CENTER OF EXCELLENCE IN
COMMAND, CONTROL, COMMUNICATIONS AND INTELLIGENCE

GEORGE MASON UNIVERSITY
Fairfax, Virginia 22030

ANNUAL TECHNICAL REPORT

for the period

1 July 1994 - 30 June 1995

for

ADAPTIVE DECISION MAKING AND COORDINATION IN VARIABLE STRUCTURE ORGANIZATIONS

Grant Number N00014-93-1-0912

Submitted to
Dr. W. S. Vaughan, Jr. (3 copies)
Office of Naval Research
800 North Quincy Street
Arlington, Virginia 22217-5000

Copies to:
Director, Naval Research Laboratory
Administrative Grants Office, ONR
Defense Technical Information Center

Submitted by:

Alexander H. Levis
Principal Investigator

August, 1995

Report #: GMU/C3I-161-IR

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE August 1995	3. REPORT TYPE AND DATES COVERED Technical Report 7/1/94 - 6/30/95

4. TITLE AND SUBTITLE ADAPTIVE DECISION MAKING AND COORDINATION IN VARIABLE STRUCTURE ORGANIZATIONS	5. FUNDING NUMBERS N00014-93-1-0912
6. AUTHOR(S) Alexander H. Levis, Didier M. Perdu, Abbas K. Zaidi	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Center of Excellence in Command,Control, Communications and Intelligence George Mason University Fairfax VA 22030-4444	8. PERFORMING ORGANIZATION REPORT NUMBER GMU/C3I-161-IR
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research 800 North Quincy Street Arlington VA 22217-5660	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT UNLIMITED	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Progress in research on coordination in distributed decision making organizations with variable structure is reported.The focus of this report is on CAESAR II (for Computer Aided Evaluation of System Architectures) which is a suite of algorithms, interfaces, and COTS software for the design, analysis, and evaluation of decision making organizations. The suite has four major modules:Requirements Generation; Decision Making organization architecture generation; Selection and analysis of decision making organization; and Performance evaluation. Theoretical and empirical results of this and previous research efforts have been integrated in CAESAR II.

14. SUBJECT TERMS Decision Making; Organization theory; Colored Petri Nets; Rule based Systems; Validation and Verification	15. NUMBER OF PAGES 35
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

TABLE OF CONTENTS

1. PROGRAM OBJECTIVES

The objective of this research, as described in the proposal and the previous progress report, is the investigation of several issues related to coordination in organizations. In particular, an organization is coordinated through direct and indirect means. The direct means includes the set of decision rules that the organization members use and the commands that they issue to each other. Indirect means include the dissemination of information within the organization; for example, organization members may share information or they may inform each other as to the actions they plan to take or decisions they have made. Coordination becomes a complex issue in variable structure organizations. Not only do the decision rules and the information architecture have to work for each fixed structure, but the designer has to deal with the problem, a metaproblem, of coordinating the variability. This becomes a particularly difficult problem in organizations that exhibit substantial complexity and redundancy in their information structure. The redundancy is necessary both for robustness and for flexibility and reconfigurability. In order to address these problems two main tasks were defined; they are described in the next section. In addition, some basic work in algorithms and Colored Petri Nets needs to done to develop tools and techniques for supporting the analysis and design.

2. STATEMENT OF WORK

The statement of work, as outlined in the proposal, is given below.

Task 1: Consistency and Completeness in Distributed Decision Making

Develop a methodology for analyzing and correcting the set of decision rules used by an organization with distributed decision making. The methodology is to be based on the modeling of the set of decision rules in the form of a Colored Petri Net and on the analysis of the net using S-invariant and Occurrence graphs. The ability to verify and correct the set of decision rules has direct impact on the extent of coordination needed in an actual organization and the resulting communication load.

Task 2: Variable Structures: Heuristic rules in the Lattice Algorithm Constraints

Develop a methodology for considering additional constraints in the Lattice Algorithm. Such constraints include the degrees of redundancy and complexity at the different processing nodes (to be derived from the DFS algorithm of Andreadakis), the projected response time of the organization, ands some user-specified constraints on connections between decision making units. Develop a procedure for checking the validity of such constraints and incorporate them in the Lattice Algorithm. Generalize the approach to multilevel organizational structures and to variable structures, where variable structures are obtained by folding together different fixed structures. The real focus of the task is to introduce these additional constraints as a way of containing the dimensionality problem inherent in flexibility and reducing the coordination requirements.

Design a symbolic interface for the Lattice algorithm. The interface would have the capability of interpreting natural language inputs entered by the user and will include some symbolic processing. The system will generate the interconnections matrices used as input to the Lattice algorithm. The designer would then use the various tests described in the proposal (such as DFS algorithm) to check the validity of the interconnection constraints and to make required modifications.

Task 3: Information Dissemination

Semi-annual progress reports are submitted in place of annual reports. The results of this research will appear in thesis reports and in technical papers to be presented at professional meetings and published in archival journals. In addition, oral presentations will be given periodically as arranged with ONR.

3. RESEARCH PLAN

The research plan describes the strategy for meeting the program objectives. Specifically the research plan is organized around a series of specific well-defined research tasks that are appropriate for theses at master's and Ph.D. levels. Individual students are assigned to each task under the supervision of the principle investigator. Additional staff from the C3I Center are included in the project whenever there is a specific need for their expertise.

The focus of the task 1 is the development of a methodology for analyzing and verifying the set of decision rules used by an organization with distributed decision making. The methodology is based on the modeling of the decision rules in the form of Colored Petri Net and on the analysis of the net using S-invariant properties and Occurrence graphs. The results obtained for the two analyses, when applied to a specific form of decision rules, have been presented in the Ph. D. thesis of A. Zaidi that was submitted as the third semi-annual report. During this period, the methodology was implemented as part of the CAESAR II suite of tools for the generation, analysis, and evaluation of decision making organizations (DMOs).

Task 2 has been the focus of activity during this past six-month period. The various results (algorithms, tools, etc.) developed over the last few years were reimplemented and embedded in CAESAR II, the second version of the Computer Aided Evaluation of System Architectures suite of tools. Extensive use of COTS tools and modern software engineering techniques made it possible to enhance substantially the capabilities of the suite while reducing substantially the programming effort.

4. STATUS REPORT

A full description of CAESAR II incorporates in it a status report for all tasks under this project. Most of the results obtained under the project tasks thus far have been incorporated in the suite. There are additional results from past research efforts that need to be reviewed, re-expressed in the terminology and notation used in CAESAR II, coded and integrated into the suite through the design of appropriate interfaces. This is one of the objectives for the third year of performance.

4.1. CAESAR II

CAESAR II (<u>C</u>omputer <u>A</u>ided <u>E</u>valuation of <u>S</u>ystem <u>AR</u>chitectures) is a suite of software tools that puts together the results of more than 10 years of research in the design, modeling and evaluation of decision making organizations. Implemented in C, ML, and C++ on a Macintosh, it realizes interfaces between COTS applications (Design/CPN™, Design/IDEF™, Microsoft Excel™, MATLAB™), COTS development tools (Visual Architect™, AppleScript™) and "in-house" applications and algorithms (Lattice Algorithm, Cube Tool, RULER).

There are four major stages in CAESAR II:

- Requirements Generation,
- Decision Making Organization (DMO) Architecture Generation,
- Selection and Analysis of DMO Architecture
- Performance Evaluation

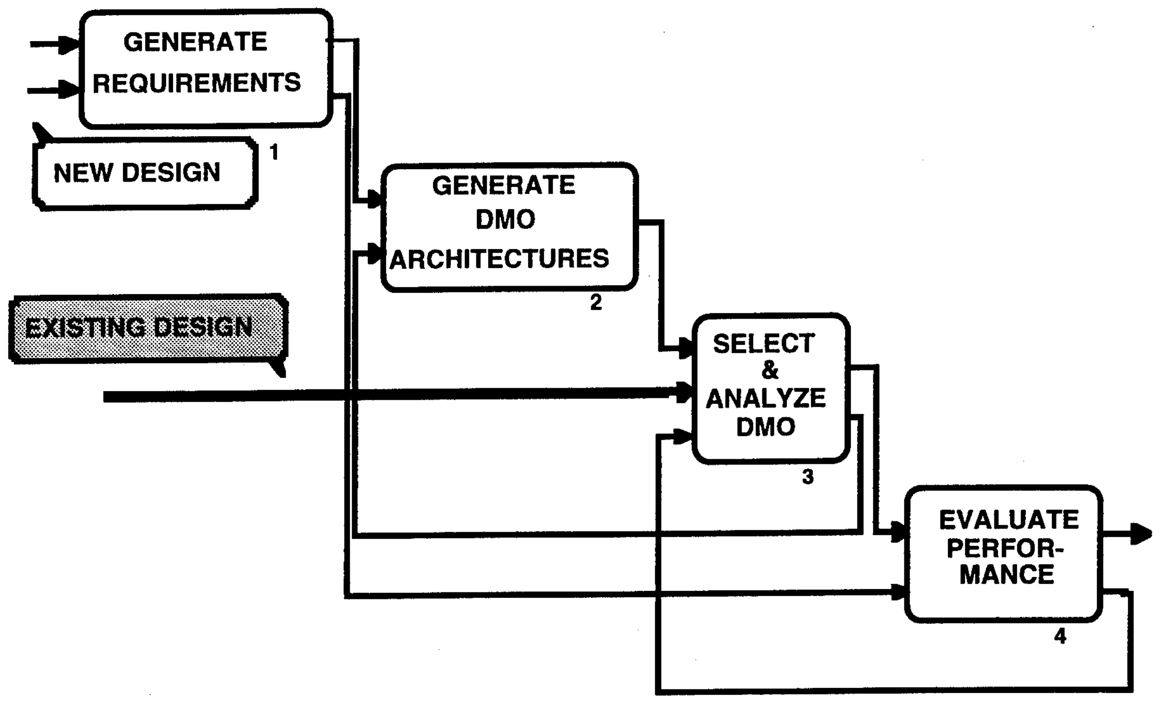


Figure 1 The Four Stages of CAESAR II

As shown in Figure 1, CAESAR II allows the design of new organizations or the consideration of existing designs. In this last case, only stages 3 and 4 are exercised.

Stage 1, Requirements Generation, is a heuristics process. From the definition of the mission to be performed by the organization and the operational concept that defines how the organization will perform the mission, the designer has to determine (1) the functionality (that is the decision process, the operations or methods to be used by the decision makers to perform this decision process, and the interactions among decision makers), (2) the structural constraints, (3) the performance requirements and, (4) the scenarios consistent with the operational concept that will be used in stage 4 for the derivation of performance measures.

The aim of the second stage is to generate a set of feasible decision making organizations that satisfy the structural constraints specified by the designer and exhibit the desired functionality. Organizations are represented as Petri Nets and the tools used in this stage rely on the mathematical properties of Petri Nets. Central to this stage is the Lattice algorithm (Remy, 1986; Demaël, 1989; Zaidi, 1991) that allows to determine all admissible organizational forms satisfying these constraints. Given a number of decision makers, the algorithm determines all the admissible organizational forms. The innovativeness of the approach is the way it addresses this combinatorial problem. Instead of characterizing every single solution, the complete solution is expressed in the form of a small set of minimally connected organizations

(MINOS), a small set of maximally connected organizations (MAXOS), and the lattice structure (the partially ordered set or Hasse diagram) of all intermediate solutions.

Two approaches can be used. In the first approach, depicted in Figure 2, the decision makers and their operations are given. One has to define the decision maker roles and the interactions among them, apply the Lattice algorithm, select the candidate design and then decompose the decision rules/tasks and assign them to roles. In the second approach, shown in Figure 3, the number of decision makers and their methods are derived from a given decision process. Their interactions are derived and the Lattice algorithm is then applied. Finally rules are assigned to roles.

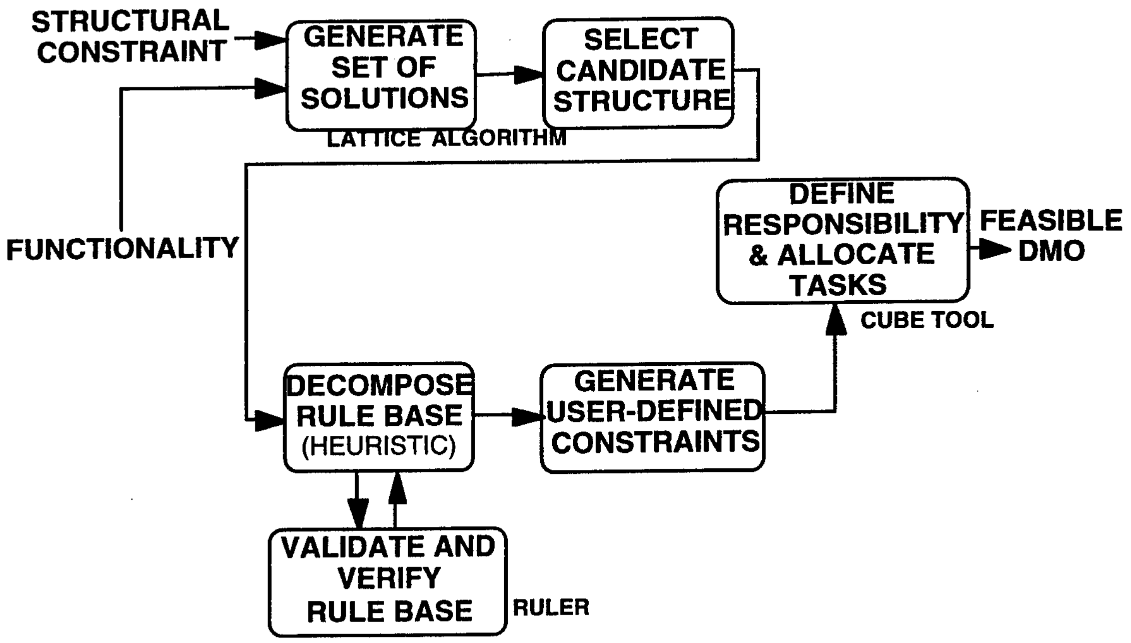

STRUCTURAL CONSTRAINT
FUNCTIONALITY
GENERATE SET OF SOLUTIONS
LATTICE ALGORITHM
SELECT CANDIDATE STRUCTURE
DEFINE RESPONSIBILITY & ALLOCATE TASKS
FEASIBLE DMO
CUBE TOOL
DECOMPOSE RULE BASE (HEURISTIC)
GENERATE USER-DEFINED CONSTRAINTS
VALIDATE AND VERIFY RULE BASE
RULER

Figure 2 From Structures to Rules

The organizations forms obtained from this approach correspond to the functional architecture. This functional architecture is then mapped to specific decision makers (human resources). Information sources and communication systems are added to the design (i.e., the physical architecture is used). Incorporation of the operational concept as a set of rules in the Petri Net completes the organizational design - the Operational Architecture is obtained.

The third stage, analyze and select DMO, leads to the reduction of the set of candidate organizations by doing a static evaluation. Different tools can be used at this stage. S-Invariants are used to analyze the functionality by identifying key threads from sources to sinks (Valraud, 1989). Deadlocks and traps (Jin, 1994) can be identified. They correspond to parts of the organization where problems can occur. Occurrence graph analysis allows the characterization all the states reachable from an initial state and the identification of undesirable ones. Temporal properties can be analyzed without any simulation by deriving the time equation of the net. Finally, for variable structure organizations, the coordination constraint can be checked to see whether each decision maker has the right combination of information to perform the desired function in the current mode of operations. It has to be noted that this set of tools can be used on any organization; for example, on an existing design not necessarily generated through the first two stages of CAESAR II.

