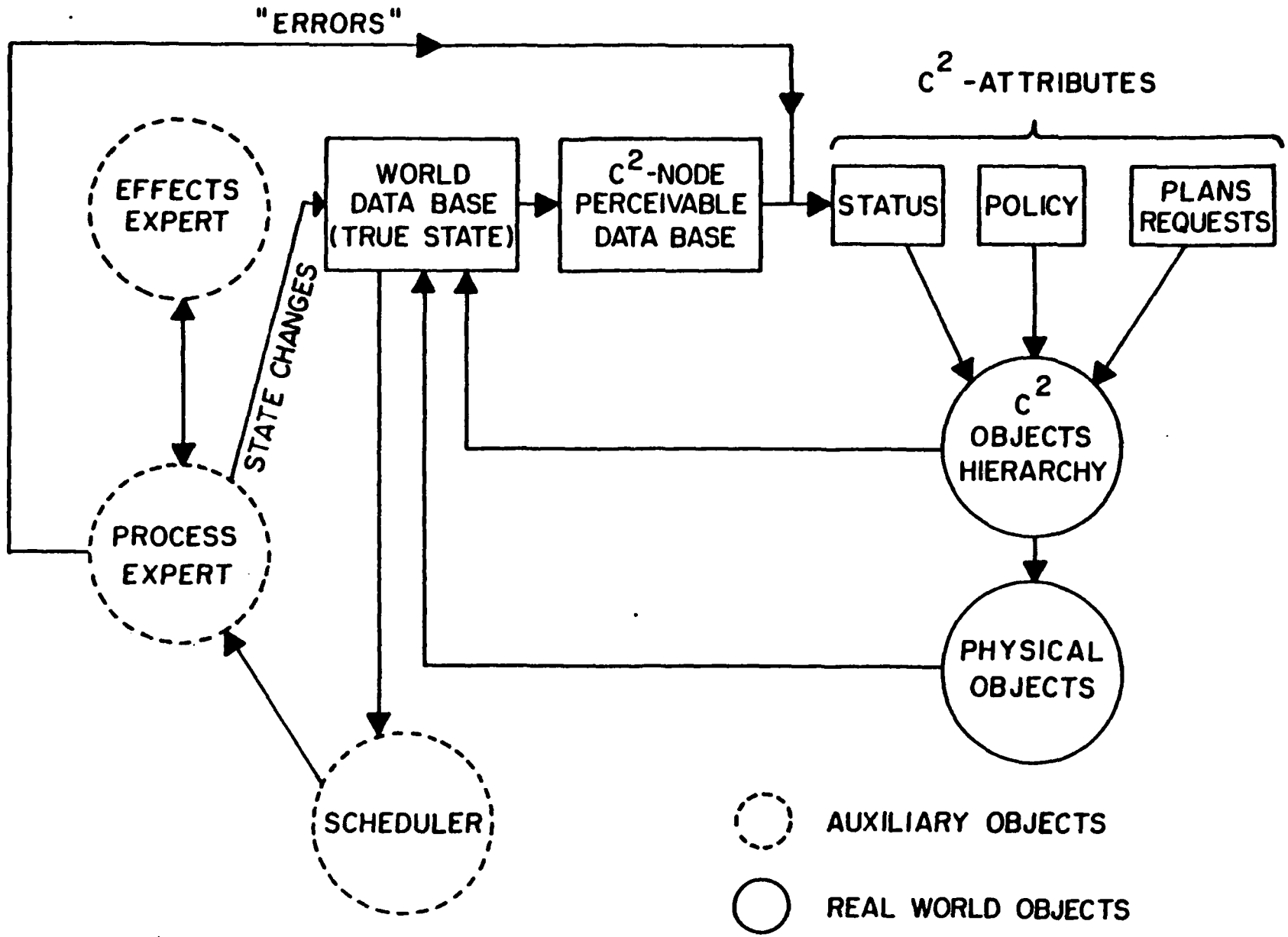



Fig. 11. BASIC INTERACTIONS OF OBJECTS

specific behavior of the real world actors is essentially a function of their perception of the state of the world, including the physical world (as defined by the status attributes) and the conceptual world (as defined by the policy attributes). Their behavior eventually results in actions which (might) change the true state of the world and, depending on the respective actors' perceptions, their status information. The status information can be thought of as the individual perceptual data base of a $C^2$-object/actor derived from the perceivable information on the true state of the world (i.e., the information which the respective actors would have available in reality, dependent upon the rules and performance characteristics of the $C^2$-system and its communication channels).

