SEMI-ANNUAL STATUS REPORT NO. 2

(COVERING THE PERIOD FROM 1 NOVEMBER 1976 TO 30 APRIL 1977)

MODELS AND TECHNIQUES FOR EVALUATING THE EFFECTIVENESS OF AIRCRAFT COMPUTING SYSTEMS

PRINCIPAL INVESTIGATOR

JOHN F. MEYER

SEL REPORT NO. 111

PREPARED FOR:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
LANGLEY RESEARCH CENTER
HAMPTON, VIRGINIA 23365

SAL J. BAVUSO - NASA TECHNICAL OFFICER

NASA GRANT NSG 1306

JULY 1977

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
SYSTEMS ENGINEERING LABORATORY
THE UNIVERSITY OF MICHIGAN, ANN ARBOR

TABLE OF CONTENTS

1. INTRODUCTION

This report is the second Semi-Annual Status Report on the research project "Models and Techniques for Evaluating the Effectiveness of Aircraft Computing Systems" being conducted for the NASA Langley Research Center under NASA Grant NSG 1306. The report concerns work accomplished during the period from 1 November 1976 to 30 April 1977, hereafter referred to as the "reporting period."

The purpose of this research project is to develop models, measures and techniques for evaluating the effectiveness of aircraft computing systems. By "effectiveness" in this context we mean the extent to which the user, i.e., a commercial air carrier, may expect to benefit from the computational tasks accomplished by a computing system in the environment of an advanced commercial aircraft. Thus the concept of effectiveness involves aspects of system performance, reliability and worth (value, benefit) which must be appropriately integrated in the process of evaluating system effectiveness. More specifically, the primary objectives of this project are:

1) The development of system models that can provide a basis for the formulation and evaluation of aircraft computer system effectiveness,
2) The formulation of quantitative measures of system effectiveness, and
3) The development of analytic and simulation techniques for evaluating the effectiveness of a proposed or existing aircraft computer.

Work proposed for the first year of this project was to be concerned primarily with objectives 1) and 2). During the previous reporting period (1 May 1976 to 31 October 1976), the main thrust of our work was in line with the first objective (see the first Semi-Annual Status Report [1]). During the current reporting period, our effort has been aimed at both model development (objective 1)) and the formulation of effectiveness measures (objective 2)). More specifically, our work has concerned:

i) More detailed development of the model hierarchy at mission, functional task, and computational task levels, with emphasis placed on the modeling of an air transport mission.

ii) Investigation of an appropriate class of stochastic models that can serve as bottom level models in the hierarchical modeling scheme. The scope of a model at this level is some specified aircraft computer (the system to be evaluated) and the level of abstraction is the "operational state" of the computer's hardware and software.

iii) Definition and formulation of a unified measure of effectiveness called "performability" and, in particular, the "capability" aspect of performability which expresses top model behavior (levels of performance) as a function of base model behavior.

We believe that the following report attests to a substantial amount of progress in each of these areas. Moreover, we feel that the progress to date is compatible with what we had anticipated when writing the proposal for the second year of the project [2].

Section 2 of the report describes the manpower effort proposed for the past year, the personnel involved in conducting the investigation, and their levels of effort during the reporting period. Section 3, the body of the report, describes the technical status of the research performed during the reporting period.

2. PERSONNEL

At the initiation of the project, it was estimated that the following effort would be required during the first year.

Principal Investigator

100%, two months, summer
25%, nine months, academic year

Graduate Student Research Assistants

The equivalent of:

3 at 100%, three months, summer
3 at 25%, eight months, academic year

During the reporting period from 1 November 1976 to 30 April 1977, research personnel and their levels of effort have been:

Principal Investigator

John F. Meyer
25%, November 1976-March 1977
50%, April 1977

Graduate Student Research Assistants

Robert A. Ballance
25%, January 1977 - April 1977

David G. Furchtgott
25%, November 1976 - April 1977

Liang T. Wu
25%, November 1976 - April 1977

3. TECHNICAL STATUS

As stated in the introduction, our research during the reporting period has progressed in three principal areas.

The first area has been a more detailed development of the higher level models in the hierarchy, i.e., the models at the mission, functional task, and computational task levels. In particular, our efforts here have focused on i) establishing a general methodology for formulating higher level models and ii) applying this methodology to the formulation of a prototype model hierarchy for a specific type of mission, i.e., an air transport mission.

The second area has been the investigation of an appropriate class of stochastic models that can serve as bottom level models in a model hierarchy. A model in this class is a model of some specified aircraft computer and the level of abstraction is the "operational state" of the computer's hardware and software. The goal here is not to develop a bottom level model for a specific aircraft computer architecture (e.g., SIFT [3] or OSIRIS [4]) but, instead, to determine a type of stochastic model which is i) capable of representing a variety of fault-tolerant computer architectures and ii) compatible with a hierarchical modeling scheme.

The third area of effort has been the definition and formulation of a unified measure of effectiveness called "performability" which comprises three principal measures: "availability," "dependability" and "capability." The first two measures quantify the behavior of the bottom model of the hierarchy and are the usual objects of study in classical structure-based reliability analysis. Accordingly, the bulk of our effort here has concerned the third measure, "capability," which expresses top model behavior (levels of

total system performance) as a function of the "basic variables" of the bottom and intermediate level models. In particular, we have established a class of capability measures referred to as "capability functions" and have developed a general concept of "functional dependence" (called "R-dependence") which is applicable to capability functions.

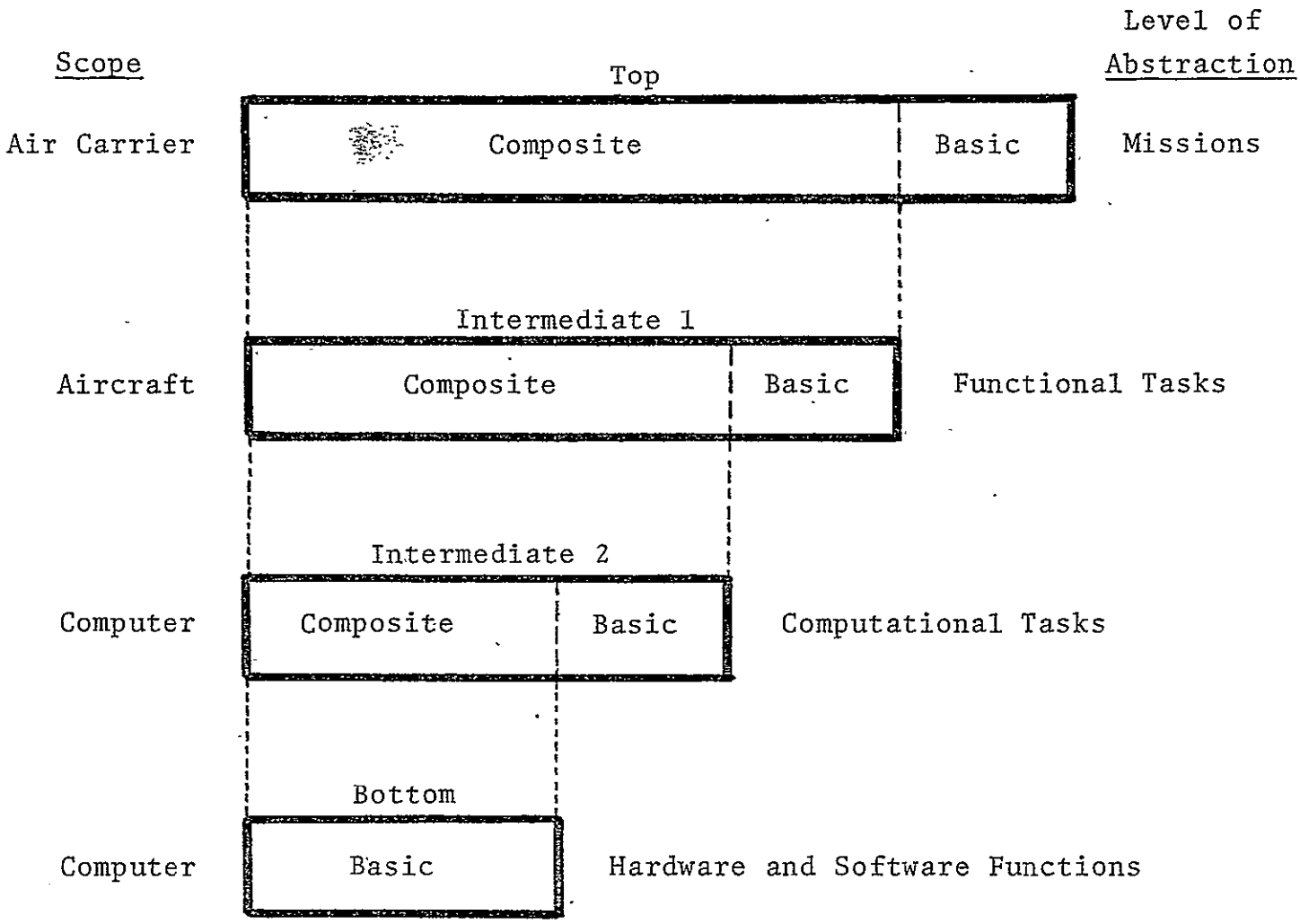
The status of our research in each of these areas is described in the subsections that follow.