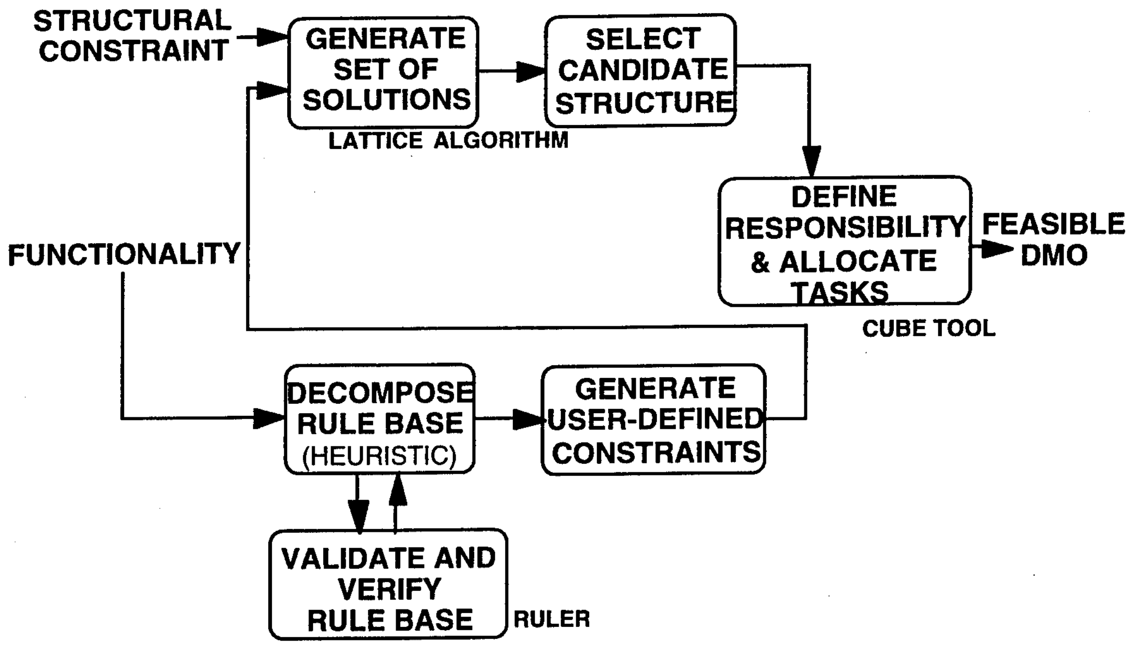

STRUCTURAL CONSTRAINT
GENERATE SET OF SOLUTIONS
SELECT CANDIDATE STRUCTURE
LATTICE ALGORITHM
DEFINE RESPONSIBILITY & ALLOCATE TASKS
FEASIBLE DMO
CUBE TOOL
FUNCTIONALITY
DECOMPOSE RULE BASE (HEURISTIC)
GENERATE USER-DEFINED CONSTRAINTS
VALIDATE AND VERIFY RULE BASE
RULER

Figure 3 From Rules to Structures

The last stage is to evaluate the performance of the organization. This is done through simulation of the Petri Net on a scenario defined in the first stage. Prior to the simulation, the parameters to be varied need to be defined and the net needs to be instrumented to collect data. To each setting of the parameters corresponds a given set of Measures of Performance attained by the organization. When varying the parameters and executing the Petri Net for each setting, the MOP space is derived. The Measures of Effectiveness are derived by comparing the MOPs to the performance requirements. They constitute the basis for the analysis for the final selection of the organization.

An example describing the overall approach is presented in Section 4.3. The next section describes in more detail the software development effort.

4.2. SOFTWARE IMPLEMENTATION

The software development effort focused on updating existing algorithms from previous research to take advantage of the new technologies available in computers and software systems engineering (intensive use of graphics, processing power, extended memory capacities, automatic code generation...). The aim was to develop a suite of software that interact in a seamless manner for the design and evaluation of decision making organizations. As much as possible, COTS software were used to decrease the programming effort and be consistent with best commercial practice.

As shown in section 4.1, the Lattice Algorithm is a key element of CAESAR II. This algorithm has been completely recoded in C. Interfaces to the program have been added to facilitate the input of constraints by the architecture designer and to display graphically the results. Previous versions of the lattice algorithm required that alphanumeric data be entered in the form of connection matrices; it generated results in the form of vectors that needed to be processed manually to obtain a graphical representation of the organizational structure.

Design/CPN™ is a key component of CAESAR II, not only because it is a Colored Petri net editor and simulator, but also because it offers a functional programming language (the ML language) that permits the use of graphical functions that can be used to generate Petri nets automatically or read any graphical structures drawn in the program (not necessarily Petri nets). As a result, the interfaces to the Lattice Algorithm have been implemented within the framework of Design/CPN. In addition, programs have been written in ML to read the structure of a Petri Net drawn in Design/CPN, to generate its incidence matrix and to compute its time delay equation. The incidence matrix of the Petri Net is the input to several algorithms used in the third stage of CAESAR II: S-invariants, deadlocks and traps, ... Finally, the RULER algorithm for the validation and verification of a rule base is supported by programs written in ML to transform a rule base expressed in First Order Predicate logic into a CPN to conduct different types of analyses.

Microsoft Excel™ and MATLAB™ are used in the fourth stage to conduct performance analysis. Excel is used to process and sort the data gathered from the simulation of the Colored Petri Net of the organization under investigation. MATLAB offers graphical capabilities to plot data sets in 3 dimensions. "M files" have been written to read data files created in Excel and draw Performance and Effectiveness Spaces. The different modules contained within CAESAR II will be described now in more detail.

4.2.1 Lattice Input

The purpose of the program, a sample window of which is presented in Figure 4, is to allow the architecture designer to specify graphically the connection constraints between the decision makers of the organization. The organization is drawn in the Design/CPN editor. To accomplish this, a palette that contains the different needed graphical objects can be used. By selecting an object in the palette, as many objects of this type as desired can be drawn in the workspace. The environment is represented by ellipses, decision makers by rounded boxes. Connections between decision makers and with the environment are represented by arrows drawn by selecting the "Connector" command in the Aux(iliary) Menu. The types of connections are specified by using the different labels contained in the palette: "inp" is for arrows connecting the input environment to decision makers and corresponding to connections from the environment to the Situation Assessment (SA) stage of the decision makers, "out" corresponds to the output produced by the decision makers (connection from the Response Selection (RS) stage of the decision makers to the environment); "asse" is used to model the exchange of assessments between decision makers and corresponds to the connections from the SA stage of a decision maker to the Information Fusion (IF) stage of another one. Processed responses exchanged between decision makers are represented by the label "resp" (connection from RS stage to IF stage). Commands issued by the decision makers are modeled using the label "cmd" (connection from RS stage to Command Interpretation (CI) stage). Finally, the connections from the RS stage to the SA stage are modeled using the label "ctrl". Once these labels have been inserted in the drawing, each of them needs to be defined as a region of the connectors they need to be attached to. This is done by selecting the label, choosing the Make Region item in the Aux Menu and clicking on the desired arrow. The drawing is completed when the designer specifies which links are optional and which ones are compulsory. Compulsory links are represented by thick arrows, optional ones by thin arrows. The architecture designer may select a group of arrows and then change their attributes by selecting the "Set Attributes" item in the Aux Menu and specifying the desired thickness of the lines. Once the drawing is complete, the program written in ML is launched by selecting the box at the bottom of the workspace and typing Command-;. This program reads the graphics and generates a text file that contains the connection matrices, input to the Lattice Algorithm.

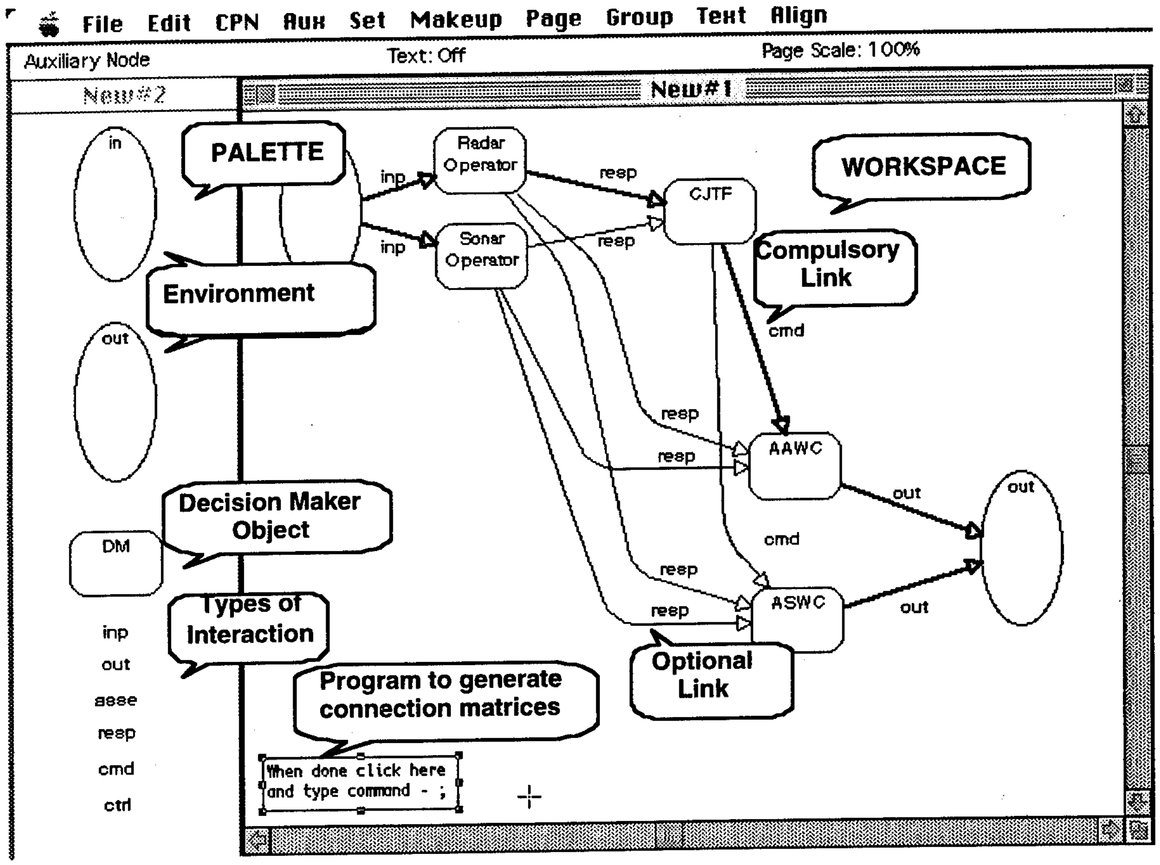


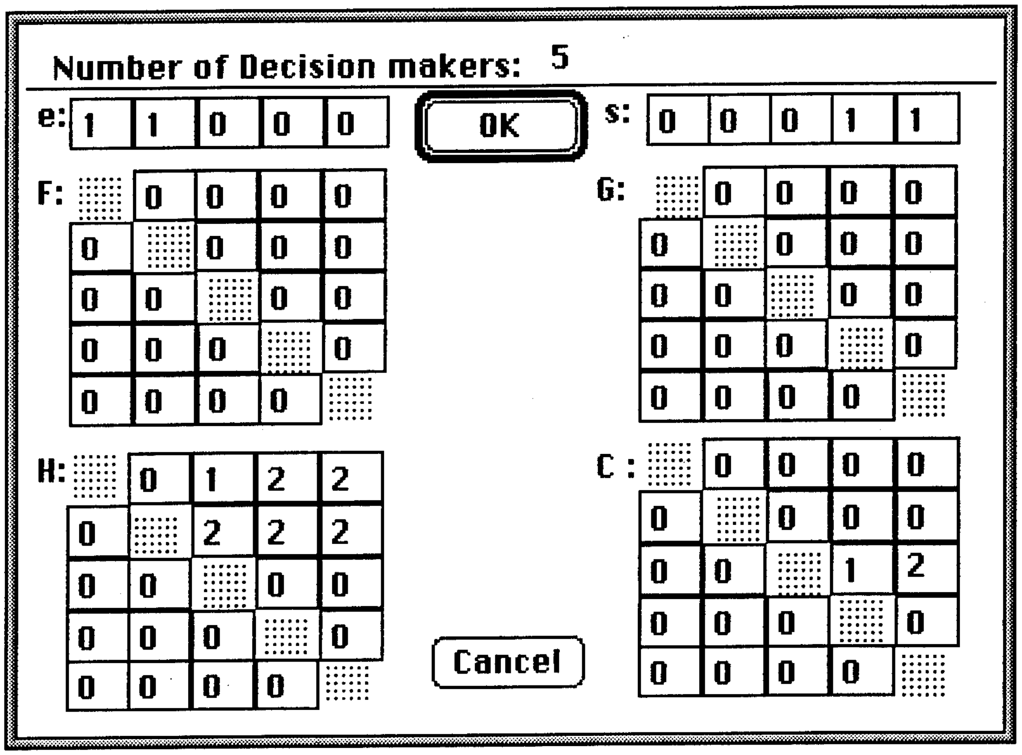
Figure 4 Lattice Input Specifications

4.2.2 The Lattice Algorithm

The lattice algorithm allows the automatic generation of candidate architectures satisfying a set of connection constraints and of structural constraints. The candidate architectures (which are numerous) are not listed singly but gathered in lattices defined by their largest and their smallest element: the maximally and minimally connected architectures (MINOs and MAXOs). Each candidate architecture belongs to one of the lattices and can be defined by addition of simple paths to the MINO or the subtraction of simple paths from the MAXO. The initial work by Pascal Remy (1986) was limited to five decision making units because of hardware and software (operating system) constraints. Advances in computer hardware and software have weakened this limitations; larger organizations can be designed. More recent work by Abbas Zaidi (1991) has overcome the theoretical limitations by introducing a layered representation of the architecture. While the current interfaces anticipate the inclusion of hierarchically multi-layered organizations, the algorithms that support the design have not been implemented yet in CAESAR II.

This implementation of the Lattice Algorithm takes into account additional user-defined constraints that reduce the number of solutions by taking into consideration additional requirements: temporal constraints, degrees of redundancy at processing nodes (SA and RS stages) and degrees of complexity at fusion nodes (IF and CI stages), and user-defined structural constraints expressed in logic terms. These constraints were previously considered later in the design.

As mentioned earlier, the input of the lattice algorithm is a set of connection constraints represented by matrices: **e** (input to the organization), **s** (output of the organization), **F** (SA to IF stages), **G** (RS to SA stages), **H** (RS to IF stages), and **C** (RS to CI stages). Each cell of the different matrices contains either "1" if the corresponding connection exists, "0" if it does not, or "2" if it is optional. These matrices are generated automatically by the program "Lattice Input" or can be filled or modified manually in the window displayed in Figure 5.



Number of Decision makers: 5

e:

1	1	0	0	0

s:

0	0	0	1	1

F:

▨	0	0	0	0
0	▨	0	0	0
0	0	▨	0	0
0	0	0	▨	0
0	0	0	0	▨

G:

▨	0	0	0	0
0	▨	0	0	0
0	0	▨	0	0
0	0	0	▨	0
0	0	0	0	▨

H:

▨	0	1	2	2
0	▨	2	2	2
0	0	▨	0	0
0	0	0	▨	0
0	0	0	0	▨

C :

▨	0	0	0	0
0	▨	0	0	0
0	0	▨	1	2
0	0	0	▨	0
0	0	0	0	▨

Figure 5 Connection Matrices

The flow chart of the Lattice Algorithm is displayed in Figure 6. From the interaction constraints, the algorithm generates the incidence matrices of two Petri nets: the Kernel Net corresponding to the structure deduced from the connection matrices where all the optional links are considered inactive, and the Universal Net corresponding to the structures where all optional links are considered active. At the same time, a specific label is assigned to every place and the fixed places that have to be included in every solution structure are determined.