The "Scheduler" is an artificial object that knows the true state of the world, i.e., the instantaneous attribute values of all real-world objects/actors. Thus he is aware of all interactions between actors and/or objects which influence the execution of the actors' plans. He calls upon the "Process Expert" for information on the outcome of these interactions, considering the effects of whatever physical and psychological phenomena are involved. These in turn are provided upon request by the third auxiliary object, the "Effects Expert."

The terms "process expert" and "effects expert" indicate that in the context of land warfare the artificial ROSS-objects "mathematician" and "physicist" need to be extended so that they may accommodate mental process models and psychological phenomena. Thus, they resemble expert systems with two sets of experts each, namely the

Process Expert with the
- Tactician and
- Mathematician, and the

Effects Expert with the
- Psychologist and
- Physicist.

"Tactician" and "psychologist" are generic names for sets of judgmental or empirical input-output relations (black box models) on tactical/operational processes and human factor effects (e.g., suppression caused by fear). The terms "mathematician" and "physicist" are generic names for sets of mathematical models designed to calculate state changes (by the mathematician) and the requisite physical effects parameters (by the physicist).

For example, the movement of a unit involves an interaction between that unit and the terrain. Whenever a unit's plan requires that unit to move, the scheduler asks a mathematician to determine (for example) the time required to perform the move. To this end, he provides the mathematician with all the necessary information on the instantaneous state of the world, such as the unit's mobility characteristics, weather, terrain data, route information and others. Given this initial information, the mathematician calls upon a physicist to provide information such as the average velocities of that unit over the various legs of the route. As th. position changes (calculated by the mathematician) take place, the true state of the world is changed accordingly.

In our example, the mathematician's model may be of the simple form

$$T = \sum_i d_i / v_i$$

with i denoting the legs of the movement route (defined by the route itself and the terrain types encountered), $d_i$ their distance and $v_i$ the average velocity over each leg i. The physicist's model to determine $v_i$ might be either a mathematical function or a look-up table accounting for parameters such as the respective unit's vehicle types, the movement formation, movement tactics, terrain parameters (type of ground, topological features) and weather.[11)]

The perceivable state of the world includes the information on that part of the real world which is accessible to the respective actor in a given $C^2$-system under normal operating conditions. The status information can be thought of as the "corrupted perceivable information," as defined by Erickson et al. [11], i.e., the information available to the respective actors accounting for "errors" due to system malfunctions or exogeneous interference

11) Thus, the models represent basic operational and tactical/organizational processes. The process expert is considered to be the set of all tactical/organizational process models (such as firing, maneuvering, supply, maintenance, etc.); the effects expert is the set of physical and psychological process models (such as vehicle movement, LOS, search and detection, weapon effects, suppression, etc.).

(ECM, weapon effects). In any case, the scheduler would be aware of such events or plans and call upon the mathematician to determine, together with the physicist, these errors. Similarly, he would be aware of endogenous and exogenous interference in the message traffic between $C^2$-objects and item-objects and treat them in an analogous manner.

## 5. MESSAGES IN A LANDWARFARE GAME

Most links in Fig. 11 symbolize "artificial" messages required to carry on, in a DWG-system, the simulation of the combat system or parts thereof. In order to distinguish these "simulation messages" from the messages passed in real life operations between the basic objects of the defense system, the latter will be called "system messages." Of course, in DWG-exercises all system messages would be "life" messages which are passed on through the communication channels of the operational $C^2$-system, at least to the extent that $C^2$-nodes participate in the exercise interactively. In that case, the DWG-system might be used as a testbed to experimentally investigate operations related to the further evolution of $C^2$-systems, in particular with respect to the development and test of decision support systems and strategies to cope with counter-$C^2$ measures.