3.1 Higher-Level Models

3.1.1 A Model Hierarchy

During the reporting period, we have continued the development of the system models described in the first Semi-Annual Status Report [1]. As discussed in [1], we seek a model of the total system with a behavior relating directly to the user's requirements and a structure accurately describing the probabilistic nature of the system's components. This view requires a high, user-oriented level with scope comprising the total system (i.e., the air carrier) as well as a low, structure-oriented bottom level comprising the object system (i.e., the computing system and closely related peripheral equipment). Also, in order to relate the performance of the computer hardware (bottom level) to the accomplishment of user-oriented missions (top level), we have concluded that at least two intermediate levels are necessary. These are the aircraft functional task level (the higher of the two intermediate levels) and the computational task level (the lower of the two levels).

Because the bottom level concerns the object system, we have found that information from non-object systems (e.g., environment, supporting and related systems) is more easily introduced at these higher levels. Using what we call "basic variables," we incorporate each non-object system into the hierarchy based on the level at which that information is used (see Figure 1). For example, "weather" does not depend on any aircraft function and yet it can affect the mission outcome; thus, weather may be introduced at the aircraft functional level.



Basic variables at some level are newly introduced at that level.

Composite variables at some level are supported by variables at the next lower level.

Figure 1

General description of a model hierarchy for aircraft computing systems

The bottom model, along with the higher level basic variables, are referred to collectively as the "base model" of the total system. Formally, the connection between the behavior of the base model and that of the top (mission level) model is expressed by a "capability function," the discussion of which is deferred to Section 3.3.2.2. In general, the interaction between various levels of a model hierarchy can be viewed either as a part of the hierarchy, per se, or as something which is determined later, in the process of using the model to analyze some aspect of system behavior, e.g., its performability. Either view is legitimate, but the latter appears to be more convenient for the purpose of classifying and discussing these interactions.

3.1.2 Model Descriptions

3.1.2.1 Top Level Models

Extending the effort described in the first Semi-Annual Report [1], we have established the following methodology for formulating the top level model. We begin with an informal general description (or concept) of system missions. These are simple English statements telling us what system activities the user deems desirable and pertinent to each mission. Thus, for a transport mission, this statement may be "Transport passengers between two points quickly, safely, conveniently and with minimum fuel consumption." Note that one can always vary or expand upon this statement to address other missions; to illustrate, we could talk about long, intermediate or short range missions. Other mission types have also been examined, especially a maintenance mission

and a "standby" mission (in which a spare plane is stored to use as a backup in case a plane in service breaks down).

Next, from the mission statement we derive a list of the relevant mission factors. This is called the "mission requirement set." For the air transport mission above, we have:

1) Passengers are to be transported with time constraints
2) A given safety rate is to be attained
3) Inconveniences (diversions, arrival delays, etc.) are to be minimized
4) Fuel consumption is to minimized.

The next step is a temporal decomposition of missions. Typically, there exist intervals ("phases") during which a system is concentrating most of its facilities upon one activity (or set of activities) and allocating fewer resources to other activities [5]-[7]. Furthermore, during these intervals, system requirements are generally constant, although they may change radically between phases. For instance, the computational demand on a computer during a cruise portion of a flight may be much less than the demand during an autoland portion. An air transport mission, for example, can be naturally decomposed into

i) Takeoff,
ii) Climb,
iii) Cruise,
iv) Descent,
v) Approach and Landing.

Once we have described a mission and its desired goals, we then classify each mission outcome by determining various

mission "qualities" or "levels of accomplishment." This is an indication of the general outcome of each mission based on how well the demands of the mission requirement set are fulfilled. For instance, with the air transport mission, several levels of accomplishment could be informally stated as follows:

i) Flight with no diversion and low fuel consumption

ii) Flight with no diversion and medium to high fuel consumption

iii) Flight with diversion

iv) Flight involving a fatal accident.

To obtain a more formal characterization of these levels of accomplishment, with each mission we associate a "mission variable set" which is a set of variables such that if the value of each variable is known, then so is the level of accomplishment. It is also desirable (but not necessary) that each variable depend on the nature of the object system in the sense that the value assumed by a given variable will differ for at least two different object systems. To illustrate this concept, consider the air transport mission whose requirement set is given above. Then a sufficient mission variable set might be the following:

a) Seating capacity [integer, in passengers]

b) Flight distance [real, in kilometers]

c) Aircraft speed [real, in kilometers/hour]

d) Fuel consumption [trinary valued, with
0 = low consumption rate
1 = medium consumption rate
2 = high consumption rate]

e)	Fuel capacity	[integer, in kilograms]
f)	Safety	[binary valued, with 0 = no fatalities 1 = fatalities]
g)	Arrival delay	[integer, in minutes]
h)	Diversion	[binary valued, with 0 = no diversion 1 = diversion].

Note that the mission requirement set connotes certain necessary values for certain items, while the mission variable set places no such bound on any variable. This is because the mission requirement set describes what the user wants the system to do while the mission variable set describes what the system actually does.

Given a set of mission variables, to formulate levels of (mission) accomplishment, we define an "outcome" of a mission to be a particular sequence of values of mission variables, one value for each variable in the mission variable set. More precisely, if there are ℓ mission variables, let z_i denote the i^{th} variable and let R_i denote the range of z_i (i.e., the variable z_i assumes values in set R_i). Then a mission outcome is an element of the set $R_1 \times R_2 \times \ldots \times R_\ell$ and a level of (mission) accomplishment is a nonempty set of mission outcomes. The interpretation of a level of accomplishment is that all outcomes in the level are relatively indistinguishable from the user's point of view. Generally, for a given mission, there will be several levels of accomplishment (referred to as the range of accomplishment) such that every possible mission outcome is contained in one and only one level.

To illustrate this notion, if mission variables a) - h), described above, are denoted $z_1, z_2, \ldots, z_8$, respectively, then accomplishment level ii) (Flight with no diversion and medium to high fuel consumption) is formally represented by the set:

$$A_2 = \{(z_1, z_2, \ldots, z_8) \in R_1 \times R_2 \times \ldots \times R_8 \mid z_4 \in \{1,2\} \text{ and } z_8 = 0\}.$$

In addition, one can incorporate the notion of phase into the process of delineating levels of accomplishment by allowing the requirements to change with time. For instance, in level i) above, we might require "fuel consumption" = 0 during the cruise phase, and then allow "fuel consumption" = 0 or 1 during all other phases. Figure 2 illustrates this point.

3.1.2.2 Intermediate Level Models

The intermediate level models represent successive layers of coarseness in bridging the bottom, object system (computer) model and the top, mission level model. For an aircraft computing system, we believe that at least two such intermediate levels are needed to facilitate the determination of the relation between the bottom and top levels (i.e., the determination of the capability function; see Section 3.3.2.2).