In a second stage, the algorithm computes the S-invariants of the Universal Net and stores those that contain the source place and the sink place (the simple paths) and those that correspond to circuits (loops). The Lattice Algorithm uses a specific labeling scheme for each place and for each transition as shown on Figure 7. The computation of the S-invariants results in the generation of row vectors [1xm]. Each entry corresponds to a place of the Universal Net and takes value 1 if the place is included in the S-invariant, 0 if it is not. This vector representation is also used later when generating MINOs and MAXOs for structures obtained by adding (joining) simple paths. The join operation is noted by $\cup$.

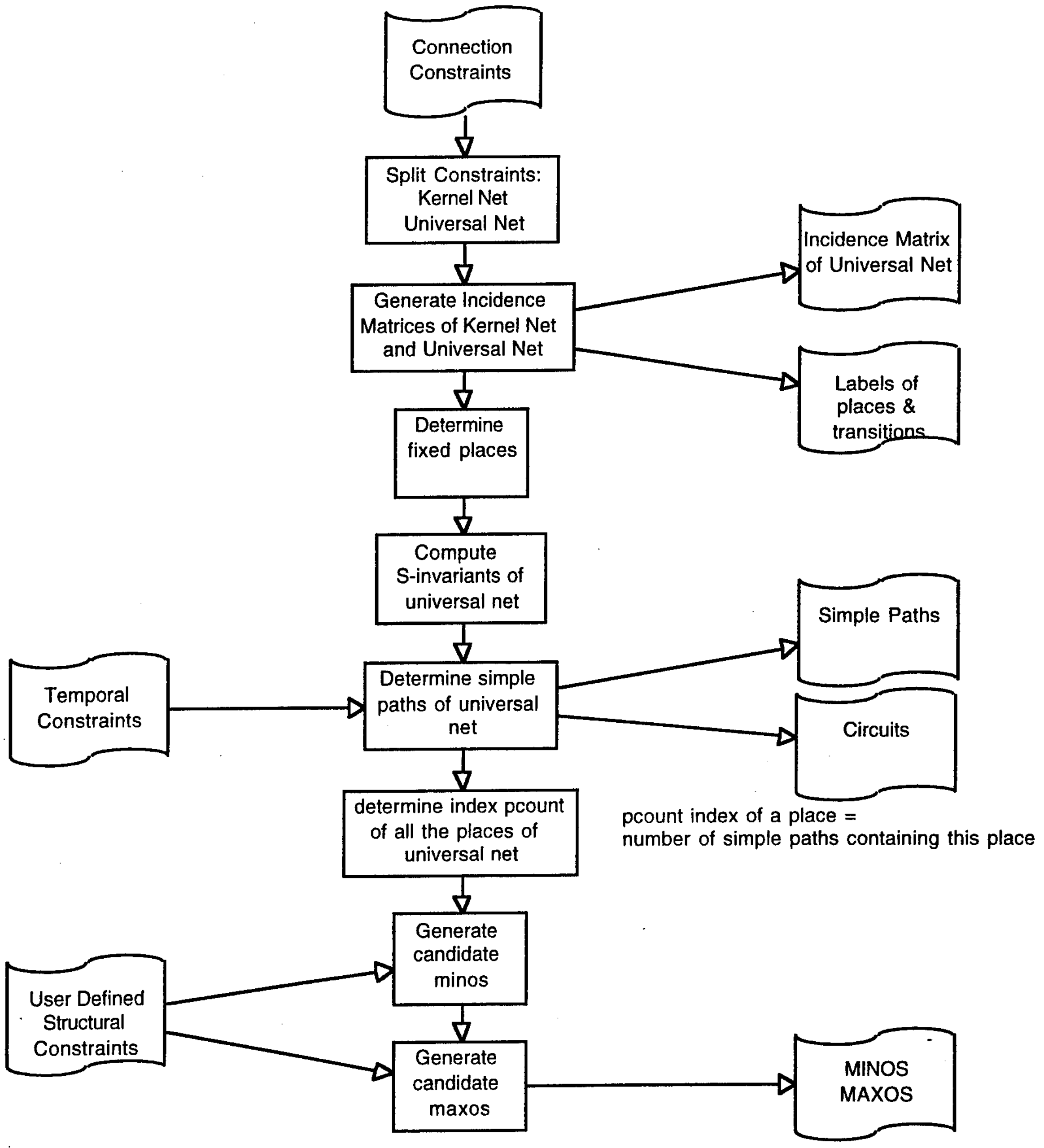

Connection Constraints
Split Constraints: Kernel Net Universal Net
Generate Incidence Matrices of Kernel Net and Universal Net
Incidence Matrix of Universal Net
Labels of places & transitions
Determine fixed places
Compute S-invariants of universal net
Temporal Constraints
Determine simple paths of universal net
Simple Paths
Circuits
determine index pcount of all the places of universal net
pcount index of a place = number of simple paths containing this place
User Defined Structural Constraints
Generate candidate minos
Generate candidate maxos
MINOS MAXOS

Figure 6 Flow Chart of the Lattice Algorithm

$$[X_1 \ldots X_i \ldots X_n] \cup [Y_1 \ldots Y_i \ldots Y_n] = [(X_1 \text{ or } Y_1), \ldots (X_i \text{ or } Y_i) \ldots (X_n \text{ or } Y_n)]$$

where $X_i, Y_i \in \{0,1\}$ and X_i or $Y_i = 0$ if $X_i = 0$ and $Y_i = 0$
X_i or $Y_i = 1$ otherwise.

Thus, a structure is also represented by a vector, each entry corresponding to a place of the Universal Net and taking value 1 if the place is included in the structure, 0 if it is not.

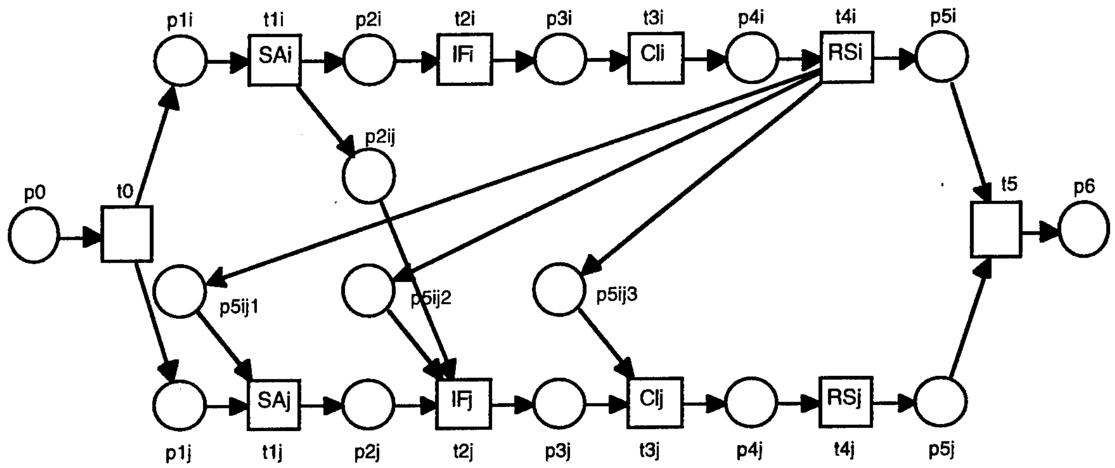


Figure 7 Labeling of Places and Transitions in the Lattice Algorithm

In the third stage, MINOs are generated by adding simple paths stored in the second stage to the Kernel Net. The number of simple paths that can be added can be limited by considering the temporal constraints. The user can associate a delay with each transition that corresponds to the amount of time required to perform the process it represents and specify a requirement for the response time in the dialog window displayed in Figure 8. Since the Petri Nets handled by the Lattice Algorithm are acyclical marked graphs, the response time of the system is equal to the largest sum of the delays of the transitions of the S-component of the S-invariants of the Petri Net. The delay associated with each Simple Path is then computed and those whose delay is larger than the required response time are discarded from the set of simple paths that can be added.

Simple Paths are added to the kernel net until the different constraints are satisfied. Five basic structural constraints have been defined:

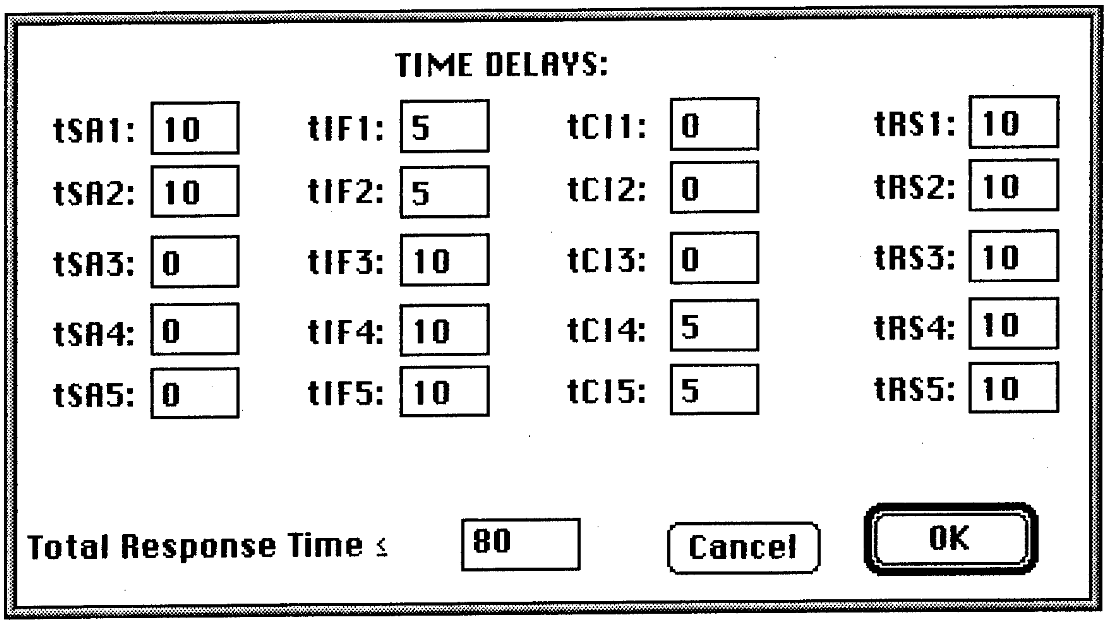
R1 The structure should be connected

R2 The structure must be acyclical (no loop)

R3 There exists at most one link from the RS stage of DM_i to the SA, IF, CI stage of DM_j: $G_{ij} + H_{ij} + C_{ij} \leq 1$

R4 Information Fusion takes place only at IF and CI stages. The SA stage has at most one input with preference to external input: $e_j + G_{ij} \leq 1$

R5 There should not be simultaneously a connection from the SA stage of DM_i to the IF stage of DM_j and a connection from the RS stage of DM_i to the SA stage of DM_j: $F_{ij} + G_{ij} \leq 1$


TIME DELAYS:

tSA1:	10	tIF1:	5	tCI1:	0	tRS1:	10
tSA2:	10	tIF2:	5	tCI2:	0	tRS2:	10
tSA3:	0	tIF3:	10	tCI3:	0	tRS3:	10
tSA4:	0	tIF4:	10	tCI4:	5	tRS4:	10
tSA5:	0	tIF5:	10	tCI5:	5	tRS5:	10

Total Response Time ≤ 80

Figure 8 Specification of Time Delays and Total Response Time

In addition, the algorithm checks for user-defined structural constraints: degrees of redundancy and of complexity and logical relations

Degrees of redundancy and of complexity
Redundancy and Complexity are two aspects of importance in the design of an organization's information architecture. Redundancy is related to the dissemination of information in the organization for back-up purposes. While redundancy is desirable for survivability and reliability, it is limited by the communications network capacity. Complexity addresses the problem of fusion of information from different sources. The more sources, the more complex the fusion process and the more resource consuming the process is. The architecture designer has to deal with these two parameters and perform some tradeoffs. As more redundancy is introduced, the fusion algorithms will be more complex or more resources will be needed. Pascal Remy in his Master's thesis (1986) gives an interpretation of MINOs and MAXOs. MINOs represent organization with a minimal number of interactions between DMs. They represent not very survivable architectures because there is no redundancy to disseminate received or generated information throughout the system. However, they represent timely organizations. On the other hand, MAXOs represent architectures with a lot of redundancy but that are less timely. An interesting feature for the Lattice Algorithm would be to take into account redundancy and complexity requirements in the generation process of feasible structures. Stamos Andreadakis (1988) describes an alternative methodology to generate architectures. After defining basic Information Flow Paths, he introduces the concepts of

degree of Complexity at Fusion nodes and degree of Redundancy at Processing nodes. Redundancy is related to the number of Information Flow Paths that receive data generated from a Processing node while Complexity is related to the number of Information Flow Paths that send data to a Fusion Node. The definition of these degrees of redundancy and complexity is the basis for the generation of a set Data Flow Structures from which architectures can be derived by allocating functions to decision makers.

The concept of complexity and redundancy can be applied to the four stage model of Decision Making Unit. Situation Assessment and Response Selection stages can be considered as Processing nodes while Information Fusion and Command Interpretation stages are Fusion nodes. We can define SAR_i (resp. RSR_i) as the degree of redundancy of the SA (resp. RS) stage of DM_i. SAR_i (resp. RSR_i) is equal to the number of output places of the transition representing the SA (resp. RS) stage of DM_i. We can also define IFC_j (resp. CIC_j) as the degree of complexity of the IF (resp. CI) stage of DM_j. IFC_j (resp. CICj) is equal to the number of input places of the transition representing the IF (resp. CI) stage of DM_j. If n is the number of DMs, using the connection matrices of the Lattice Algorithm and the vector representation of a structure s, these degrees can expressed as:

$$SAR_i(s) = \begin{cases} 1+\sum_{j=1}^{n} F_{ij} & \text{if stage IFi exists} \\ \sum_{j=1}^{n} F_{ij} & \text{if stage IFi does not exist} \end{cases} = P_{2i}(s) + \sum_{\substack{j=1 \\ j\neq i}}^{n} P_{2ij}(s)$$

$$RSR_i(s) = s_i + \sum_{j=1}^{n} G_{ij} + H_{ij} + C_{ij} = P_{5i}(s) + \sum_{\substack{j=1 \\ j\neq i}}^{n} P_{5ij1}(s) + P_{5ij2}(s) + P_{5ij3}(s)$$