### 5.1 System Messages

With due regard to the inputs and outputs of $C^2$-nodes as depicted in Fig. 3 and 6, one might distinguish four basic types of system messages, namely

a. Orders (to subordinate actors/objects),
b. Requests (to superordinate actors/objects),
c. Coordination (with neighbors and inorganic supporting actors/objects), and
d. Information

#### 5.1.1 $C^2$-Styles and Messages Flow

For a given $C^2$-level, the formal content of a message may differ depending upon tactical doctrine and operational principles, national $C^2$-styles and traditions. Order-type messages by commanders practicing a mission-oriented $C^2$-style specify only what it is they expect the immediately subordinate actors or units to do. Commanders practicing order-oriented $C^2$-styles specify, in addition to "what to do," how they want it to be done. The degree of order-orientation may be expressed in terms of the number "w" of

subordinate levels down to which the manner of executing a mission or task is specified.[12)]

If we consider level 1 as the one where the actions causing the effects necessary for mission accomplishment physically take place, then Fig. 11 tells us that the formal message content of the messages originating at the next higher level 2 should be identical for all degrees of order-orientation. However, the "semantic" message content,[13)] specifying how to perform the level 1 actions, would have been defined at the $C^2$-level (1 + (w + 1)). At each of these (and higher levels), the average "length" of the messages would be about a factor $m^w$ higher than in the purely mission-oriented case, where m denotes the number of immediately subordinate actors/objects (assumed to be identical at all levels). A similar reasoning should apply to the frequency of request-type messages, which leads us to conclude that the $C^2$-style should be of considerable consequence with regard to communication channel requirements.

5.1.2 Order-Type Messages

In a landwarfare system, order-type messages may be classified into three categories:

- Change of Tactical Posture (TP)
- Resource Allocations (RA)
- Management of Supplies (MS).

Management of supplies refers to the distribution and usage of consumables (ammo, POL, food) at hand. At the higher $C^2$-levels (Corps and Division), the respective messages usually imply distribution-orders directed at logistics and transportation units, either in anticipation of certain consumption rates to be expected with the maneuver and support force operations, or in response to requests originating at subordinate $C^2$ levels. At the lower $C^2$-levels (brigade and battalion), MS-messages most often imply usage-orders constraining somehow the usage of consumables.

---

12) Thus, a purely mission-oriented $C^2$-style may be considered to be of a zero-degree order orientation. Under a first-degree order orientation-style, a message would tell the addressed actor, how to employ his immediately subordinate actors (e.g., a brigade commander would tell a battalion commander what he expects the battalion's companies to do).

13) Following the nomenclature of object-oriented programming, the semantic message content may be considered as an instance of a generic message.

Resource allocation includes the allocation of maneuver force reserves, of fire support and (inorganic) air defense, engineering and medical support, and maintenance (battle damage repair). Similarly to MS-messages, RA-messages are directed at the respective units either as part of an operational plan or in response to requests. Combat information support systems (see Fig. 4) might, within a DWG, be considered as sources operating continuously by sending out information messages, but being subject to direction by $C^2$ nodes and to interference by counter-$C^2$ forces and means. Counter-$C^2$ itself is employed strictly in the context of operational plans in direct or indirect support of maneuver forces.

Change of Tactical Posture refers to the control of maneuver forces. A tactical posture is indicative of a certain behavior of a maneuver unit. Among others, tactical postures include the combat postures of

- advance to contact
- attack
- defend
- delay
- withdraw
- observe
- search

and the preparatory postures of

- march (movement into rest areas and standby positions)
- standby
- rest
- resupply
- prepare (for any of the above).

At any given point of time, maneuver force actors/objects either assume one of the above postures or are in the process of transition from one to another.

For each of these postures, the behavior depends not only on the type of object, its equipment and certain environmental states (e.g., threat, terrain, weather), but also on tactical doctrine and operational principles, i.e., on national $C^2$-styles and traditions. So does the behavior when changing from one posture to another. In fact, under mission-oriented $C^2$, behavior descriptions may contain chains of posture sequences which the respective object may assume on its own, depending on instantaneous conditions.

### 5.1.3 Information Messages

All incoming messages providing a $C^2$-node with data on the state of the world (status information, see section 4.2.2) are classified as information messages. With regard to information on the enemy, such messages may originate from the actors/objects of the combat information system either routinely and/or upon request from $C^2$-nodes of the combat or combat service support forces. The latter may also order such information messages to be generated by their organic means (e.g., dispatching of recce patrols by a company commander).

With regard to information on the friendly status, such messages originate, in each functional hierarchy, from the subordinate $C^2$-nodes either as part of a routine or upon request.

As is true for all system messages, information messages are passed on through the real (or simulated) communication channel subject to real or simulated interference (see section 4.4).

## 5.2 Simulation Messages

The messages required to stimulate the simulations in a DWG-exercise (simulation messages) are passed either from auxiliary objects to real world objects or between auxiliary objects. In the former case, these messages would be information messages, in the latter, they would be order-type messages. In addition, input messages are required to specify the initial simulation scenario and to effect changes of simulation entities during the course of a DWG-exercise. Through administrative messages the user initiates documentation or is instructed about errors.