First, below the air carrier level, we have the "aircraft functional task" level, characterizing the aircraft and especially those aircraft systems affected by the computer (e.g., autoland systems, stability augmentation systems and navigation systems; see Ratner, et al. [9] for one such list of requirements). As with

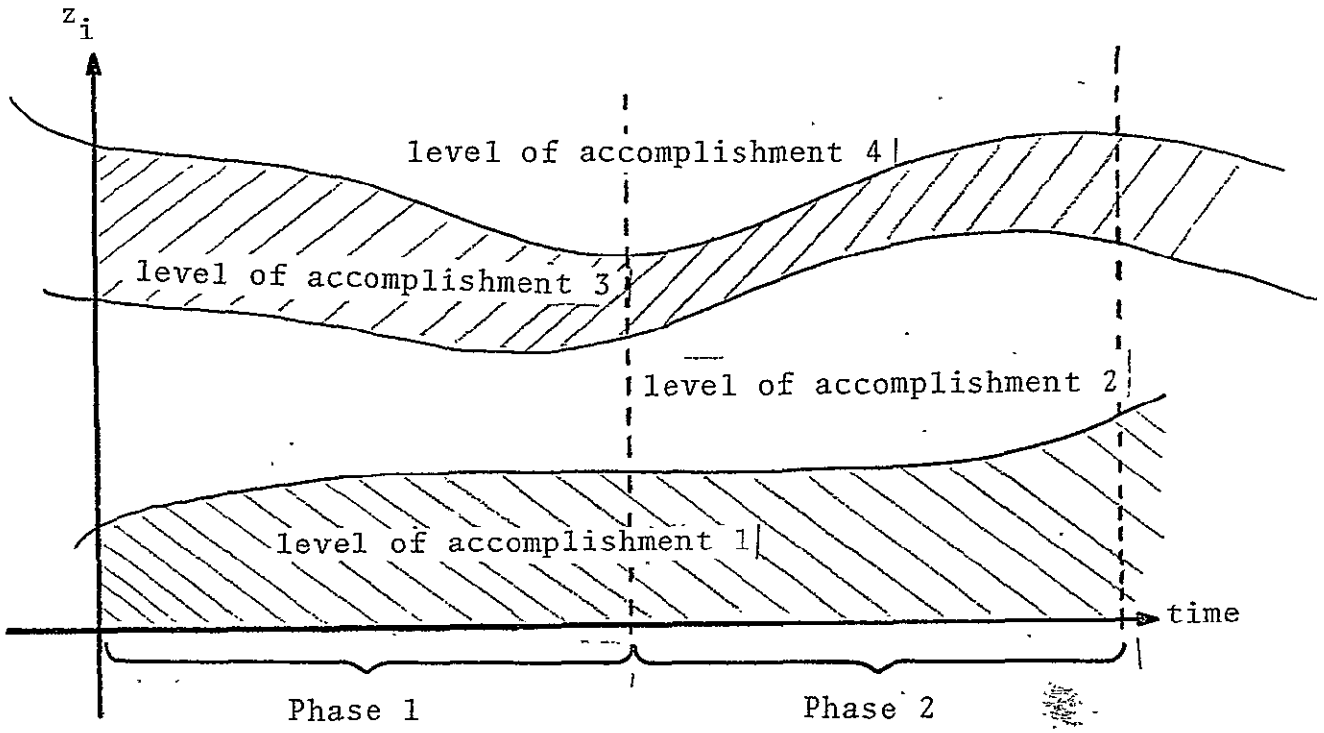


Figure 2

Levels of accomplishment delineated by some variable i and phase

missions, we can define task requirement sets to describe the demands of a given task and task variable sets to describe system performance relative to those demands. Thus, for example, stability augmentation requirements may be stated simply as the (singleton) set:

a) Aircraft stability is to be kept within a specified tolerance level (where the level may vary according to phase).

Control theory abounds with variables which could be used to describe system performance relative to this requirement (see [10] for details. \ One example of a stability variable set might be:

a) Steady state error (pitch, roll, yaw) [real, in degrees]

b) Maximum overshoot (pitch, roll, yaw) [real, in degrees]

c) Rise time (roll, pitch, yaw) [real, in seconds]

d) Settling time (roll, pitch, yaw) [real, in seconds].

Alternatively, a simpler example which might well suffice is the singleton variable set: \

a) Stability [trinary valued, with

0 = high degree of stability
1 = medium degree of stability
2 = no stability].

The development of variable sets for other functional tasks (e.g., autoland, cruise navigation, etc.) is carried out in a similar fashion.

Next, between the aircraft functional tasks level and the bottom level, we have introduced a "computational task" level, describing the basic operations the computer is required to

perform in order to accomplish the aircraft functional tasks. These computational tasks include such activities as those suggested by Ratner, et al. [9].

a) Vertical guidance
b) Horizontal guidance
c) Engine control.

Here, too, we can define task requirement sets and task variable sets. Thus, for task a) above, we may define the following task requirement set:

a) Computations involving vertical guidance are to be accomplished within given time and accuracy constraints

from which we derive the following task variable set:

a) Program access [binary valued, with
0 = access to vertical guidance program
1 = no access]

b) Instruction rate [integer, in average number of instructions devoted to vertical guidance computations per second]

c) Computation size [integer, in average number of instructions used in the performance of a single pass of the vertical guidance program].

We have also been studying the problem of incorporating non-computer related information into the model hierarchy in a systematic way. One solution which shows promise is to distinguish two types of model variables at each level of the model hierarchy. More precisely, a model variable at level i is a

i) basic variable if its values cannot be expressed in terms of model variables at level i+1 (the next lower level),

ii) composite variable if it is not basic, i.e., its values can be determined by knowing the values of the level i+1 variables.

Thus, the set of basic variables at level i represents the increase in scope of the level i model relative to that of the level i+1 model. (At the lowest level, all variables are basic.) Composite variables, on the other hand, lie within the scope of the next lower model and are determinable as a function of lower level model behavior (see Figure 1).

To illustrate this distinction, consider the "diversion" variable (as discussed above) introduced at the top level mission model (i=0). At the next lower level this could be expressed in terms of two variables, i) weather and ii) autoland. Thus "diversion" is a composite variable at level 0. If "autoland" is further divided into the computational tasks required for autoland, then "autoland" is composite at level 1. Without further decomposition, "weather" is a basic variable at level 1 (see Figure 3).

3.1.3 Hierarchical Modeling of an Air Transport Mission

During the present reporting period, we have investigated several prototype air transport models. In the sections that follow, we present a simplified model intended to demonstrate some of the major points discussed in the preceeding sections. We will develop the model in a top down manner, applying the general method described above. Furthermore, many of the examples in the prior discussion will be incorporated below, though usually in a simplified form.

It should be noted that, within the hierarchy, there are several other facets of modeling which we have been investigating but which are not reflected in this example. These are discussed in Sections 3.2 and 3.3 and include the bottom model (hardware and