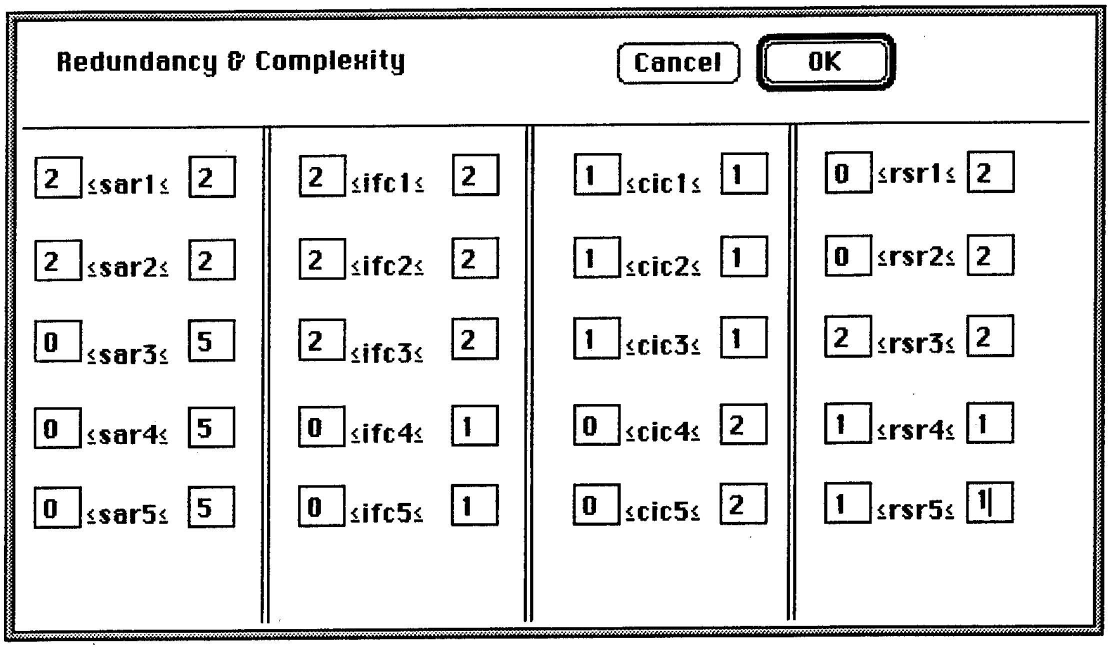
$$IFC_j(s) = \begin{cases} 1+\sum_{i=1}^{n} F_{ij} + H_{ij} & \text{if stage SAj exists} \\ \sum_{i=1}^{n} F_{ij} + H_{ij} & \text{if stage SAj does not exist} \end{cases} = P_{2j}(s) + \sum_{\substack{i=1 \\ i\neq j}}^{n} P_{2ij}(s) + P_{5ij2}(s)$$

$$CIC_j(s) = \begin{cases} 1+\sum_{i=1}^{n} C_{ij} & \text{if stage IFj exists} \\ \sum_{i=1}^{n} C_{ij} & \text{if stage IFj does not exist} \end{cases} = P_{3j}(s) + \sum_{\substack{i=1 \\ i\neq j}}^{n} P_{5ij3}(s)$$

The architecture designer can specify ranges for the different degrees of redundancy and complexity for the architecture he envisions in the window displayed in Figure 9. These ranges introduce additional constraints that the Lattice algorithm checks the same way as it checks the other constraints

User-Defined Rules
User-defined rules include different heuristics used by the architecture designer to describe characteristics of the envisioned architecture. Some of them translate directly into setting some cells of the connection matrices to 1 or 0. The new version of the Lattice Algorithm accepts

more complex logical rules of the kind: "if interactions X_{ij} and Y_{jk} exist, then the interaction Z_{lm} exists" or "either link X_{ij} or Y_{jk} exists". An approach has been developed to use techniques from Integer Programming (IP) to translate conditions and logical relations on interactions into constraints that can be checked by the Algorithm.



2 ≤sar1≤ 2	2 ≤ifc1≤ 2	1 ≤cic1≤ 1	0 ≤rsr1≤ 2
2 ≤sar2≤ 2	2 ≤ifc2≤ 2	1 ≤cic2≤ 1	0 ≤rsr2≤ 2
0 ≤sar3≤ 5	2 ≤ifc3≤ 2	1 ≤cic3≤ 1	2 ≤rsr3≤ 2
0 ≤sar4≤ 5	0 ≤ifc4≤ 1	0 ≤cic4≤ 2	1 ≤rsr4≤ 1
0 ≤sar5≤ 5	0 ≤ifc5≤ 1	0 ≤cic5≤ 2	1 ≤rsr5≤ 1

Figure 9 Specifications of the Degrees of Redundancy and Complexity

In what follows, capital letters (A, B, ...) denote cells of the connection matrices e, s, F, G, H or C. A indicates that the connection A must exist (A=1). $\neg A$ indicates that the connection A must not exist (A=0). User defined constraints can then be expressed in terms of logical formulae that use the operators $\neg$ (not), $\wedge$ (and), $\vee$ (or), and $\rightarrow$ (implies / if-then). Any logical formula can be transformed into a Conjunctive Normal Form where the formula is expressed as the conjunction (operator $\wedge$) of disjunctions (operator $\vee$). For example $(A \vee B) \wedge (\neg A \vee C) \wedge (B \vee \neg D \vee E)$ is a conjunctive normal form because expressions containing only operators $\vee$ are connected with operators $\wedge$. Each of the disjunctions of the resulting expression can then be processed independently. A disjunction contains a finite number (m) of positive literals and a finite number (n) of negative literals. Its generic form is:

$$\neg A_1 \vee \neg A_2 \vee \dots \vee \neg A_n \vee A_{n+1} \vee A_{n+2} \vee \dots \vee A_{n+m}$$

which can be rewritten:

$$(A_1 \wedge A_2 \wedge \dots \wedge A_n) \rightarrow (A_{n+1} \vee A_{n+2} \vee \dots \vee A_{n+m})$$

Any user defined constraints can be expressed in terms of rules of this type. In previous semi annual reports, we used an Integer Programming formulation to derive an equation that can be included in the set of constraints checked by the Lattice Algorithm:

$$A_1 + A_2 + \dots + A_n - n+1 \leq A_{n+1} + A_{n+2} + \dots + A_{n+m}$$

If $A_1 \wedge A_2 \wedge \ldots \wedge A_n$ is true then Ai = 1 for all i and the equation becomes :

$$1 \leq A_{n+1} + A_{n+2} + \ldots + A_{n+m},$$

requiring that at least one of the $A_{n+j} = 1$.

If $A_1 \wedge A_2 \wedge \ldots \wedge A_n$ is false then there exists i in {1, ..., n} such that $A_i = 0$ and therefore:

$$A_1 + A_2 + \ldots + A_n - n+1 \leq 0 \leq A_{n+1} + A_{n+2} + \ldots + A_{n+m}$$

and the A_{n+j} can take any values.

Constraints Checking
The constraints are checked on the vector representation of the current structures. Constraint R1 is true by construction. The addition of simple paths from the source place to the sink place ensures that there exists a non directed path from any node to any other node. Constraint R2 is checked by verifying that any of the circuits is not a subnet of the current structure. The same approach is used to ensure that any simple path that violates the time requirements is not contained in the current structure because the addition of several allowable simple paths can result in the presence of an undesirable simple path. The basic constraints R3, R4 and R5, the constraints for redundancy and complexity and the constraints derived from user-defined logical statements are all expressed in terms of inequalities. These constraints are checked in identical manner by applying the inequalities on the appropriate entries of the vector representation of the current structure.

The flow chart for the generation of MINOs is displayed on Figure 10.

A depth first strategy is used to add simple paths. At each stage, the fixed place not included in the current structure and which is contained in the fewest number of simple paths is identified. The first simple path is added and the process continues until a MINO is found, i.e., all the constraints are satisfied and the structure contains all the fixed places. Before saving the MINO, the algorithm checks whether the new MINO is equal to a previously found MINO. The algorithm backtracks and adds the next simple path to the previous structure.

In a fourth stage, MAXOs are generated by removing simple paths from the Universal Net until none of the constraints are violated. The approach used is similar to the one used to generate the MINOs, the only difference being that the algorithm identifies the non-fixed places that are the reasons for the violation of constraints and that lead to the simple path that should be remove.

The output of the Lattice Algorithm is a text file that can be read by the program "Solution Display and Candidate Selection" described next.

4.2.3 Solution Display and Candidate Selection

The aim of this program is to display graphically the solution generated by the Lattice Algorithm and to construct the candidate solutions. Implemented in the framework of Design/CPN, it consists of five different modules written in ML that can be launched by selecting the appropriate module and typing "command - ;".

The first module reads the output file of the Lattice Algorithm and the file containing the invariants of the universal net. It reads the different MINOs and MAXOs, determines which MINOs are subnets of the different MAXOs and draws on a new page, as shown on Figure 11, on two different lines circle nodes that represent the MINOS and the MAXOs and connects the nodes with an arrow to indicate that the corresponding MINO is a subnet of the corresponding MAXO. Each MINO-MAXO pair connected by an arrow corresponds to the boundaries of the Hasse diagram bounded by the MINO and the MAXO.

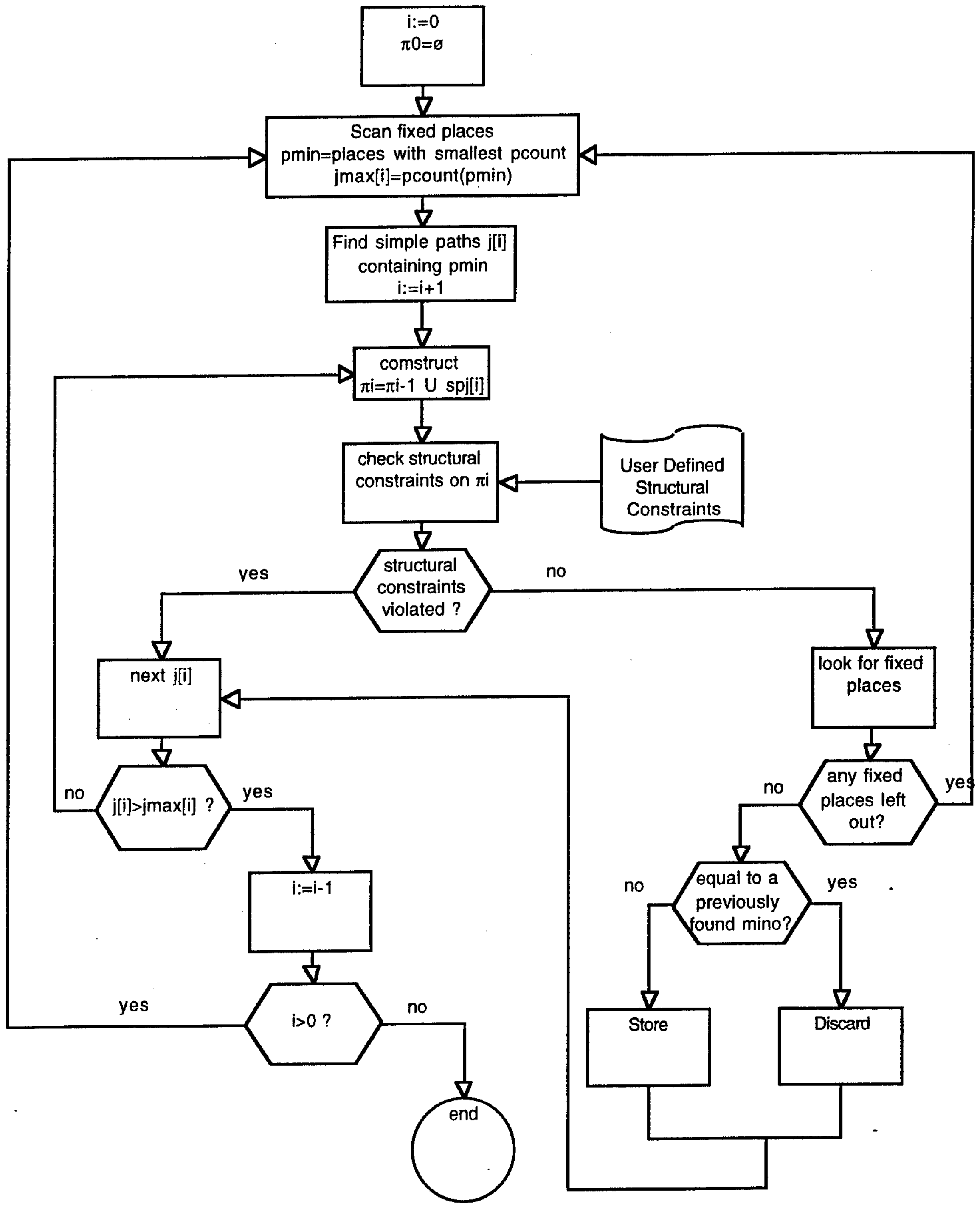
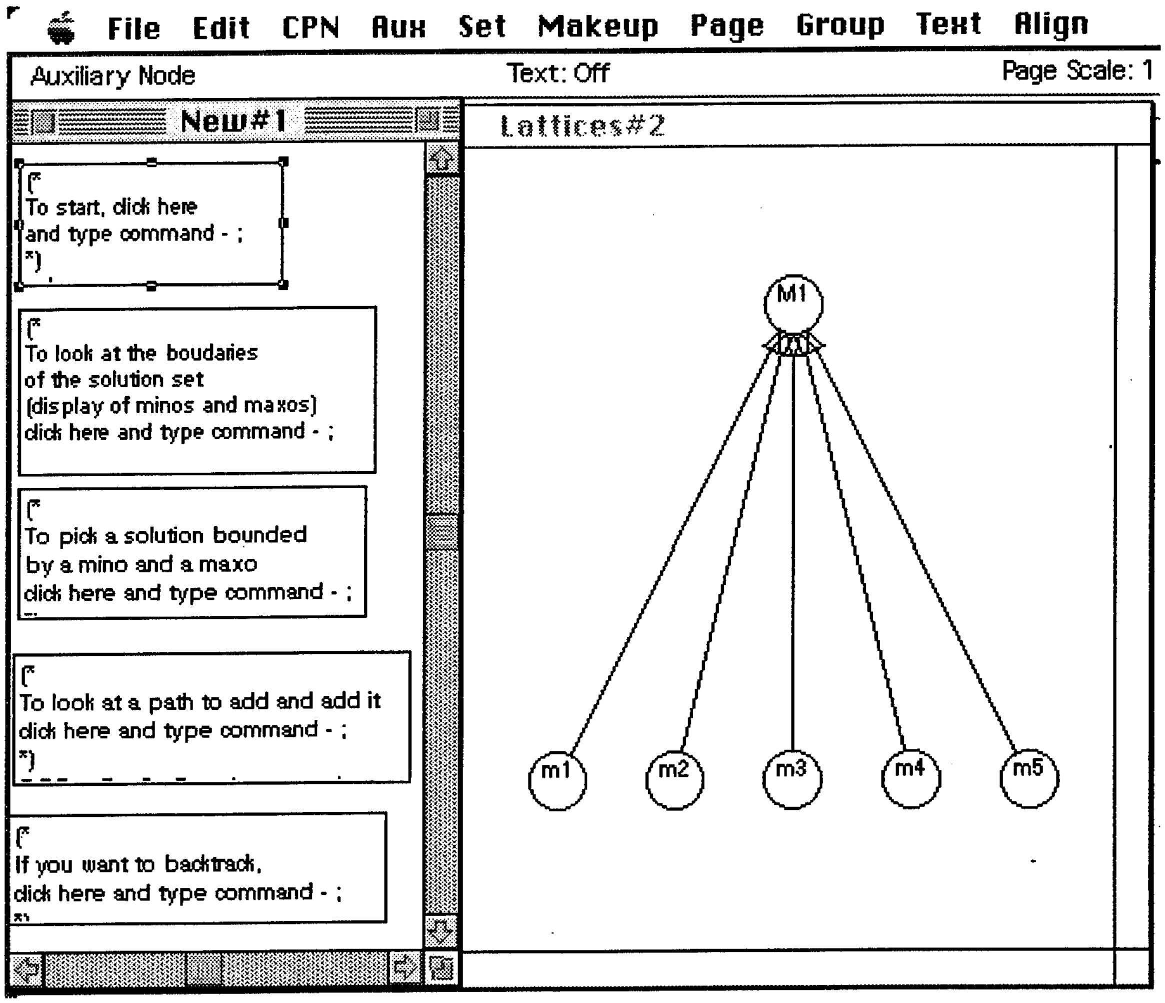


Figure 10 Flow Chart of MINOs Generation

The second module draws the Petri Nets of the MINOs and MAXOs. When launched, it displays the page drawn with the first module and asks the user to click on the node corresponding to MINO or MAXO he wants. The module then creates a new page and draws automatically the Petri Net of the selected MINO or MAXO. When the drawing is completed, the user is asked whether he wants to have another net drawn or not. Answering "no" leaves the module, while answering "yes" starts the process all over again. If the selected MINO or MAXO has already been drawn, the page containing the net is displayed. The generated Petri Nets are ready for simulation or can be edited with the Design/CPN Editor. To simulate one of the generated Petri Net, the user needs only to specify the corresponding page as prime by using the "Mode Attributes" command in the "Set" menu in *Design/CPN* and to switch to the simulator.