Information messages are essentially messages from the "Scheduler" to $C^2$-actors/objects, corrupted by the endogenous or exogenous errors of the respective communication channels and sensors. As Fig. 11 illustrates, the Scheduler would essentially release, from his real world data base, the information that is perceivable by the respective $C^2$-actor/object while simultaneously ordering the "experts" on the involved physical processes (e.g., mathematicians and physicists) to corrupt that information by simulating the involved collection and communication processes subject to the reliability characteristics of the employed systems and to ongoing interference by the enemy.

A typical sequence of information messages in a land warfare DWG-exercise might be the one required to simulate the interactions between

attacking armor and defending anti-armor units. Suppose the simulation is taking place at the lowest level (e.g., company defending against a battalion-level attack with resolution down to the individual weapon system); then the following sequence might emerge:

(1) The Scheduler, being aware of both sides' plans, orders the movement expert to generate the positions of the attackers in the respective terrain.

(2) As soon as the attackers get into mutual visibility range, the Scheduler orders the LOS expert to determine line of sight incidents between attacking units and defending units in search mode.

(3) Given LOS, the Scheduler orders the search and detection expert to determine detection incidents.

(4) The Scheduler informs the detecting unit(s) that it (they) detected the enemy.

(5) Depending upon the prevailing conditions, the detection message triggers one of the behaviors defined for the detecting side. Suppose the defenders detected a platoon-size tank formation travelling in a certain direction outside the expected main axis of attack and beyond effective firing range. In that case, the commander might alert all his defensive elements to keep searching with special attention to the sector where the initial detection occurred.

(6) The Scheduler again orders the LOS and the search and detection experts to determine LOS and detection incidents (see 2-4).

(7) Suppose the defender detected further tank formations in the vicinity of the initial one heading in the same direction. The resulting simulation message to the defending commander makes him respond by ordering a fall-back and regrouping of his defenses so as to be able to meet the main threat.

(8) The Scheduler orders the movement expert to generate the time-position trajectories of the defender and to determine detection incidents (see 2-4).

(9) Suppose the defenders remain undetected and the attacking enemy comes within effective firing range of the defenders. If the defender's behavior prescribes opening fire in a coordinated fashion, the Scheduler would order the attrition expert to determine mutual losses in the ensuing firefights, accounting for fire coordination and initial surprise of the

attackers. These firefights might constitute the mini-battle type exchange described by Rowland [12].

(10) Consistent with the "judgment" of the "experts," the schedule's world data base is continually updated, with the real world actors (opposing commanders) being informed about those state changes that they should be able to perceive when fighting the battle in reality.

This simple example (e.g., no air/ground interactions) illustrates that the simulation message flow could indeed become somewhat complex and dense, perhaps requiring a parallel processing capability if there is a large number of units that are explicitly simulated, so that the behaviors of each need to be invoked by matching the patterns of system and simulation messages against each of the pattern-action pairs in the set of behaviors of the simulated actors/objects. For this reason, Erickson believes, that, while the message passing style of programming of ROSS is "... natural for expressing the communications behaviors of (real world) entities participating in the simulation, it complicates the processing required to model the physics of the simulation environment, such as the interaction effects between detectors or weapons and their environment." (see [11], p. 11).

However, it seems that there are possibilities to economize on these processing requirements. For example, Faught and Klahr [13] describe "proximity detection" algorithms aimed at reducing, by significant margins, the number of checks required to establish when basic objects are close enough in their physical environment to interact. Also, the communication (in the simulation) between the "process experts" and the simulation entities via the "scheduler" is certainly a very natural one, if we accept that the "messages" (e.g., detection, kill) generated by the experts do indeed resemble perceptions by the simulated entities in the real world.

The input messages essentially include the commands for defining and changing basic actors/objects, i.e., their physical and conceptual attributes and their behaviors. They would be similar to the ROSS commands for creating (generic and instance) actors and manipulating their attributes, behaviors and plans (see [3] pp. 15-30).

### 5.3 Message Formats

Table 2 summarizes the basic messages in a DWG-system briefing discussed in the previous section.