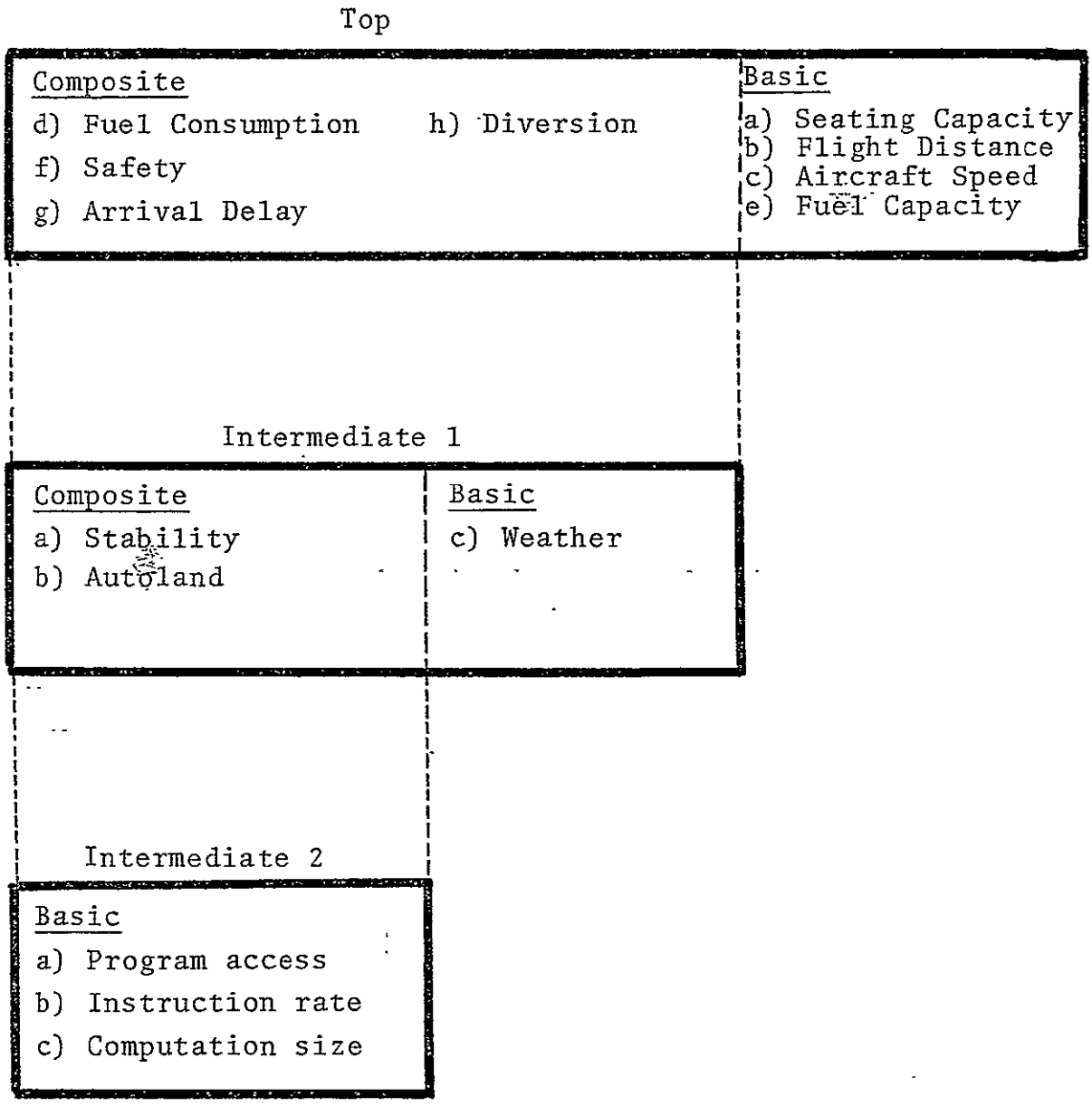


Figure 3

Sample variables of a model hierarchy
for aircraft computing systems

software functions), "interphase transition functions" (functions yielding the system configuration after a change in phase) and dependencies (both temporal and structural).

3.1.3.1 Top Level Model Development

The mission developed here is a basic air transport mission which can be informally described as follows:

Mission Statement: "Transport passengers between two points safely, conveniently and with minimal fuel consumption."

Mission Requirement Set:

i) A given safety rate is to be attained.

ii) Inconveniences (diversions) are to be minimized.

iii) Fuel consumption is to be minimized.

Levels of Accomplishment

1) Flight with no fatalities, no diversion and low fuel consumption

2) Flight with no fatalities, no diversion and high fuel consumption

3) Flight with no fatalities, diversion and low fuel consumption

4) Flight with no fatalities, diversion and high fuel consumption

5) Flight with fatalities.

Given the mission requirement set we designate the following mission variable set:

z_1: Safety [binary valued, with
0 = no fatalities
1 = fatalities]

z_2: Diversion [binary valued, with
0 = no diversion
1 = diversion]

z_3: Fuel Consumption [binary valued, with
0 = low fuel consumption
1 = high fuel consumption]

and, accordingly (see the general discussion in the previous subsection), the five levels of accomplishment are formally represented by the sets:

$A_1 = \{(0,0,0)\}$
$A_2 = \{(0,0,1)\}$
$A_3 = \{(0,1,0)\}$
$A_4 = \{(0,1,1)\}$
$A_5 = \{(1,0,0),(1,0,1),(1,1,0),(1,1,1)\}$.

3.1.3.2 Intermediate Level Development

At the aircraft task level, there are a number of functional tasks which are needed to support the accomplishment of the mission. To simplify the exposition, however, let us suppose that there are only two types of tasks that need to be accomplished:

a) Active Control (stability augmentation/fuel regulation)
b) Autoland.

Let us suppose further that the air transport mission has three phases: takeoff, cruise and landing, that active control is required in varying degrees throughout the flight and that autoland is required if Category III weather conditions exist at the time landing is to be initiated. If autoland is required but is not available at the time landing is to be initiated (due either to a faulty computer or to a computer which is not designed to support the autoland task), the flight is diverted to another airport.

Given these requirements, we formalize the Intermediate 1 model as follows. The takeoff, cruise and landing phases are denoted as phase 1, phase 2 and phase 3, respectively, and for the active control task we designate three task variables:

$$y_{11}, y_{12}, y_{13}$$

where

$$y_{1j} \in \{0,1,2\} \quad , \quad j = 1,2,3.$$

The interpretation of y_{1j} is the level of accomplishment of the active control task during the j^{th} phase where:

$$y_{1j} = \begin{cases} 0 \text{ if there is stability augmentation and fuel regulation during the } j^{th} \text{ phase} \\ 1 \text{ if there is stability augmentation but no fuel regulation during } j^{th} \text{ phase} \\ 2 \text{ if there is no active control during } j^{th} \text{ phase.} \end{cases}$$

For the autoland task we designate two task variables y_{22} and y_{23} where:

$$y_{22} = \begin{cases} 0 \text{ if the autoland capability is available at the end of the cruise phase} \\ 1 \text{ otherwise} \end{cases}$$

and

$$y_{23} = \begin{cases} 0 \text{ if the autoland function is accomplished during the landing phase} \\ 1 \text{ otherwise.} \end{cases}$$

Finally, we designate a single basic variable (weather) at the Intermediate 1 level which is denoted y_{32} and is interpreted as follows:

$$y_{32} = \begin{cases} 0 \text{ if the designated landing site does not have Cat III weather at the end of the cruise phase} \\ 1 \text{ otherwise.} \end{cases}$$

To summarize, the Intermediate 1 level variables can be described as a single matrix valued variable:

$$\mathbf{y} = \begin{bmatrix} y_{11} & y_{12} & y_{13} \\ (y_{21}) & y_{22} & y_{23} \\ (y_{31}) & y_{32} & (y_{33}) \end{bmatrix}$$

Composite variables

Basic variable

where y_{11}, y_{12}, y_{13}, y_{22}, y_{23} and y_{32} are defined above. The circled entries are variables whose values are irrelevant to this analysis and are assigned a constant value = ¢. The probabilistic nature of composite variables y_{1j}, y_{2j} is determined by that of lower level variables; the probabilistic nature of the basic variable y_{32} is determined by the non-computer part of the base model (see Section 3.2.2). In keeping with the general definition of our model hierarchy, the matrix value of **y** should suffice to determine the values of the composite variables at the top level. Furthermore, since there are no basic variables at the top level, we can resolve the **z** matrix and hence obtain the level of accomplishment.

The process of determining the level of accomplishment which results from a given value of **y** is part of the more general problem of formulating the capability function (see Section 3.3.2). To illustrate this connection, however, let us suppose that the aircraft is such that an active control level of 0 or 1 is required throughout the flight (see variables y_{1j}, i=1,2,3) for aircraft survival (i.e., without stability augmentation, the plane crashes). Suppose further that if the degree of active control drops from 0 to 1 or 2 any time before the landing phase, then fuel consumption is increased to the point where it is classified as "high." Finally, let us suppose that when Cat III weather exists at the intended landing site, if autoland capability is available (at the end of the cruise phase) the autoland system is used to attempt an automatic landing; if not available, the aircraft is diverted to an alternate landing site. If autoland is attempted but not accomplished, the aircraft crashes.