The three last modules construct a candidate structure bounded by a MINO and a MAXO. When the third module is launched, the user is asked to select a pair MINO-MAXO in which the MINO is a subnet of the MAXO. A new page is created and the MAXO is drawn with thin lines. The places, transitions, and interconnecting arcs of the MINO are then displayed with thick lines. This drawing constitutes the workspace on which simple paths will be added to design the candidate solution, as shown on Figure 12.



(*
To start, click here
and type command - ;
*)

(*
To look at the boudaries
of the solution set
(display of minos and maxos)
click here and type command - ;

(*
To pick a solution bounded
by a mino and a maxo
click here and type command - ;

(*
To look at a path to add and add it
click here and type command - ;
*)

(*
If you want to backtrack,
click here and type command - ;
*)

Figure 11 Display of Solutions

The fourth module determines which simple paths need to be added to the MINO to get to the MAXO. The user is informed of the number of simple paths that can be added and asked to select one of them. The selected simple path is then displayed on the net in bold gray and the

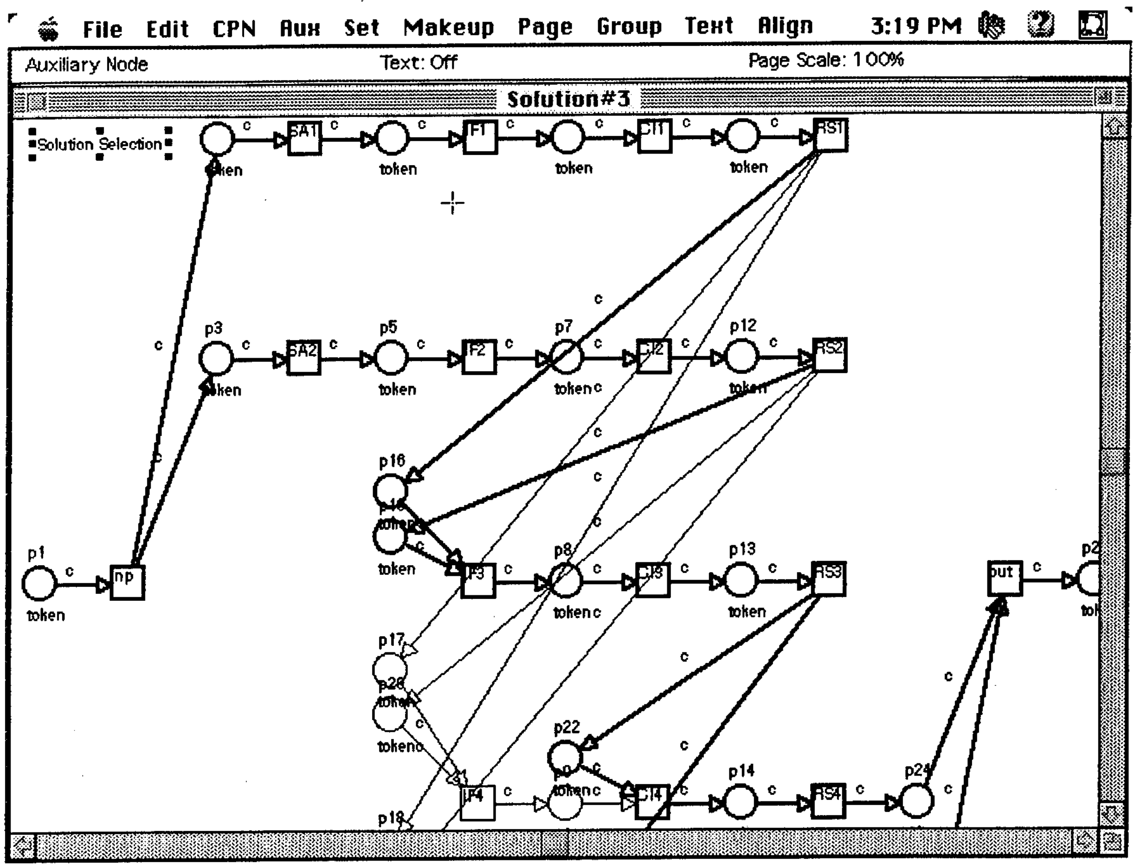


Figure 12 Workspace for Candidate Selection
(the command window has been hidden)

user is asked whether he wants this path to be added. If the answer is yes, the resulting net is displayed in black thick lines. If the answer is no, the drawing returns to its previous stage. A new simple path can then be added by launching again the fourth module. A solution is thus constructed by using these modules repetitively to visualize the different simple paths that can be added and add only the desired ones. At any time during the design process, the user can backtrack to a previous solution by launching the fifth module. The program keeps in memory the intermediate solutions. When the candidate solution has been determined, the user can delete the nodes that are represented in thin lines. The resulting Petri Net is ready for simulation.

4.2.4 RULER

A methodology for the validation and verification (V&V) of decision making rules has been proposed and reported in previous progress reports. The approach taken in RULER (RULe Evaluation Routine) is based on viewing a rule base as an organization of information that flows from one process (rule) to another. Since Petri Nets provide a powerful modeling and analysis tool for information flow structures, the methodology transforms a set of decision rules into an equivalent Petri Net representation. The static and dynamic properties of the graph are shown to reveal patterns of Petri Net structures that correspond to the problematic cases. For a detailed description of the algorithms and techniques used in RULER, see Zaidi (1994).

Figure 13 represents the three main modules (shown as boxes in the figure) of RULER. The figure also shows all the inputs and outputs of these processes. Next to each are shown all the algorithms and techniques that constitute the module.

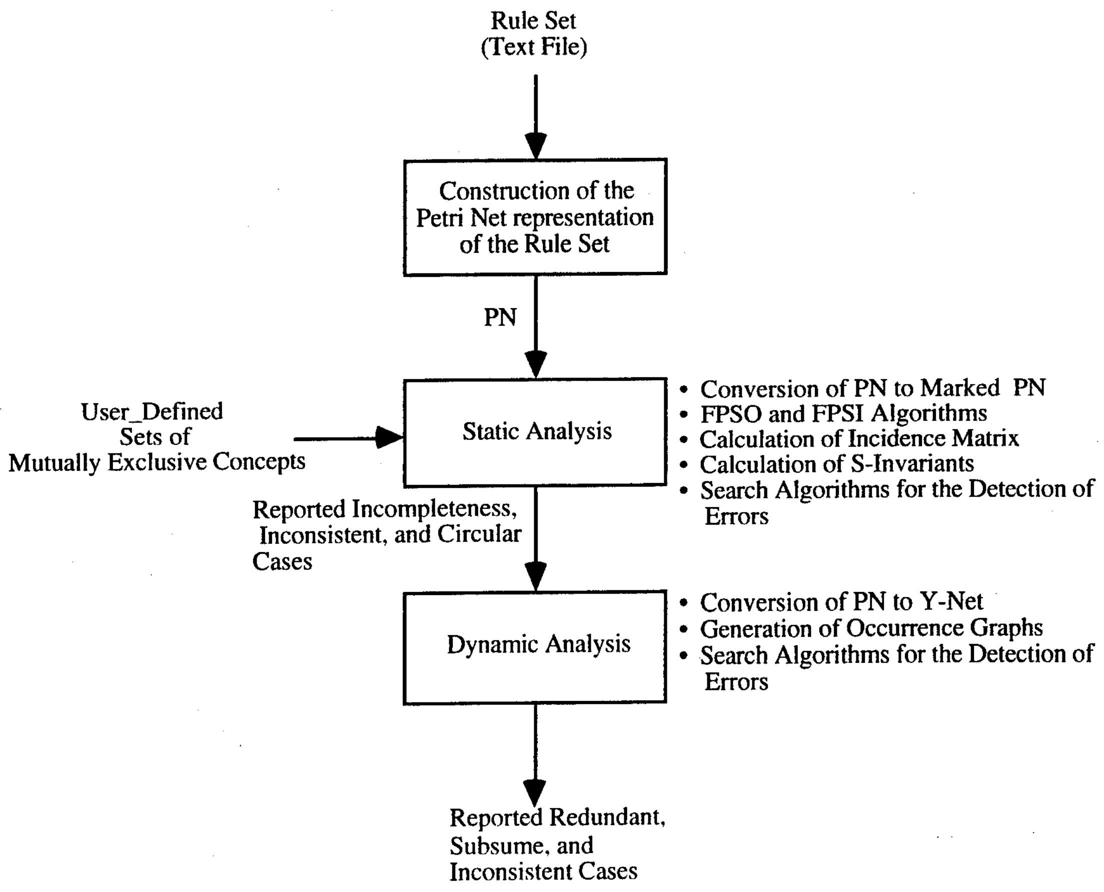

Rule Set
(Text File)
Construction of the Petri Net representation of the Rule Set
PN
User_Defined Sets of Mutually Exclusive Concepts
Static Analysis
• Conversion of PN to Marked PN
• FPSO and FPSI Algorithms
• Calculation of Incidence Matrix
• Calculation of S-Invariants
• Search Algorithms for the Detection of Errors
Reported Incompleteness, Inconsistent, and Circular Cases
Dynamic Analysis
• Conversion of PN to Y-Net
• Generation of Occurrence Graphs
• Search Algorithms for the Detection of Errors
Reported Redundant, Subsume, and Inconsistent Cases

Figure 13 Main Modules of RULER

Module 1: Construction of the Petri Net Representation

This module has been implemented in *Design CPN*™ in its entirety. The program requires an input text file that can be imported in *Design CPN*™ using its 'Load Text' command. The program can be used to construct Petri Nets for both Propositional and First-Order Predicate systems.

Module 2: Static Analysis

The following algorithms have been implemented in *Design CPN*™:

- Conversion of PN to Marked Graph PN
- FPSO and FPSI Algorithms
- Construction of Incidence Matrix

The same S-Invariant algorithm implemented for the Lattice algorithm is used to calculate the S-Invariants, therefore, RULER does not have its own implementation of this process.

The remaining algorithms and analysis tools have not been implemented yet and, therefore, are carried out manually.Their implementation is planned during the next few months.

Module 2: Dynamic Analysis

This module has yet to be implemented. The dynamic analysis presently is carried out by first instrumenting the PN manually and then the occurrence graph is automatically constructed by the*OG Analyser*™. The erroneous states are searched with the help of commands provided in the software.

These different programs have been used in the illustrative example described in the next section.

4.3. AN EXAMPLE

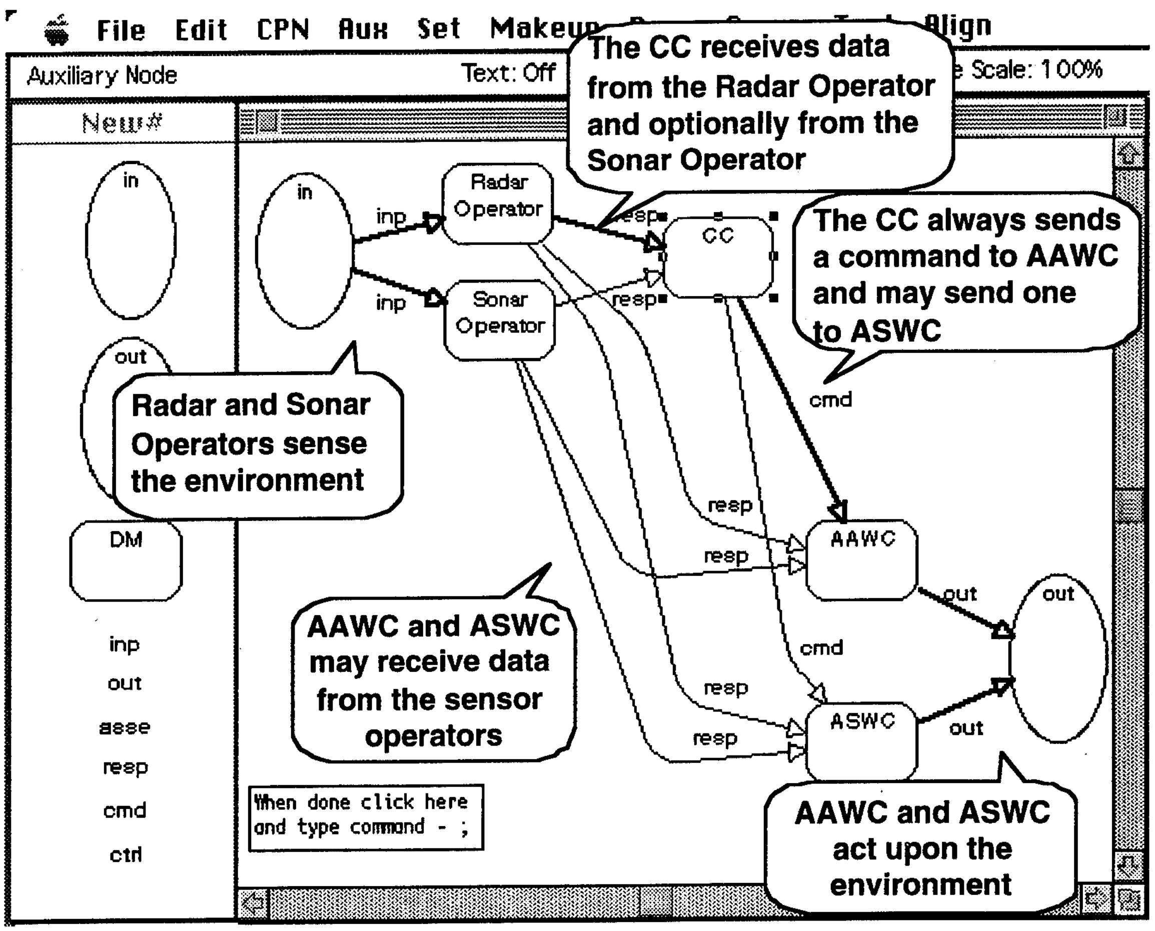
The CAESAR II approach is illustrated through an example: the design of a Navy Tactical Command and Control Center. This example is a simplified version of the example of Remy and Levis (1988).

4.3.1 Description

The organization under consideration faces air, surface and underwater threats. It has 5 decision makers: DM_1 and DM_2 act as the sensors of the organization (Sonar Operator (SO) and Radar Operator (RO), for example). They both receive information from the external environment (threat detection). DM_1 can detect air and surface threats, while DM_2 can detect surface and underwater threats. DM_1 and DM_2 do not share their assessment. DM_1 has to send this information to DM_3 who acts as the Commander in Chief (CC). DM_2 is not obliged to send his response to DM_3. Finally, DM_4, the Anti-Air Warfare Commander (AAWC) and DM_5, the Anti-Submarine Warfare Commander (ASWC) produce the organization's response (firing of missiles for AAWC or torpedoes and depth charges for ASWC). DM_4 deals with air and surface threats. DM_5 aims at underwater and surface threats. They receive orders from the coordinator DM_3 and may receive information from DM_1 and DM_2. They may also share their results. One can see that DM_1 and DM_2 on one side and DM_4 and DM_5 on the other have overlapping area of responsibilities. The role of DM_3 as coordinator is therefore critical for the correct execution of the mission.