TABLE 2

Basic Messages in a DWG-System

| Message Class | Message Type | Message Flow |
|---|---|---|
| System Message | Order | $C^2$-actor → subordinate actors |
| | Request | $C^2$-actor → superordinate actors |
| | Coordination | $C^2$-actor → "neighboring" actors |
| | Information | all actors → all $C^2$ actors |
| Simulation Message | Order | auxiliary objects → auxiliary object |
| | Information | scheduler → $C^2$-actors |
| | Input | user → DWG-system |
| | Administration | user → DWG-system |

The principal message format might have the format

(<time><sender><receiver><type><message>)

where

<time> is the time at which the message is issued

<sender> is the originator of the message

<receiver> is the addressee of the message

<type> indicates the type of message

<message> denotes the message content being sent to the addressee.

In case of orders and requests, the message content indicates what is being ordered or requested and at what time the respective actions should be initiated and/or when they need to be completed. For example, an order-type system message from a brigade commander to a battalion might read

(take hill 59 before 041645Z July 83).

In case a completion time is specified, noncompletion would cause the battalion commander to send to the brigade either requesting an extension of the completion time, in case there is a good chance of still accomplishing the task, or else requesting new orders.

Request messages might convey requests for combat support, combat service support, information, (new) orders, and special events (e.g., meetings). In addition to the request, they specify the time span (from-to) over which the support is requested, or the time point as of which the information is requested. The message

(status of battalion 289 at 04030Z July 83)

requests information on the attributes and plans (see section 4.2) of battalion 289 as of 4:30 a.m. Greenwich time on July 4, 1985.

Coordination messages inform neighbors and/or supporting units of one's own plans and actions. Information messages include an indication of the actor/object about whom/which the information is provided, the age of the information, a specification of the information requested or provided (e.g., attribute values and plans of the respective actor/object). Message content formats need to be designed with due regard to the type of information and might follow the attribute classification scheme proposed in section 4.2.

5.4 System Message Processing

In addition to facilitating standardization of message formats, a classification of messages similar to the one proposed above is also necessary in order to establish message processing priorities in automated $C^2$-nodes. Prioritization is necessary either when several messages come in nearly simultaneously or when additional messages arrive while the $C^2$-actor is still in the process of selecting a response to a message which arrived previously.

Of course, the requisite prioritization rules essentially capture the very essence of staff work. In case of simultaneously arriving messages, priorities might be simply established by determining the "dominance" of messages with respect to the conditions controlling the response of the $C^2$-actor. For example, information and coordination messages may alter the conditions upon which the responses to requests and orders depend. Thus, they should be processed first. Also, responses to requests might severely curtail or make impossible responses to orders and vice versa. Since the latter originate from superordinate and the former from subordinate $C^2$-actors, one might establish the rule that order messages dominate request messages. However, whether such a rule makes sense operationally is open to question. It might be better to request from or propose (to the superordinate $C^2$-actor) new orders compatible with the request (from the subordinate $C^2$-actor).

While such a request is on its way or being processed at the superordinate $C^2$-node, new information messages might come in changing the requisite conditions such that the request is rendered invalid. This is an infraction similar to the one indicated above, where a $C^2$-actor is given updated information while searching for a response to an order or request. To reinstate the search based on the new data might seem advisable in many cases.

But in some instances it might lead to a "decision-blockade" where the incoming message traffic is of a density that does not permit the $C^2$-actor ever to finish his search for a response. Of course, to prevent such a case additional rules limiting the number of such up-dating instances could be introduced. However, even this would not necessarily make sense.

These examples indicate that a sensible set of priority rules can very likely not be established by considering formal criteria (such as type of message, dominance, time of arrival, etc.) only. Significant empirical research on staff work, perhaps as part of DWG-experiments, may well be necessary in order to define $C^2$-actor behavior adequately.

## 6. SPECIAL ASPECTS OF DWG-DESIGN

There appear to be two principal problem areas which distinguish the design of a DWG-system from that of a traditional game operating on one mainframe computer. One is related to the supporting data bases, the other to the timing of distributed processing.

### 6.1 Timing Distributed Processing in DWG

Synchronization of distributed processing of interdependent and interacting processes is one of the major problems facing distributed simulations in general (see, e.g., Graham [14] and LaPlue [15]). Given sufficient data processing and transmission capability, of course, the message-passing mechanisms controlling the dynamics of simulations using object-oriented programming languages would eliminate synchronization problems altogether, at least in theory, if it could be assured that all messages in a DWG-exercise are received by the addressed objects exactly at the time when they would be received in reality (i.e., real-time message flows).