Intermediate 1	Mission	
$\mathbf{y} = \begin{bmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \\ y_{31} & y_{32} & y_{33} \end{bmatrix}$ (Phase 1, Phase 2, Phase 3; Active Control, Autoland, Weather)	$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}$ (Safety, Diversion, Fuel Consumption)	Level of Accomplishment
$\begin{bmatrix} 0 & 0 & 1 \\ ¢ & 1 & 1 \\ ¢ & 0 & ¢ \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$	A_1 (Total success)
$\begin{bmatrix} 0 & 1 & 1 \\ ¢ & 0 & 0 \\ ¢ & 1 & ¢ \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	A_2
$\begin{bmatrix} 0 & 0 & 0 \\ ¢ & 1 & 1 \\ ¢ & 1 & ¢ \end{bmatrix}$	$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$	A_3
$\begin{bmatrix} 1 & 1 & 1 \\ ¢ & 1 & 1 \\ ¢ & 1 & ¢ \end{bmatrix}$	$\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$	A_4
$\begin{bmatrix} 0 & 2 & 2 \\ ¢ & 1 & 1 \\ ¢ & 0 & ¢ \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$	A_5
$\begin{bmatrix} 0 & 0 & 0 \\ ¢ & 0 & 1 \\ ¢ & 1 & ¢ \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$	$\bar{A}_5$ (Total failure)

Table 1

Values of mission variables and level of accomplishment as determined by sample values of Intermediate 1 variables

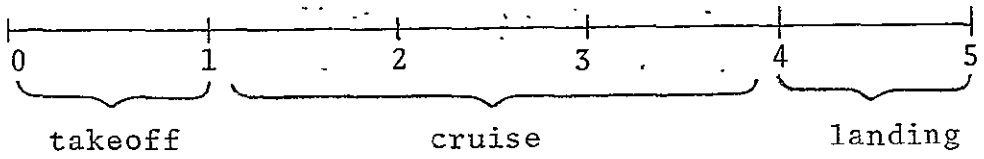
Under the above set of conditions, Table 1 gives the corresponding mission variable values and, subsequently, the level of (mission) accomplishment for several representative values of the Intermediate 1 model variables (i.e., the variable **y**). An exhaustive analysis of all 216 possible values of **y** shows that 10 values yield an accomplishment level equal to A_1, 30 values yield A_2, 4 yield A_3, 12 yield A_4 and 160 yield A_5. It should be noted, however, that these numbers have no direct bearing on the system's performability since the probabilistic nature of the matrix-valued random variable **y** has yet to be accounted for. Indeed, many of the values of **y** are "logically inconsistent" and hence have a zero probability of occurrence. For example, the value

$$\mathbf{y} = \begin{bmatrix} 2 & 2 & 2 \\ ¢ & 0 & 0 \\ ¢ & 1 & ¢ \end{bmatrix}$$

says that the aircraft crashes during takeoff, due to loss of active control, and yet autoland is accomplished during landing.

The next intermediate model in the hierarchy (Intermediate 2) is intended to represent the behavior of the computer per se, in terms of the computational tasks it performs throughout the utilization interval. The purpose of this model is to provide a description of computer behavior that is generally applicable to the class of fault-tolerant computers envisioned for use in the next generation of commercial aircraft. The Intermediate 2 model thus serves as common interface between specified, architecture dependent bottom models and the Intermediate 1 model.

During the reporting period, some effort was devoted to examining alternative types of representation which might be appropriate at the Intermediate 2 level, in preparation for a more detailed development which is just now underway. We can, however, illustrate the role of such a model via a simplified example which describes how computational tasks are accomplished during the utilization interval. More precisely, suppose that the utilization interval is divided into five computational periods as follows:



Suppose further that the duration of a computational task is taken to be the duration of the period during which the task is executed, and that there are four types of computation tasks:

1) Stability computations
2) Fuel regulation computations
3) Autoland computations
4) Internal computations (I/O management, on-line fault-detection, etc.).

If we let (i,j) denote the task which is of type i and is to be accomplished during the j^{th} period, then Intermediate 2 model can be taken to be a matrix-valued variable

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} \\ x_{31} & x_{32} & x_{33} & x_{34} & x_{35} \\ x_{41} & x_{42} & x_{43} & x_{44} & x_{45} \end{bmatrix}$$

where

$$x_{ij} \in \{0,1\} \text{ for } i \in \{1,2,4\} \text{ and } 1 \leq j \leq 5$$
$$\text{or for } i = 3 \text{ and } j \in \{4,5\}$$
$$\text{(i.e., for the non-circled entries)}$$

and

$$x_{ij} = \phi \quad \text{for } i = 3 \text{ and } 1 \leq j \leq 3$$
$$\text{(i.e., for the circled entries).}$$

The interpretation in the first case is that

$$x_{ij} = \begin{cases} 0 \text{ if task } (i,j) \text{ is accomplished} \\ 1 \text{ otherwise.} \end{cases}$$

(E.g., $x_{23} = 0$ means that fuel regulation computations were accomplished during period 3.) In the second case, ϕ is assigned to those variables whose values are irrelevant to the analysis.

The variable $\mathbf{x}$ can now be employed to determine the model Intermediate 1 composite variables. To describe this process, define the composite variable submatrix $\mathbf{y}_c$ of a variable matrix $\mathbf{y}$ to be the matrix composed of those rows of $\mathbf{y}$ which correspond to composite variables. For instance, given the Intermediate 1 level $\mathbf{y}$ discussed above:

$$\mathbf{y} = \begin{bmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \\ y_{31} & y_{32} & y_{33} \end{bmatrix} \begin{matrix} \} \text{ composite variables} \\ \\ \} \text{ basic variables} \end{matrix}$$

the $\mathbf{y}_c$ would be

$$\mathbf{y}_c = \begin{bmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \end{bmatrix}.$$

We can now make certain assumptions regarding the relations between computational tasks and aircraft functional tasks. First,

we assume that all aircraft tasks require the successful accomplishment of general internal computations as well as the computations specific to that aircraft task. Thus autoland, for instance, necessitates both autoland computations and internal computations.

Second, if the computational tasks required to perform an aircraft functional task are achieved, we assume that the functional task is accomplished. The need for this second assumption is due to the simplicity of this specific example. In general, the accomplishment of aircraft functional tasks can also depend on related aircraft systems as represented by additional Intermediate 2 model variables.

Finally, we make some assumptions with regard to how the three periods of the Intermediate 2 cruise phase relate to the single period of the Intermediate 1 cruise phase. If the stability, fuel regulation, or internal computations fail at any point during the cruise phase (e.g., if $(x_{22}, x_{23}, x_{24}) = (0,0,1)$), then those computations are unable to support their respective functional tasks represented by the Intermediate 1 composite variables. For instance, $(x_{12}, x_{13}, x_{14}) = (0,1,0)$ represents a stability computation that results in loss of stability augmentation during the cruise phase. Autoland computations, on the other hand, need to be available only at the end of the cruise phase. Therefore, we assume that the autoland computations are good if $x_{34} = 0$ and failed if $x_{34} = 1$. Thus, $(x_{32}, x_{33}, x_{34}) = (¢, ¢, 0)$ yields an autoland computation condition of 0 for the cruise phase. Note that the three periods of the Intermediate 2 cruise phase allow a closer examination of the computer's activities than would have been possible with only the one Inter-

medate 1 period. Indeed, this ability to alter the time scale from level to level is an important feature of the hierarchy since it permits a refinement of time as well as structural detail when descending the hierarchy.

With these assumptions, the values of the Intermediate 1 model composite variables $\mathbf{y}_c'$ are determinable from the computational outcomes $\mathbf{x}$. As an example of applying these results, consider the following matrix:

$$\mathbf{x} = \left[\begin{array}{c|ccc|c} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ ¢ & ¢ & ¢ & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right].$$

This says that the stability and internal computations were successful during the entire mission, the autoland computations were successful during the last two periods of the flight, while the fuel regulation computations failed during the last three periods of the flight. The above value of $\mathbf{x}$ yields the following value for the composite variables at the next higher level:

$$\mathbf{y}_c = \left[\begin{array}{ccc} 0 & 1 & 1 \\ ¢ & 0 & 0 \end{array}\right]$$

Here, $y_{12} = y_{13} = 1$ since the fuel regulation computations failed during the 2^{nd} and 3^{rd} phases. All other computations were successful; hence $y_{11} = y_{22} = y_{23} = 0$.

Table 2 shows some other possible values of $\mathbf{x}$ along with the corresponding values of $\mathbf{y}_c$. These outcomes follow naturally from the definitions of $\mathbf{x}$, $\mathbf{y}_c$ and the assumptions regarding the relationship of computational tasks to functional tasks.