4.3.2 Architecture Generation and Solution Selection

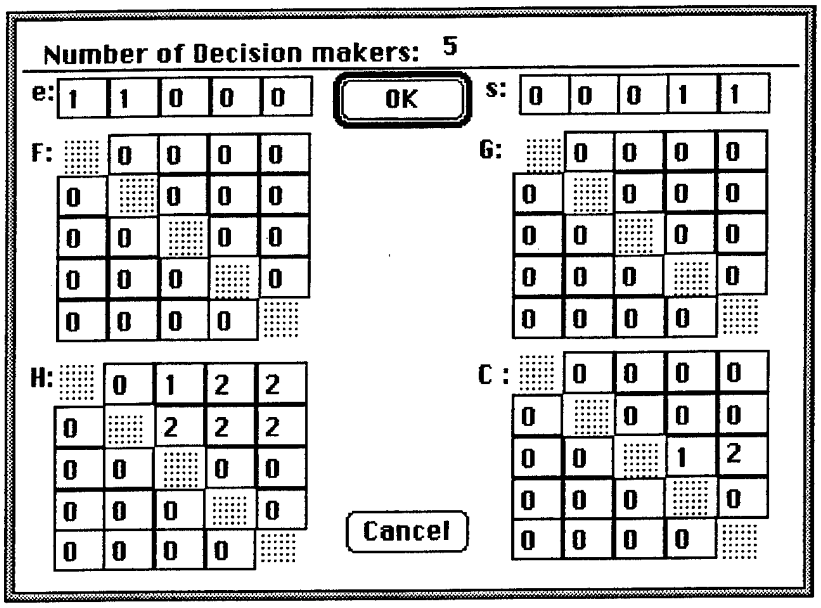
The graphical representation of this description of the organization, as done using the program "Lattice Input," is displayed in Figure 14. The five decision makers are represented as rounded boxes. The connections between them and with the environment have been represented by arrows. The appropriate labels (asse, cmd or ctrl) have been associated with the arcs. Finally, the thickness of these arrows determine if the link is optional or compulsory. For example, the requirement that the CC has to send a command to the AAWC is represented by a thick arrow from the rounded box representing the CC to the rounded box representing the AAWC with the label "cmd" attached to it.



File Edit CPN Aux Set Makeup Align
Auxiliary Node
Text: Off
Scale: 100%
New#
in
out
DM
inp
out
asse
resp
cmd
ctrl
in
inp
inp
Radar Operator
Sonar Operator
CC
resp
resp
cmd
resp
resp
AAWC
cmd
resp
resp
ASWC
out
out
out
The CC receives data from the Radar Operator and optionally from the Sonar Operator
The CC always sends a command to AAWC and may send one to ASWC
Radar and Sonar Operators sense the environment
AAWC and ASWC may receive data from the sensor operators
AAWC and ASWC act upon the environment
When done click here and type command - ;

Figure 14 Connection Specifications for the Example

The connection matrices generated by the program are shown in Figure 15. All the optional links result in "2's " being placed in the corresponding cells of the matrices. The compulsory links are specified by "1's" in the appropriate cells. Links that should not exist correspond to the "0's". The specifications result in the matrices F (SA->IF) and G (RS->SA) whose cells contain only 0's.



Number of Decision makers: 5

e:

1	1	0	0	0

s:

0	0	0	1	1

F:

▨	0	0	0	0
0	▨	0	0	0
0	0	▨	0	0
0	0	0	▨	0
0	0	0	0	▨

G:

▨	0	0	0	0
0	▨	0	0	0
0	0	▨	0	0
0	0	0	▨	0
0	0	0	0	▨

H:

▨	0	1	2	2
0	▨	2	2	2
0	0	▨	0	0
0	0	0	▨	0
0	0	0	0	▨

C :

▨	0	0	0	0
0	▨	0	0	0
0	0	▨	1	2
0	0	0	▨	0
0	0	0	0	▨

Figure 15 Connection Matrices for the Example

The set of solutions generated by the Lattice Algorithm is bounded by one MAXO, M1, and five MINOs, m1 to m5, and can be characterized by Figure 16. Each arrow connecting each MINO to the MAXO corresponds to a lattice in which every node is a solution. Each solution is obtained by adding simple paths to the MINO or subtracting simple paths from the MAXO. To continue the design process, the pair MINO m5 - MAXO M1 has been selected.

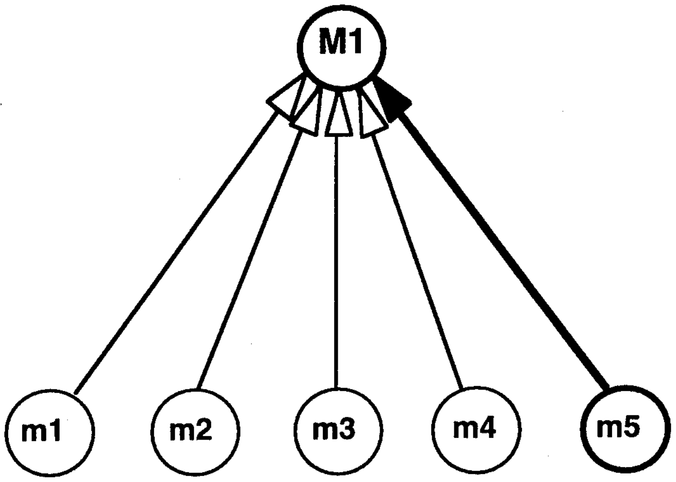


Figure 16 Characterization of the set of solutions

Figure 17 represents the Hasse Diagram of the lattice bounded by MINO m5 and MAXO M1. The example has been chosen so that all the solutions could be shown on this figure. In a more general case, the solutions bounded by a MINO and a MAXO can be very numerous. The numbers inside every node correspond to the simple paths of the corresponding organization. Thus, MINO m5 is made of the combination of paths #3, #4, #7, and #8. The subtraction of any of these paths violates the connectivity and structural constraints. MAXO M1 is obtained by adding to MINO m5 simple paths #1, #2, #5 and #6. Addition of any other simple path (if any existed) would result in the violation of the connectivity and structural constraints. Any intermediate solution is obtained by adding a subset of these paths to the MINO m5.

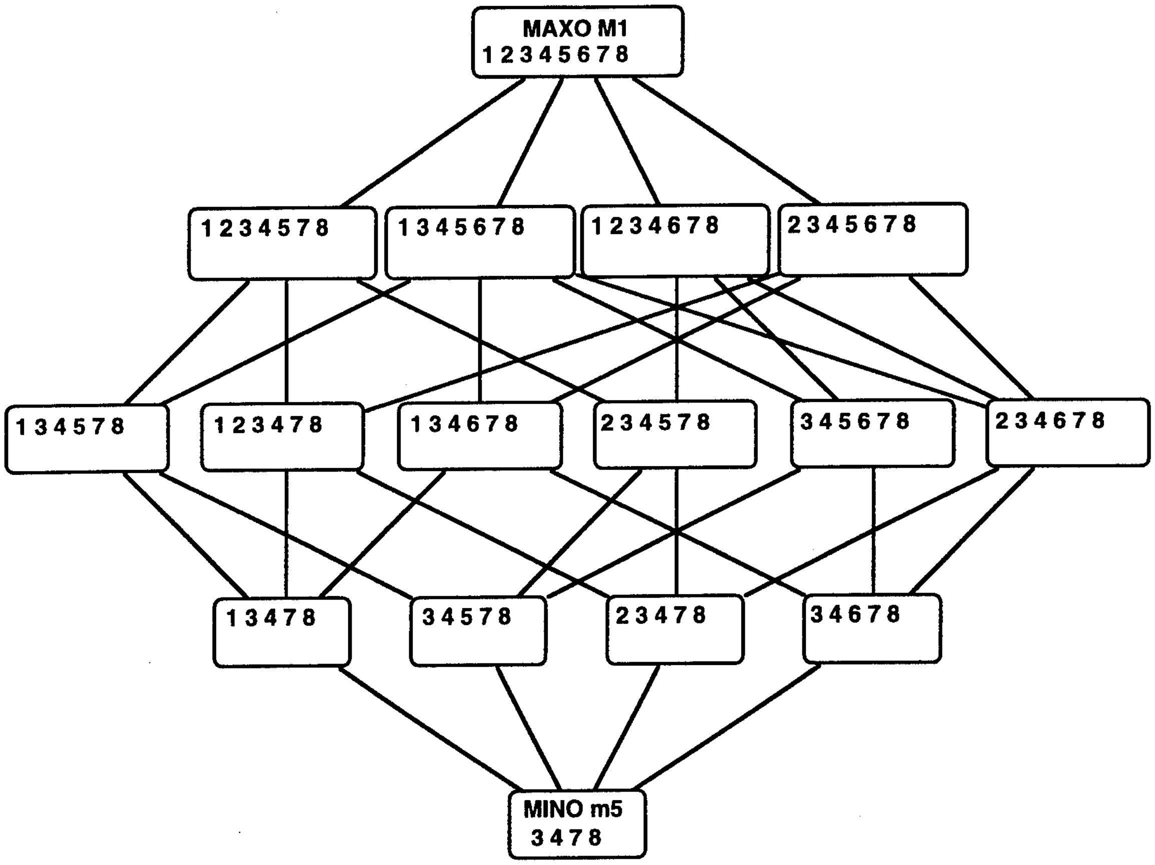


Figure 17 Hasse Diagram of the solutions bounded by MINO m5 and MAXO M1

The Petri Net representation of m5 and M1 are displayed on Figure 18 and 19.

As an example, Figure 20 represents the organization of interest obtained by adding simple paths #1 and #6 to the MINO or subtracting simple paths #2 and #5 from the MAXO. It corresponds to the organization where ASWC and AAWC receive a command from the CC, and where ASWC receives every processed information from the Sonar Operator and AAWC receives every processed information from the Radar Operator.

Once the structure has been determined, the decision process to be carried out by the organization needs to be defined and validated so that tasks can be derived and allocated to the different decision makers.

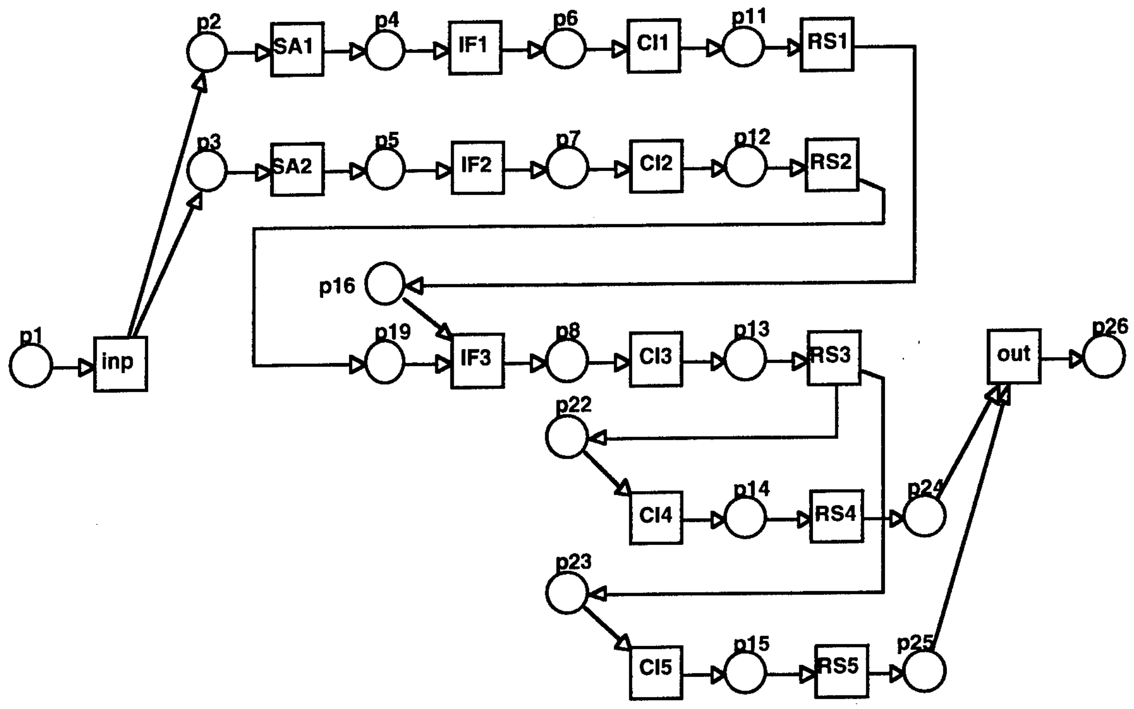


Figure 18 Petri Net Representation of MINO m5

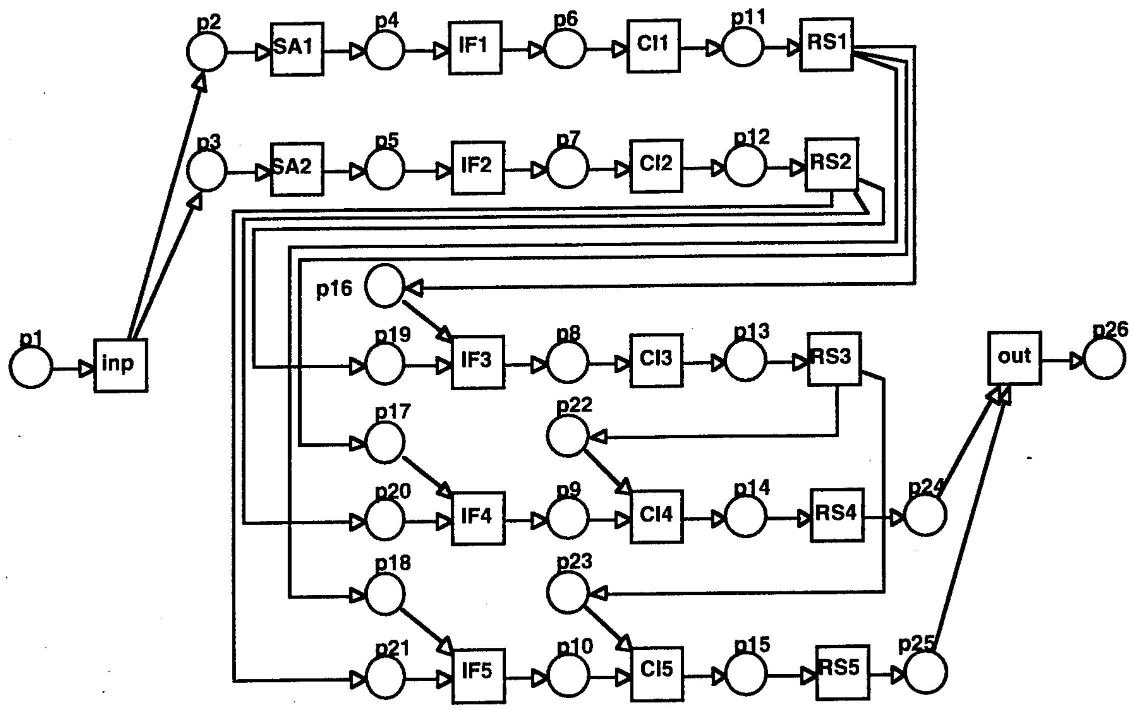


Figure 19 Petri Net·Representation of MAXO M1

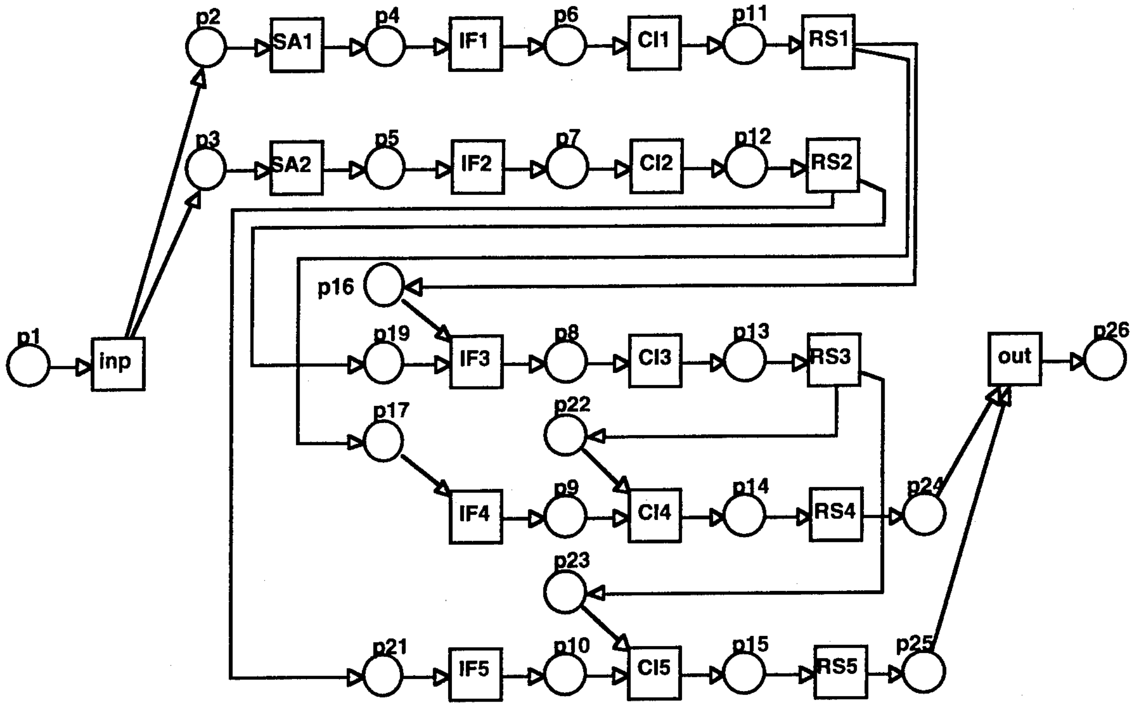


Figure 20 Petri Net representation of the Selected Organization

4.3.3 Decision Process and Task Allocation

Definition *Decision Process*

Given the feasible valid input space **U** and the set of valid responses Ψ, the Decision Process Γ associated with an organization is defined as:

$$\Gamma: \mathbf{U} \rightarrow \Psi$$

Figure 21 presents a schematic representation of a decision process, where $u_i \in$ U (the set of basic inputs) and $y_i \in \Psi$.

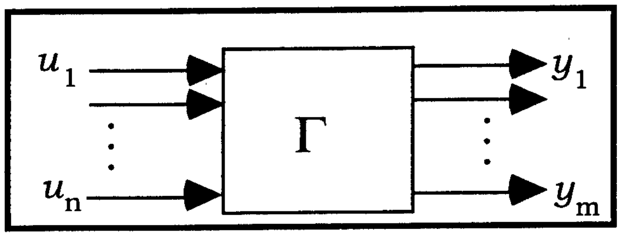


Figure 21 Decision Process

The decision making process Γ of an organization can be viewed as a set of decision making rules $\{r_i\}$, where each rule maps a valid input vector into a valid output response, as depicted in Figure 22. Using the terminology of Propositional Calculus (PC) a rule r in Figure 22 takes the form:

$$u_1 \ \& \ u_2 \ \& \ldots \& \ u_i \rightarrow y_j$$

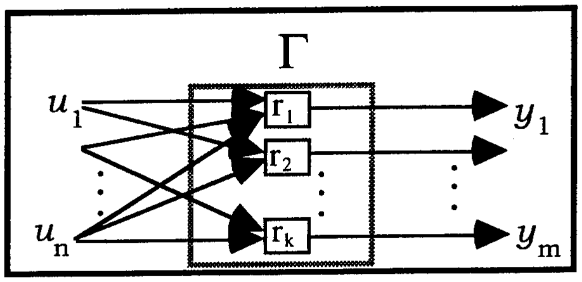


Figure 22 Decision Making Rules

The decision making rules of the type shown in Figure 22 represent the generic functionality of a decision process. However, such a partitioning of the set Γ may not be realizable when the rules are assigned to individual physical systems in the physical architecture of an organization. This could be due to the inaccessibility of certain inputs to the systems required to process them, sharing of information between two distant systems to select a response, lack of problem solving abilities at a node that has access to inputs, and so on. This, in turn, leads to the redefinition of the decision rules into a form that takes into consideration the constraints and requirements put forth by the physical architecture of the organization. Such a redefinition is termed as *Decomposition* of rules. Therefore, in order to assign the functionality present in the decision process to the individual decision makers in the organization, the original set of decision rules is required to be decomposed. The manner in which these rules are obtained, decomposed, and eventually partitioned across the decision making entities (human and machine) can introduce inconsistencies, incompleteness, redundancies, as well as problems in coordination.

In an organization, even if its physical and coordination structures are feasible, the presence of such problems in the set of rules assigned to decision making entities can result in poor performance and unreliable system response. Some of the examples of incompleteness and inconsistencies in the rule base associated with the Navy Tactical Command and Control Center are shown in Table 1.

Table 1 Examples of Incompleteness and Inconsistency in the Rule Base Associated with Navy Tactical Center

INCOMPLETENESS	INCONSISTENCY
• Rules requiring inaccessible inputs resulting in deadlocks • Relevant information or data not used in decision making resulting in poor performance	• Incorrectly ID friendlies as threats resulting in friendly fire • Correctly ID hostiles but produce no action • Correctly ID hostile but fire wrong weapon

As part of the example, the RULER algorithm is applied to the rule base associated with the Navy Tactical Command and Control Center. The inputs (basic concepts) and outputs (main concepts) of the system's decision process are listed in Table 2.

Table 2 Inputs and Outputs to the Rule Base Associated with Navy Tactical Command and Control Center

INPUTS	OUTPUTS
Radar: {r1, r2, r3}	Fire_SSM
Sonar Sensor: {p1, p2, p3}	Fire_SAM
Database: {db1, db2, ...}	Fire_Torpedo
Intel: {intel1, intel2, ...}	Fire_DepthCharge
	No_Action

The syntax of RULER requires that the input rules be expressed as statements in formal logic. An example of such a representation is presented below:

Example

Let a decision rule be expressed in a descriptive IF-THEN form as:

IF
the assessed information from sonar operator is Pi and form radar operator is Rj

THEN
the incoming object is an enemy vessel and ASW is commanded to attack

This rule in RULER syntax will be expressed as:

Pi & Rj —> Enemy_Vessel & Attack_ASW

The entire set of decision rules (45 rules) associated with the illustrative example of Navy Tactical Command and Control Center is given in Table 3.

In order to verify the correctness of the given rule base, RULER first transforms it to an equivalent Petri Net representation. An algorithm implemented on *Design CPN*™ using *ML*™ automatically generates a Petri Net from a text file containing the decision rules in the conditional form presented. A slightly edited version of the Petri Net so obtained is shown in Figure 23. All the inputs (basic concepts) and the outputs (main concepts) of the decision process are shown aggregated into two input and output places. This aggregation of inputs and outputs is required to identify the incompleteness present in the rule set.

Table 3 Rule Set Associated with the Naval Tactical Command and Control Center

1. *r1->R10*
2. *r2->R20*
3. *r3->R30*
4. *r1&r2->R12*
5. *p1->P10*
6. *p2->P20*
7. *p3->P30*
8. *p1&p2->P12*
9. *P10&R10;P20&R10;P10&R20->Unknown_Vessel*
10. *P12&R10;P12&R20->Friendly_Vessel*
11. *P20&R20;P10&R12;P20&R12->Enemy_Vessel*
12. *P10&R30->Enemy_Sub*
13. *P20&R30->Unknown_Sub*
14. *P12&R30->Friendly_Sub*
15. *P30&R10->Enemy_Plane*
16. *P30&R20->Friendly_Plane*
17. *P30&R12->Neutral_Plane*
18. *Unknown_Vessel&db1->Alert_AAW&Alert_ASW*
19. *Unknown_Vessel&db2&intel1->Attack_AAW&Attack_ASW*
20. *Unknown_Vessel&db3->No_Action*
21. *Friendly_Vessel->No_Action*
22. *Enemy_Vessel&db1->Attack_AAW&Attack_ASW*
23. *Friendly_Sub->No_Action*
24. *Neutral_Sub&db2&intel2->Attack_ASW*
25. *Neutral_Sub&db3->Alert_ASW*
26. *Neutral_Sub&db4->No_Action*
27. *Enemy_Sub&db3->Attack_ASW*
28. *Enemy_Sub&intel3->Alert_ASW*
29. *Enemy_Plane&db5->Alert_AAW*
30. *Enemy_Plane&intel3->Attack_AAW*
31. *Friendly_Plane->No_Action*
32. *Neutral_Plane&db6&intel2->Attack_AAW*
33. *Neutral_Plane&db7->Alert_AAW*
34. *Neutral_Plane&intel3->Alert_AAW*
35. *Neutral_Plane&db4->No_Action*
36. *R10&Alert_AAW;R12&Alert_AAW->Prepare_Launch*
37. *R10&Attack_AAW;R12&Attack_AAW->Fire_SAM*
38. *P10&Sub_Threat&Attack_ASW->Fire_DepthCharge*
39. *P10&Surface_Threat&Attack_ASW->Fire_Torpedo*
40. *P10&Sub_Threat&Alert_ASW->Prepare_DepthCharge*
41. *P10&Surface_Threat&Alert_ASW->Prepare_Torpedo*
42. *P20&Sub_Threat&Attack_ASW->Fire_DepthCharge*
43. *P20&Surface_Threat&Attack_ASW->Fire_Torpedo*
44. *P20&Sub_Threat&Alert_ASW->Prepare_DepthCharge*
45. *P20&Surface_Threat&Alert_ASW->Prepare_Torpedo*

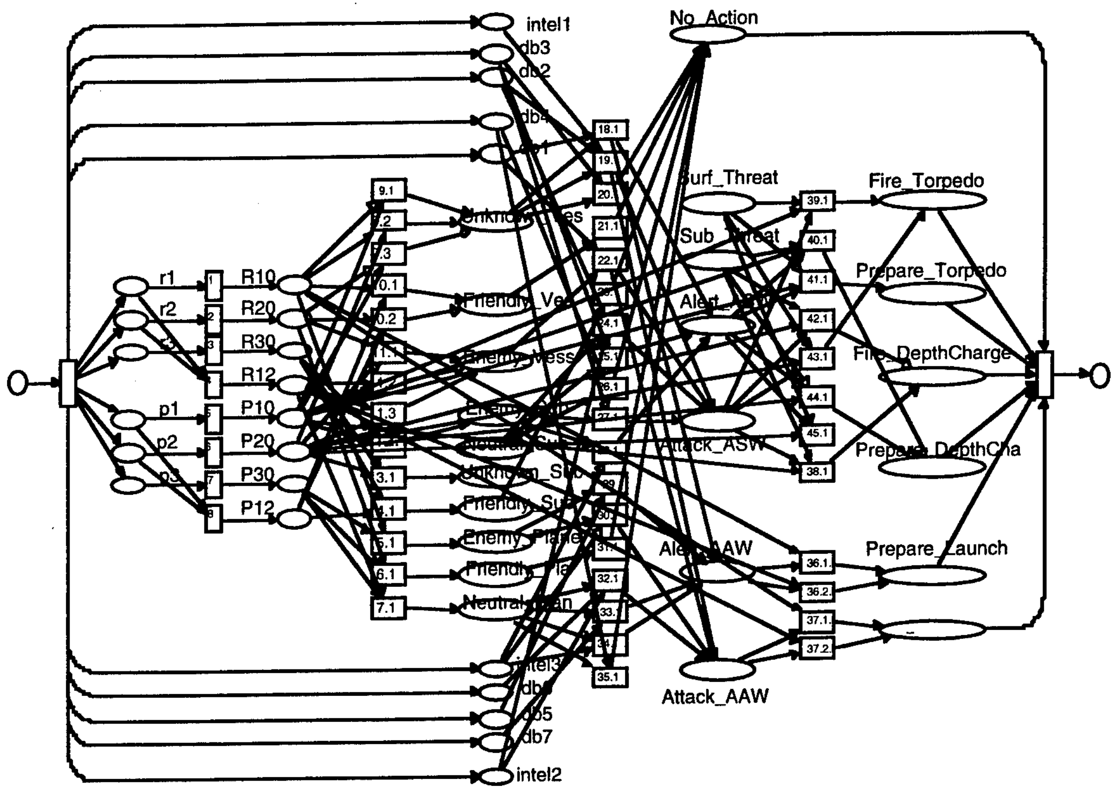


Figure 23 Petri Net Representation of the Rule Set

The application of RULER resulted in the following reported incompleteness and inconsistencies.

Reported Incompleteness

The following rules are reported as incomplete:

- P10 & Surface_Threat & Attack ASW —> Fire_Torpedo
- P10 & Subsurface_Threat & Attack_ASW —> Fire_DepthCharge
- P10 & Surface_Threat & Alert_ASW —> Prepare_Torpedo
- P10 & Subsurface_Threat & Alert_ASW —> Prepare_DepthCharge

The reported incompleteness is due to the missing rules conveying the nature of the incoming threat to the decision rules that are required to act upon it. In the rule set the concepts Surface_Threat and Subsurface_Threat appear as places with only outputs; they are used to make decisions but can not be inferred from the sensory inputs. Although the required information is available through the sensory inputs, but it is not explicitly represented in the rule set. And, it is this lack of explicit representation that causes the incompleteness reported by RULER.

Reported Inconsistency

The following rules are reported inconsistent:

- Enemy_Vessel & db1 —> Attack_ASW & Attack_AAW

- (R10 or R12) & Attack_AAW —> Fire_SAM

The reported inconsistency is due to the fact that the first rule conveys 'Attack' command to AAW based on identification of an incoming 'Enemy_Vessel', and the second rule acts upon this command with the firing of a wrong weapon (SAM).

Once the set of decision rules is checked for correctness, it is partitioned into several subsets where each set is assigned to an individual decision making entity in the organization. The process of assigning rules to roles is a heuristic process and requires information about the phyisical structure of the organization and a mapping between the capabilites of the individual decision making units to the functionality depicted by the decision process. Figure 24 represents a possible assignment of rules to five decision makers in the illustration of Navy Tactical Command and Control Center.

The complete executable model of the organization is obtained by implementing the rules into code segments attached to the transitions. It requires the definition of color sets and variables gathered in the global declaration node and the updating of the arc expressions. This process is not automated and has to be done manually in the Design/CPN editor. The model thus constructed can then be evaluated with the different analysis tools available in the third stage of CAESAR II and/or simulated to check that it exhibits the appropriate behavior.