With regard to system messages, comunication between interactive (manned) $C^2$-nodes in a DWG-exercise does imply real-time message flows if the $C^2$-nodes operate in a realistic mode and use the (physical) communicaton channels of the operational $C^2$-system. When $C^2$-nodes are automated and/or communication between them is being simulated, however, we need to have a "timing expert" who determines, as the simulation proceeds, the time $\Delta t_r$ required in real world operations for the simulation time step $\Delta t_s$ to take place. Of course, this is always necessary for "communication" between artificial objects, the processes simulated by the process and effects experts, and the communication between artificial object and $C^2$-actors.

As long as $\Delta t_s \leq \Delta t_r$ for all simulated processes, there is no timing problem. The results of the simulations are made available to the addressed object/actor or the world data base and the perceivable data base (see Fig. 11) only after the time $\Delta t_r$ has elapsed. However, if $\Delta t_s > \Delta t_r$, the results of the respective simulations come too late, thus possibly invalidating the simulations of another process to which the delayed simulation contributes inputs either directly or through data bases. Therefore, all such simulations must be re-processed with the game clock being slowed down by extending its time unit by a factor $\Delta t_s/\Delta t_r$ compared to the time unit of the real world clock. If there are several processes for which $\Delta t_s > \Delta t_R$, the reduction of the speed of the game clock is based upon the process with the longest delay, i.e., max $\Delta t_s$. In the iteration of the simulation time step, all formerly delayed processes will then be in time, provided that the messages synchronizing the game clocks are received properly. The principal burden thus falls upon the communication system. A slow-down of the game clock also implies that the DWG-exercise henceforth proceeds at a lower pace than real world operations, requiring rectification as soon as the simulation delays start to diminish again. Fig. 12 shows an algorithm for the adaptation of the game clock time $t_G$ such that real world time conditions $t_W$ are restored as soon as possible.

Of course, if delayed simulations occur frequently or $\Delta t_s >> \Delta t_r$, the time periods in the DWG-exercise over which the game clock runs behind the "world" clock may become quite extended, thus defeating one of the fundamental requirements for a DWG-system in applications such as training under realistic time conditions or when investigating human response characteristics in the $C^2$-system. In such cases, the processing capability of the respective computer and/or the bandwidth of the data transmission channels need to be enhanced.

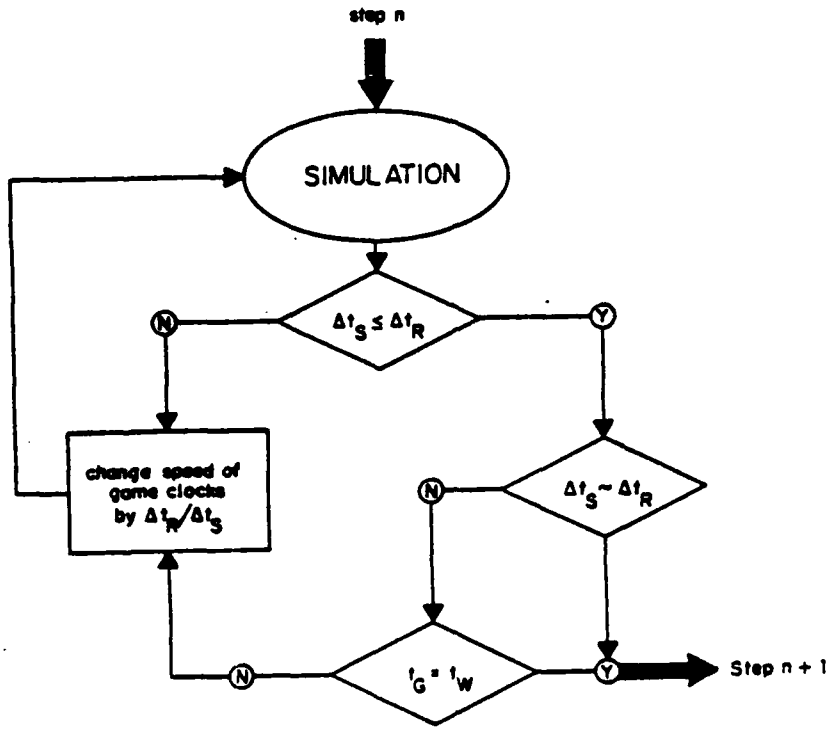



Figure 12. TIME ADAPTATION IN DWG.