Intermediate 2						Intermediate 1			
	Phase 1	Phase 2			Phase 3		Phase 1	Phase 2	Phase 3	
$\mathbf{x}$ =	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	Stability computations	$\mathbf{y}_c$ = y_{11}	y_{12}	y_{13}	Active control
	x_{12}				.	Fuel regulation computations	y_{21}	y_{22}	y_{23}	Autoland
	x_{13}				.	Autoland computations				
	x_{14}		. . .		x_{45}	Internal computations				

0	0	0	0	0	0 0 0
0	0	0	0	0	¢ 0 0
¢	¢	¢	0	0	
0	0	0	0	0	
0	0	0	0	0	0 1 0
0	1	0	0	0	¢ 1 0
¢	¢	¢	1	0	
0	0	0	0	0	
0	0	0	0	1	1 0 2
1	0	0	0	0	¢ 1 0
¢	¢	¢	1	0	
0	0	0	0	0	
0	0	0	0	0	1 1 1
1	1	1	0	1	¢ 0 1
¢	¢	¢	0	1	
0	0	0	0	0	
0	0	1	1	1	0 2 2
0	1	1	1	1	¢ 1 1
¢	¢	¢	1	1	
0	0	1	1	1	

Table 2

Values of Intermediate 1 composite variables as determined by sample values of Intermediate 2 model variables

3.2 Computer Models

The objective of this effort is to delineate an appropriate class of stochastic models that can serve as bottom level models in the hierarchy. The scope of a model in this class is some specified aircraft computer and the level of abstraction is the "operational state" of the computer's hardware and software. Accordingly, this class of models must be general enough to cover a variety of computer architectures of the type being considered for use in advanced commercial aircraft. At the same time, this class must be specific enough to permit the study of how a bottom level model relates to higher level models in the hierarchy.

This effort was initiated during the previous reporting period and, during the current period, we have continued our examination of Markov models that have recently been employed as computer models for the reliability analysis of fault-tolerant computing systems (see [11]-[15], for example). We have found that such models are compatible with the hierarchy in the sense that the state behavior of a model can be used to determine whether the system is able to accomplish a higher level task or mission. However, we have also found that in order to formulate these higher tasks and missions in terms of the system's operation and environment, the model should incorporate a concept of state that is capable of representing more than just the operational status of various components. As a consequence, the resulting Markov model may require an enormous state space, even for a moderately complex computing system.

In order to keep the size of the state space manageable,

we have examined possible ways of extending the concept of a stationary Markov process, the results of which are summarized in the subsections that follow.

3.2.1 A Non-Stationary Markov Process

As an extension of the traditional Markov models with stationary transition probabilities, we have examined a class of stochastic models which can be represented as finite-state non-stationary Markov processes. Since the transition probabilities of each model are presumably stationary during each phase, each of these models can be regarded as a finite sequence of stationary Markov processes, where each process in the sequence has a fixed duration.

The reliability analysis of phased missions has been studied in the past, but most of the previous work (see [6] and [8], for example) considers the case where interphase transitions are deterministic. For systems with non-repairable and statistically independent components, a general treatment of the problem of interphase dependencies was recently provided by Esary and Ziehms (see [5] and [7]). In their approach, a mission is represented by a set of fault trees, each of which denotes the computational requirements of the system during a specific phase. Each mission is then transformed into a single synthetic fault tree which can then be evaluated using the usual fault tree techniques. Although this approach may have some value from a conceptual point of view, it is of little practical use when applied to systems having the complexity of an aircraft computer. This is due to the fact that i) it assumes that the operational state model is the same throughout the utilization interval (only the structure function can

change from phase to phase), ii) the local state sets of the subsystems are two-valued (i.e., operational state are described at the component level), and iii) the approach relies on a principle of "composition" rather than "decomposition" which increases the size of the equivalent model. As a consequence of these facts, the size of the resulting fault tree is unmanageable, even for systems of moderate complexity.

An alternative to this "composition" approach is to perform a phase-by-phase analysis which accounts for the probabilistic nature of interphase dependencies. An examination of this alternative was initiated during the reporting period; the results obtained to date are discussed in the subsections that follow.

3.2.1.1 Model Description

We suppose first that the utilization interval $T = \{t | t_o \leq t \leq t_k\}$ is decomposed into k consecutive intervals $T_1, T_2, \ldots, T_k$ where $T_m = \{t | t_{m-1} \leq t \leq t_m\}$ and $t_o < t_1 < \ldots < t_k$. The set of time points $\{t_m | m = 0,1,\ldots,k\}$ is fixed for a given mission since the mission profile of the aircraft is assumed to be known in advance. Henceforth, each time interval T_m will be referred to as the m^{th} phase of the mission.

Given a utilization interval T, we suppose further that the probabilistic nature of the computing system to be evaluated is described by a finite-state stochastic process

$$Y_S = \{Y_S(t) | t \in T\}$$

where, for a given t, $Y_S(t)$ is a random variable (defined on an underlying probability space (Ω,F,P)) that takes on values in the state space Q; (i.e., $Y_S(t):\Omega \rightarrow Q$). The process is assumed to be a Markov process with transition probability

$$\Pr[Y_S(t+r) = j | Y_S(t'); t' \leq t] = \Pr[Y_S(t+r) = j | Y_S(t)].$$

The conditional probability above, may, in general depend on both t and r (in addition to j and the value of $Y_S(t)$). However, we suppose that the transition probabilities are stationary within each phase, i.e., given phase $T_m = [t_{m-1}, t_m]$,

$$\Pr[Y_S(t+r)=j | Y_S(t) = i] = P_r^m(i,j)$$

is independent of t for $t_{m-1} \leq t \leq t_m$, $t_{m-1} \leq t+r \leq t_m$, $r > 0$, $i,j \in Q$ and $m = 1,2,\ldots,k$.

Consider a simple example. A typical aircraft flight may consist of three phases - take-off, cruise and landing. Assume that the flight control on-board computer consists of four functionally independent identical units. Different units may or may not compute the same function at the same time depending on the amount of computation needed and the safety requirements of each phase. As an illustration, the system may assume a TMR configuration with one standby unit for the take-off. During the cruise phase, the system may require only a duplex-simplex configuration with two standby spare units. To meet the high computational requirements of landing, the system may require that three units operate concurrently (to support different tasks) with only one spare unit. A conventional reliability analysis of each of these configurations typically employs the concept of a stationary Markov process (see [16]-[18]).

In the following discussion, we allow each phase to choose from a list of possible system configurations depending on the outcome of the previous phases. Let $C^m = \{C_{m1},\ldots,C_{mn}\}$ be the set of possible configurations associated with the m^{th} phase T_m ($m = 1,2,\ldots,k$). Then given $t_{m-1} \leq t \leq t_m$, the state

$Y_S(t)$ of the system at time t may represent both the operational status of various components and any condition that is important to consider in the performability analysis. For example, a specific state of the model might represent that the system in configuration C_i has j subsystems failed and the rest of the subsystems are functioning in a degraded mode. Conceptually, the Markov process Y_S can be viewed as having a single large state space. From a practical point of view, however, for each phase we need only represent those system states that are possible (i.e., have non-zero probability) during that phase. Given such a model whose probabilistic structure varies from phase to phase, the question that remains is how the results of these per-phase analyses can be combined so as to adequately support an analysis of the system's performability at the mission level. This question is addressed in the subsections that follow.