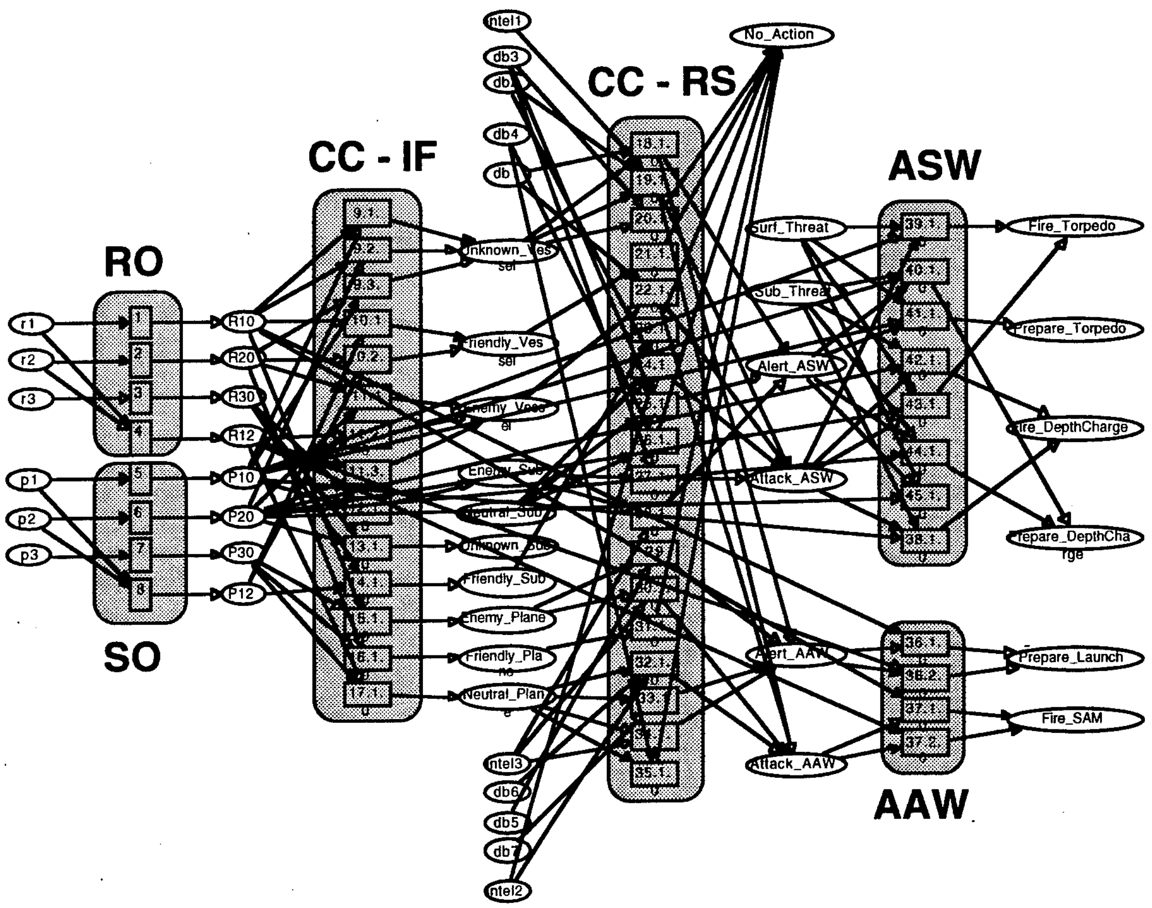


Figure 24 Rule Allocation

4.3.4 Performance and Effectiveness Analysis

To evaluate performance, a scenario needs to be defined and implemented as an initial marking of the source place. The parameters to be varied need also to be defined. These parameters can be either the presence of resources, the response time of some processes, the setting of decision strategies. In the example, the parameter of choice is the decision strategy, that is the probability to choose an algorithm in favor of another for the execution of a given process. The Measures of Performance (MOPs) of interest need to be defined. In the example, three MOPs were chosen: response time, accuracy (the expected cost for producing the organization response which is different from the desired response), and throughput rate (the number of tasks that can be executed during a certain amount of time). The model needs to be modified and instrumented to collect data that allow to compute the measures of performance. Sink places at appropriate locations in the Colored Petri Net are added to keep track of what is going on .

The model is simulated and data are collected for each setting of the parameters. Microsoft Excel™ is used to process the raw data and compute the MOPs reached for each of the settings. The Performance Locus is plotted in three dimensions in the Performance Space using MATLAB™. The performance space for the example is displayed on Figure 25.

To analyze whether the organization satisfies the requirement, the performance attained by the organization needs to be compared to the performance requirements. These performance requirements are expressed as inequalities: "the response time has to be less than 14 s". If the performance requirements are independent of each other, the requirement locus can be represented as a cube in the performance space, as shown in Figure 25. The effectiveness of the organization is then related to the extent to which the Performance locus is included in the Requirements locus. A measure of Effectiveness can be defined as:

$$\mathrm{MOE} = \frac{(L_p \cap L_r)}{L_p}$$

and can be used as a sound basis for comparing several candidate structures.

To complete the analysis, it is interesting to assess the sensitivity of the Effectiveness to requirements. By varying the levels of the performance requirements at different levels, and measuring the MOEs, the graph displayed on Figure 26 is obtained. It provides some useful information: a slight variation in the response time requirements from 13 s to 14 s result in a steep variation of the effectiveness while a variation of the requirements in accuracy introduces a smooth variation of the effectiveness. This type of analysis constitutes a sound technical basis for a refinement of the performance requirements.

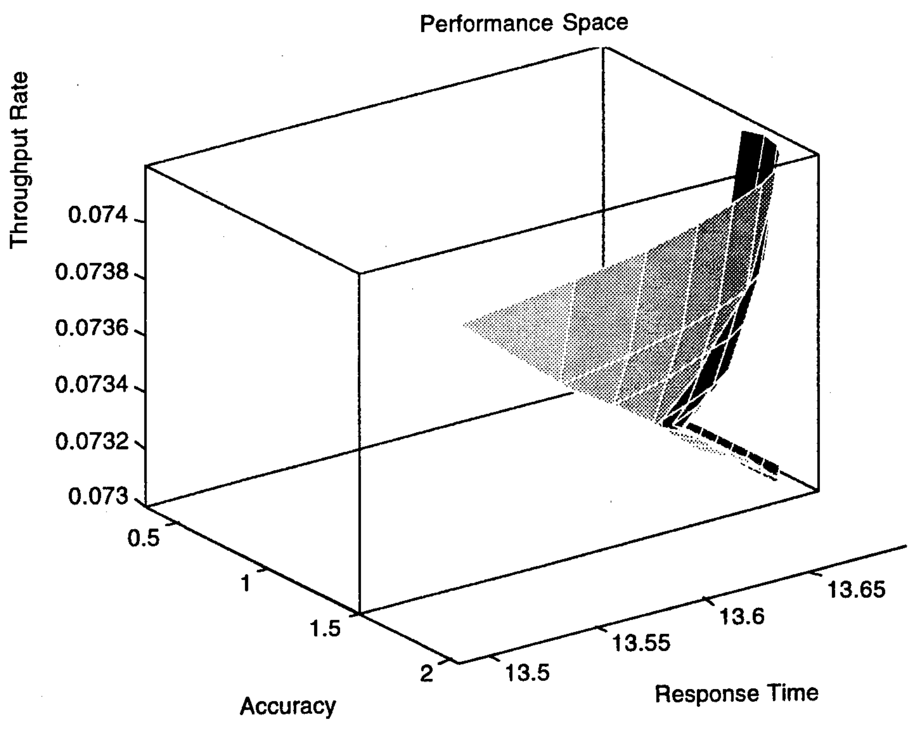


Figure 25 Performance and Requirements Loci

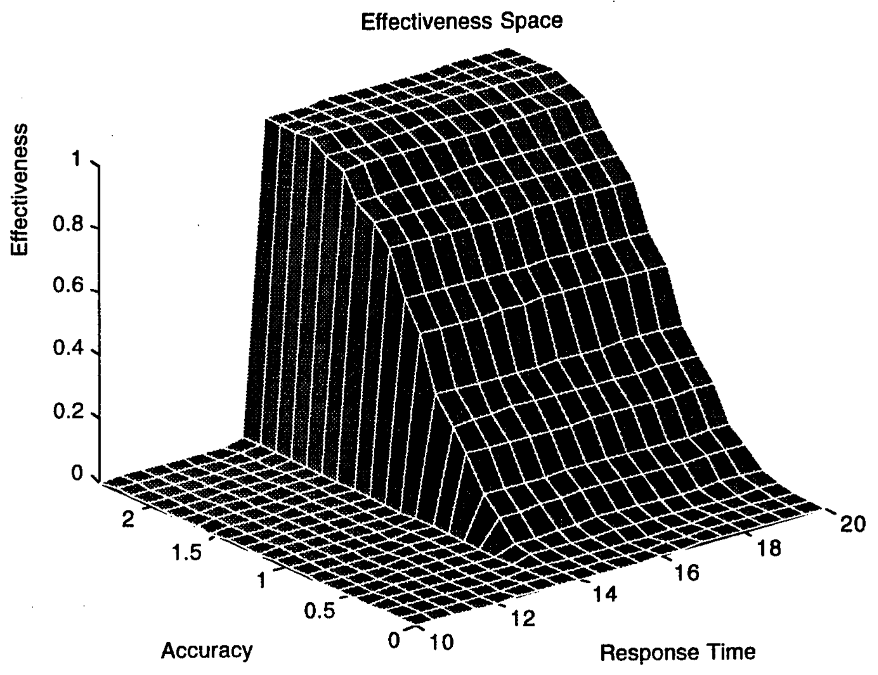


Figure 26 Sensitivity of Effectiveness to Requirements

4.4. SUMMARY

The emphasis during the second year of the project has been on fundamental research - the completion of A. Zaidi's PhD thesis - and the integration of the work into a suite of tools, called CAESAR II, that can be used to support basic research as well as transition the research results into the 6.2 and 6.3 areas.

4.5 REFERENCES

Andreadakis, S. K. (1988). *Analysis and Synthesis of Decision-Making Organizations.* LIDS-TH-1740, Laboratory for Information and Decision Systems, MIT, Cambridge, MA.

Demaël, J. (1989). *On the generation of Variable Structure Distributed Architectures.* LIDS-TH-1869, Laboratory for Information and Decision Systems, MIT, Cambridge, MA.

Jin, Zhenyi (1994). *Deadlock and Trap Analysis in Petri Nets.* MS Thesis, Systems Engineering Department, George Mason University, Fairfax, VA, May 1994.

Lu, Zhuo (1992). *Coordination in Distributed Intelligent Systems.* GMU/C3I-120-TH, Center of Excellence in Command, Control, Communications, and Intelligence, George Mason University, Fairfax, VA.

Remy P. (1986). *On the Generation of Organizational Architectures Using Petri Nets.* LIDS-TH-1630, Laboratory for Information and Decision Systems, MIT, Cambridge, MA.

Remy P. and Levis A.H. (1988). On the Generation of Organizational Architectures Using Petri Nets. in *Advances in Petri Nets 1988, Lecture Notes in Computer Science*, G. Rozenberg Ed. Springer Verlag, Berlin, Germany.

Valraud F. (1989). *Evaluation of Functionality in Distributed Systems.* LIDS-TH-1868, Laboratory for Information and Decision Systems, MIT, Cambridge, MA.

Zaidi A. (1991). *On the Generation of Multilevel Distributed Intelligent Systems Using Petri Nets.* M. S. Thesis, Report GMU/C3I-112-TH, Center of Excellence in Command, Control, Communications, and Intelligence, George Mason University, Fairfax, VA.

Zaidi A. (1994). *Validation and Verification of Decision Making Rules.* Ph. D. Dissertation, Center of Excellence in Command, Control, Communications, and Intelligence, Report GMU/C3I-155-TH, George Mason University, Fairfax, VA.

5.0 MEETINGS

- Dr. Zaidi defended successfully his Ph. D. thesis in December 1994.
- Drs. Levis and Zaidi and Mr. D. Perdu attended the 1995 Symposium on C2 Research and Technology, National Defense University, Washington DC.
- Drs. Levis and Zaidi visited Alphatech, Inc. in Burlington, MA to discuss collaboration during the third year of the project.
- Drs. Levis and Zaidi attended the 6th IFAC Man-Machine Systems Symposium at MIT, Cambridge, MA and presented a paper based on Dr. Zaidi's thesis.

6.0 CHANGES

No changes in the scope of work of this project.

7.0 RESEARCH PERSONNEL

7.1 Research Personnel — Current Reporting Period

Prof. Alexander H. Levis	Principal Investigator
Prof. Abbas K. Zaidi	Postdoctoral Fellow
Mr. Didier M. Perdu	Graduate Student (Ph.D.)
Mr. Eric Tsibertzopoulos	Graduate Res. Assistant (MS)
Ms. Etsiwohot Dinka	Undergraduate Student

7.2 Research Personnel — In Previous Reporting Periods

Prof. K. C. Chang	
Mr. Abbas Zaidi	Graduate Res. Assistant (Ph.D.)
Ms. Jenny Jin	Graduate Res. Assistant (MS)
Ms. Hedy Rashba	Graduate Res. Assistant
Ms. Azar Sadigh	Graduate Res. Assistant
Ms. Cynthia Johnson	Graphics Designer

7.3 Personnel Changes

Ms. Dinka joined the project as junior programmer and is working on the implementation of the interfaces in CAESAR II.

Mr. Eric Tsibertzopoulos joined the project in June as a Graduate Research Assistant. He will be doing his Master's thesis in this area.

Dr. Zaidi completed his Ph.D dissertation in December of 1994 and then continued to work on the project as a postdoctoral fellow.

8.0 DOCUMENTATION/PUBLICATIONS

1. Hedy L. Rashba, "Problems in Concurrency and Coordination in Decision Making Organizations," Report GMU/C3I-143-R, C3I Center, George Mason University, Fairfax, VA, September 1993.

2. A. H. Levis, B. Hu and N. Moray, "Task Allocation Models and Discrete Event Systems," *Automatica,* Vol. 30, No. 2, Feb. 1994.

3. G. Johannsen, A. H. Levis and H. Stassen, "Theoretical Problems in Man-Machine Systems and their Experimental Validation," *Automatica*, Vol. 30, No. 2, Feb. 1994.

4. T. Zhang and A. H. Levis, "Colored Invariants in Predicate Transition Nets," *Proc. First International Symposium of Young Investigators on Information/ Computer/ Control (ISYI' 94)* Beijing, China, February 1994.

5. J. J. Demaël and A. H. Levis, "On Generating Variable Structure Architectures for Decision Making Systems," *Information and Decision Technologies,* vol. 19, 1994, pp. 233-255.

6. Zhenyi Jin, "Deadlock and Trap Analysis in Petri Nets," MS Thesis, Systems Engineering Department, George Mason University, Fairfax, VA, May 1994.

7. D. M. Perdu, C. Spohnholtz and A. H. Levis, "Modeling Information Pull in the Copernicus Architecture using Colored Petri Nets," *Proc. 1994 Symposium on C2 Research,* SAIC, McLean, VA. June 1994

8. A. H. Levis and I. S. Levis, Eds. *Science of Command and Control. Part III: Coping with Change.* AFCEA International Press, Fairfax, VA, November, 1994.

9. A. K. Zaidi, "Validation and Verification of Decision Making Rules," Ph. D. Thesis, Center of Excellence in Command, Control, Communications, and Intelligence, Report GMU/C3I-155-TH, George Mason University, Fairfax, VA. December 1994.

10. A. H. Levis, "Human Interaction with Decision Aids: A Mathematical Approach," in *Human /Technology Interaction in Complex Systems,* Vol. 7, E. B. Rouse, Ed., JAI Press, 1995.

11. Zhenyi Jin, A. K. Zaidi, and A. H. Levis, "Deadlock and Trap Analysis in Petri Nets," *Proc. 1995 Symposium on Command and Control Research and Technology,* National Defense University, June 1995

12. A. K. Zaidi and A. H. Levis, "Rule Decomposition and Validation for Distributed Decision Making," *Proc. 1995 Symposium on Command and Control Research and Technology,* National Defense University, June 1995.

13. A. K. Zaidi and A. H. Levis, "On Verifying Inferences in an Influence Net," *Proc. 1995 Symposium on Command and Control Research and Technology,* National Defense University, June 1995.

14. A. K. Zaidi and A. H. Levis, "Validation and Verification of Decision Making Rules," *Proc. 6th IFAC Symp. on Man Machine Systems,* MIT, Cambridge, MA , June 1995.