6.2 Data Bases

The data requirements for a DWG-exercise depend upon the level at which the exercise is taking place and, to some extent, on the configuration selected (i.e., on the degree of involvement of live $C^2$-actors and the degree of aggregation, see section 4.1.3). Thus, the data base needs to be partly created anew for each DWG-exercise. However, in a properly designed DWG-system, creation of an exercise data base only involves 1) the nomination of the basic objects participating in the game, and 2) an eventual transfer of these objects from their "home residence" to the "exercise residence."

The "home residence" of a basic object is the data base (in the computer) of the $C^2$-node to which the respective object is immediately subordinate organizationally. If this $C^2$-node is a participant in the

exercise, no transfer is necessary. In case it is not, it and/or its subordinate objects are transferred to the data base(s) of the participating $C^2$--node(s) with which the respective objects interact in the exercise.[16] The transfer might be accomplished either through a transmission of the data upon request via the $C^2$-system's data links, or by newly "creating" the respective objects for the duration of the exercise. In an object-oriented software environment this should be rather easily accomplished through an input message.

## 7. ON THE DEVELOPMENT AND IMPLEMENTATION OF DWG-SYSTEMS

It appears that object-oriented programming systems hold the potential to enhance greatly not only the development and operation of decision support software in military command and control systems, but also the modelling of military processes and the use of models in general. The hitherto practiced philosophy of having "turn-key" systems developed and maintained by outside experts (example: GEADGE-system) could be eventually replaced by one that considers the development and implementation of $C^2$-software as a part of the evolution and adaptation of $C^2$-systems (to new technologies, tactics, operational concepts, and strategies), with the responsibility being vested in the $C^2$-function itself and basically performed with its organic man- and brain-power. Thus, it is not unlikely that, except for initial research and feasibility studies, Distributed War Gaming Systems would be largely developed and operated by the military operators themselves, supported by outside experts only when special expertise (e.g., in computer science and mathematical techniques) is required. Such an approach would greatly alleviate the acceptability problems that "turn-key" gaming systems invariably face, at least initially, especially if their design is based on models which had been

16) As an example, one might think of a brigade-level DWG-exercise in which the adjacent battalions of the neighboring brigades participate in the exercise without these brigades and their battalions being physically involved in the exercise.

originally developed for quite different purposes.[17] In the long run, its adoption should also help in improving the empirical base of military modelling in support of planning because it would permit a "compound gaming" approach as proposed by Huber [16] to materialize.

With a view to the potential benefits of DWG-systems that are based on object-oriented software systems on the one hand, and regarding some of the risks associated with the implementation of large-scale turn-key systems on the other, a program of research and study is proposed aimed at 1) demonstrating the feasibility of DWG through prototype systems, and 2) developing a procedural strategy for the evolution of operational DWG-systems.

7.1 Research and Feasibility Studies

Work in this area would essentially include two (overlapping) phases:

(1) Comparative analysis of software-oriented languages with a special view to modeling physical interaction processes (as they occur in air-land, air, and naval war) as message-passing systems;

(2) Feasibility tests on small-scale prototype DWG-systems developed from existing (and accepted) combat models.

It is estimated that this work can be accomplished in a 2-3 year period through a properly funded research program involving competent institutions.

7.2 DWG-Evolution Strategy

The development of a procedural strategy for the evolution of operational DWG-systems should draw on the experience gained through the basic feasibility tests. A first analysis suggests that such a strategy might involve an interactive procedure involving five fundamental steps:

(1) A top-down analysis to define, as a first approximation, the attributes and behaviors of the $C^2$-actors in the relevant

---

17) This is true for almost all computer-assisted training games being fielded among NATO armed forces or presently in development or in a planning stage. It is quite possible that the acceptability problems with some of the larger of these systems will be insurmountable, thus causing in due time a reversal of the current (perhaps somewhat naive) enthusiasm with which the requirements for these systems are articulated and received.

functional areas of military organizations. This could be accomplished in seminar-style workshops involving initially surrogate experts (e.g., students in higher military educational institutions) and subsequently the commanders on the respective $C^2$-levels.

(2) Analysis of the basic (physical) interaction processes in all of the four fundamental military system hierarchies (see section 4.1.1), with a view to developing the message passing structures necessary to model the process dynamics. Concurrently, undertake the design and testing of software modules for the effects and process experts. This step might be supported by pertinent thesis-projects in OR, command and control, systems and computer science.

(3) Development of initial medium-scale DWG-prototype testbeds through integration of the results of 1) and 2).