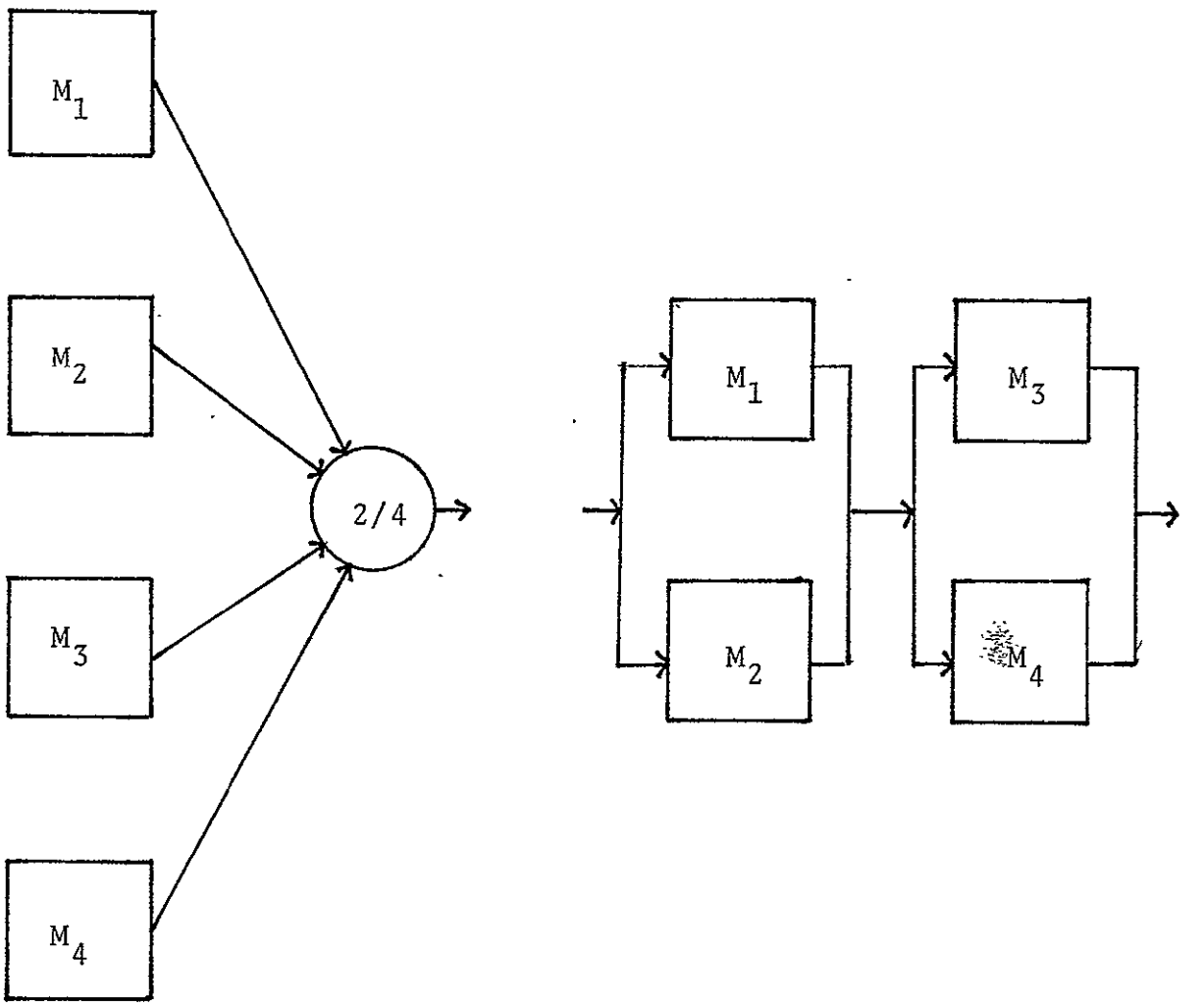
3.2.1.2 Dependencies Between Phases

To illustrate the nature of the problems encountered when combining the results of a phase-by-phase analysis, it suffices to consider a traditional two-valued mission model wherein a mission either is accomplished ("success") or is not ("failure"). In this case "performability" (see Section 3.3) reduces to the usual notion of "reliability" (probability of success). However, given that the computer is represented by a time-varying model - consisting of a sequence of stationary Markov processes (one process per phase with each phase having a fixed duration), the reliability analysis is complicated by the fact that interphase dependencies must be accounted for. Such dependencies are due to the fact that certain parts of the hardware and software

structure of the computer may be used during more than one phase of the mission.

To illustrate the above remarks, consider the following hypothetical situation. A system with four identical modules M_1, M_2, M_3 and M_4 is designed for a two-phase mission. In order for the system to perform the required tasks, at least two modules must function through phase 1. After phase 1 has been completed, the computational requirements change and the system is reconfigured as a series connection of two duplex stages (see Figure 4). Thus the second phase of the mission is a success if at least one module in each duplex stage remains functioning through phase 2. Suppose that each of M_1, M_2, M_3 and M_4 fail permanently with a constant failure rate λ. Suppose further that the failure characteristics of the modules are statistically independent and no repair is possible throughout the mission. Then the probabilistic nature of phase 1 and phase 2 can be represented, respectively, by finite-state stationary Markov processes with transition graphs as illustrated in Figure 5 and Figure 6.

Note that this phase-by-phase Markov representation enables us to choose different sets of model variables for each phase. Thus, in general, the construction of a particular phase can be tailored to both the structure of the computer and the nature of the mission requirements during that phase, thereby reducing the number of model variables at each level in the hierarchy. Using the above example, let us now examine some of the problems encountered when using such a model to analyze