(4) A test program consisting of a series of interactive games to
   a) assess alternative language concepts;
   b) provide information on DWG-operational requirements;
   c) test and modify the initially defined behaviors of object[18] and modelling concepts (iteration of steps (1)-(3).

(5) Development of deployable DWG-modules based on the analysis of the test program and testing of variable-resolution gaming configurations.

---

[18] It is recommended to test the a-priori defined behaviors of $C^2$-nodes in a series of iterations of <u>one-sided interactive</u> games, using an algorithm similar to the one proposed for the anticipatory defense planning concept (see Huber [7]). This way, the mutual compatibility of the (independently defined) behaviors of the opponents can be tested as a prerequisite to developing <u>adaptive behaviors</u>.

## REFERENCES


[1] Bergman, G.: Philosophy of Science, Madison, Wisconsin, 1958.

[2] Ingalls, D. H. H.: Design Principles Behind Smalltalk, Byte, Aug 1981, pp. 286-298.

[3] McArthur, D., Klahr, P.: The ROSS Language Manual, Rand-Rep., N-1854-AF, Sep 1982.

[4] Robson, D.: Object-Oriented Software Systems, Byte, Aug 1981, pp. 74-86.

[5] Klahr, P., McArthur, D., Narain, S., Best, E.: SWIRL: Similating Warfare in the ROSS Language, Rand-Rep., N-1885-AF, Sep 1982.

[6] Klein, T. H., Cooper, D. F.: Cognitive Maps of Decision-Makers in a Complex Game, Operational Research, Vol. 33 No. 1, Jan 1982, pp. 63-71.

[7] Huber, R. K.: Some Issues in Defence Systems Analysis and Modelling, In: Systems Analysis and Modelling in Defence (Huber Ed.), New York - London, 1983, pp. 11-35.

[8] Parry, S. H.: A Self-Contained Hierarchical Simulation Construct. Systems Analysis and Modelling in Defence (Huber, R. K., Ed.), New York - London 1983, pp. 565-574.

[9] Kiwull, E. - U.: KORA-Combat Assessment - Representation of Dismounted Infantry in the War Game Model KORA, IABG-M-SO 2125/10, April 1982.

[10] Huber, R. K.: Interactive Simulation as an Experimental Tool for the Evolution of $C^3I$-Systems, Transactions of the First U.S. Army Conference on Applied Mathematics and Computing, Washington, May 1983.

[11] Erickson, S. A., Balaban, D. J., Nelson, D. O.: Decision-Making in Large-Scale Military Simulation - A Requirement for Expert Systems, VCID-19802, Lawrence Livermore Laboratory, April 1983.

[12] Rowland, D.: The Place of Tactically Interactive Field Trials in Data Collection, In: Systems Analysis and Modelling in Defence (Huber Ed.), New York - London 1983, pp.

[13] Faught, W. S., Klahr, P.: An Analysis of Proximity-Detection and Other Algorithms in the ROSS Simulator, Rand-Rep., N-1587-AF, March 1983.

[14] Graham, R. A.: An Environment for Distributed Simulation of Command and Control Networks, Thesis, Naval Postgraduate School, Monterey, CA, March 1983.

[15] LaPlue, L. D.: Synchronizing Distributed Simulations: An Overview, Thesis, Clemson University, Clemson, S.C., May 1983.

[16] Huber, R. K.: The Systems Approach to European Defence - A Challenge for Operational Research Gaming, Phalanx, Vol. 15, No. 3, Sep 1982, pp. 18-21.

DISTRIBUTION LIST

| | | No. Copies |
|---|---|---|
| 1. | Library, Code 0212<br>Naval Postgraduate School<br>Monterey, California 93943-5100 | 4 |
| 2. | Defense Technical Information Center<br>Cameron Station<br>Alexandria, VA 22314 | 2 |
| 3. | Defense Advanced Research Projects Agency<br>1400 Wilson Blvd<br>Arlington, Virginia 22209 | 1 |
| 4. | Dr. Reiner K. Huber<br>Federal Armed Forces University Munich<br>D-8014 Neubiberg, West Germany | 5 |
| 5. | Professor John M. Wozencraft<br>Naval Postgraduate School<br>Monterey, California 93943-5100 | 1 |
| 6. | Professor Michael G. Sovereign<br>Naval Postgraduate School<br>Monterey, California 93943-5100 | 20 |

END

FILMED

9-85

DTIC