phase 1 = $[t_0, t_1]$ phase 2 = $[t_1, t_2]$

Figure 4

A two-phased mission

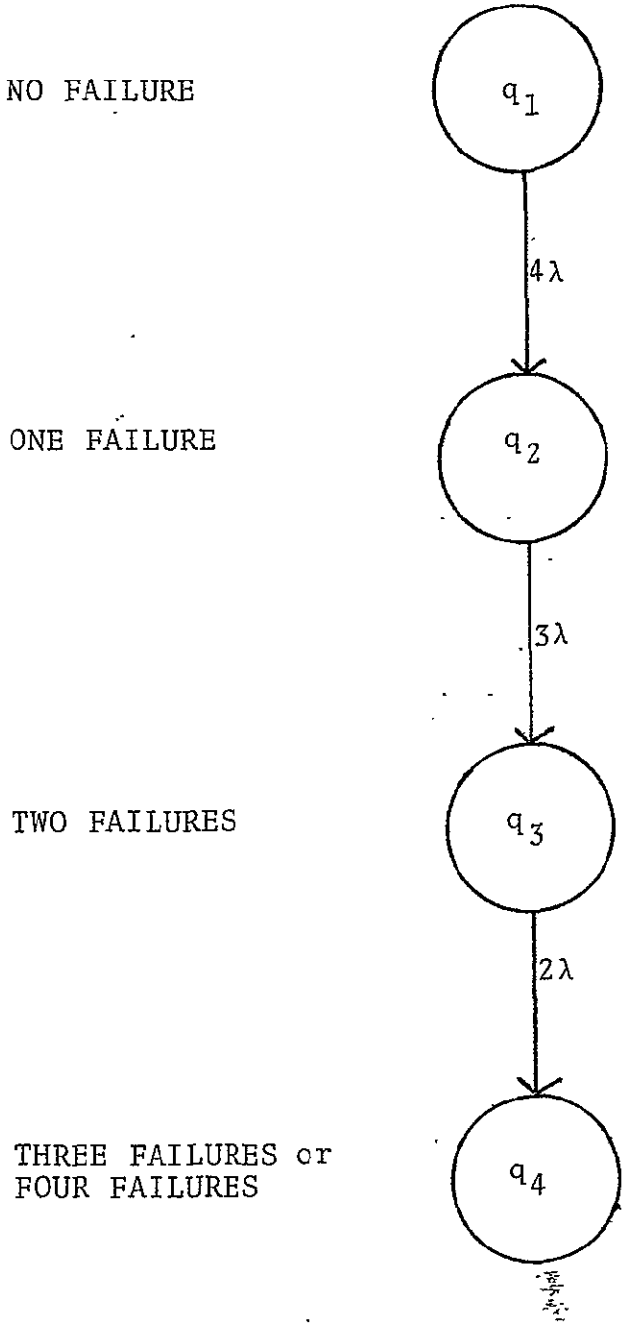


Figure 5

Markov model for a two-out-of-four system

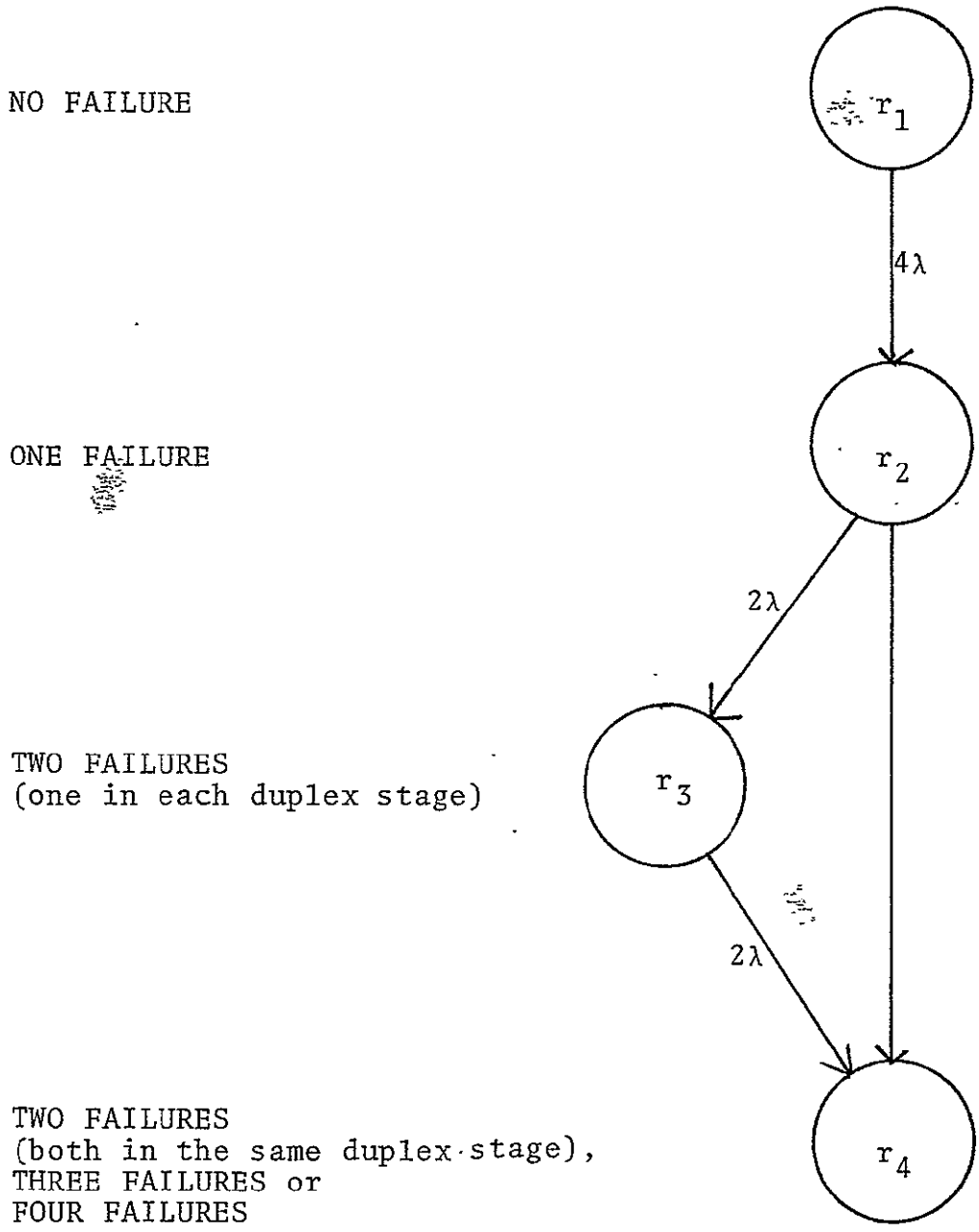


Figure 6

Markov model for a double duplex system

system performability or (since we have only two levels of performance in this case) reliability.

If we suppose that all modules are functioning at the beginning of the mission (and hence the beginning of phase 1), i.e., $Pr[Y_S(t_o) = q_1] = 1$, then the reliability of the system during phase 1 is given by

$$Rel(phase\ 1) = Pr[Y_S(t_1) \in R_1 | Y_S(t_o) = q_1]$$

where R_1 is the set of "success states" of the phase 1 model, i.e., $R_1 = \{q_1, q_2, q_3\}$. Expanding this equation, we have

$$\begin{aligned} Rel(phase\ 1) &= Pr[Y_S(t_1)=q_1 | Y_S(t_o)=q_1] + Pr[Y_S(t_1) = q_2 | Y_S(t_o)=q_1] \\ &\quad + Pr[Y_S(t_1)=q_3 | Y_S(t_o)=q_1] \\ &= e^{-4\lambda(t_1-t_o)} + 4(1-e^{-\lambda(t_1-t_o)})e^{-3\lambda(t_1-t_o)} \\ &\quad + 6(1-e^{-\lambda(t_1-t_o)})^2 e^{-2\lambda(t_1-t_o)} \\ &= 6e^{-2\lambda(t_1-t_o)} - 8e^{-3\lambda(t_1-t_o)} + 3e^{-4\lambda(t_1-t_o)}. \end{aligned}$$

If we now consider phase 2, its reliability can be similarly expressed as

$$Rel(phase\ 2) = Pr[Y_S(t_2) \in R_2 | I]$$

where $R_2 = \{r_1, r_2, r_3\}$ (the "success states" of the phase 2 model) and I is some assumed condition regarding the initial state of phase 2. More generally, the initial condition may be distributed probabilistically over several mutually exclusive possibilities $I_1, I_2, \ldots, I_S$, in which case

$$Rel(phase\ 2) = \sum_{i=1}^{S} Pr[Y_S(t_2) \in R_2 | I_i] \cdot Pr[I_i].$$

The simplest choice of I is similar to the one made for phase 1,

namely $Y_S(t_1) = r_1$, i.e., the initial state for phase 2 is the most favorable operational state (no failures). If we pursue this choice, we have:

$$\begin{aligned} \text{Rel(phase 2)} &= \Pr[Y_S(t_2) \in R_2 | Y_S(t_1) = r_1] \\ &= \Pr[Y_S(t_2) = r_1 | Y_S(t_1) = r_1] \\ &+ \Pr[Y_S(t_2) = r_2 | Y_S(t_1) = r_1] \\ &+ \Pr[Y_S(t_2) = r_3 | Y_S(t_1) = r_1]. \end{aligned}$$

Computing each of these probabilities and combining the terms, we obtain:

$$\text{Rel(phase 2)} = 4e^{-2\lambda(t_2-t_1)} - 4e^{-3\lambda(t_2-t_1)} + e^{-4\lambda(t_2-t_1)}.$$

Finally, given the per-phase reliabilities determined above, we might be tempted to express the total system (mission) reliability as the product of the phase reliabilities, that is:

$$\text{Rel(mission)} = \text{Rel(phase 1)} \cdot \text{Rel(phase 2)}.$$

Then, assuming the durations of phase 1 and phase 2 are the same, i.e., $(t_1-t_o) = (t_2-t_1) = T/2$, the mission reliability is:

$$\text{Rel(mission)} = 24e^{-2\lambda T} - 56e^{-2.5\lambda T} + 50e^{-3\lambda T} - 20e^{-3.5\lambda T} + 3e^{-4\lambda T}.$$

When the above expression is compared with an exact expression of the mission reliability (derived in the following subsection), i.e.,

$$\text{Rel(mission)} = 4e^{-2\lambda T} - 4e^{-3\lambda T} + e^{-4\lambda T},$$

we see that the above derivation is inaccurate and provides an overly optimistic view of the system's reliability. A closer examination of the derivation reveals that the cause of this discrepancy is twofold: i) the assumption regarding the initial state of phase 2 is incorrect and ii) the events

"phase 1 success" and "phase 2 success" are statistically dependent and hence the product of their probabilities is not equal to the probability of the joint event "mission success."

To correct for each of these errors, we must account for certain ways in which successive phases depend on one another. Our work in this regard is discussed in the subsection that follows.

3.2.1.3 Interphase Transitions

In general, complexities in the performability analysis of phased missions arise because the performance of a module depends on its performance during previous phases. These dependencies are of a special type, however, since temporal dependencies within a phase satisfy the Markov condition. Hence, if t_m is the time of transition between phase m and phase m+1, it suffices to determine how the initial state of phase m+1 (at time t_m) depends on the final state of phase m (at time t_m). In general, the nature of such dependencies will be probabilistic (for reasons which will be explained in a moment) and can be represented as follows.

Let Q^m denote the state set of the Markov model representation of the m^{th} phase (m=1,2,...,k) and suppose each state set is ordered so we can speak of the i^{th} state of Q^m, where if $|Q^m| = n_m$ then $1 \leq i \leq n_m$. With each successive pair of phases m and m+1 ($1 \leq m < k$) we associate an interphase transition matrix H(m), defined to be an n_m by n_{m+1} matrix

$$H(m) = [h_{ij}]$$

where

h_{ij} = probability that the state of the phase m+1 model is j (at time t_m) given that the state of the phase m model is i (at time t_m).

Note that H(m) is a stochastic matrix, i.e.,

$$\sum_{j=1}^{n_{m+1}} h_{ij} = 1, \quad i = 1,2,\ldots n_m.$$

Note also that H(m) reduces to the identity matrix (1's on the diagonal; 0's elsewhere) in the case where the phase m and phase m+1 models are identical and have the same interpretation.

The probabilistic nature of these interphase transitions is due to the fact that, in general, our knowledge of the system at the end of phase m, as conveyed by the state of the phase in model, may lack the detail needed to uniquely determine the state of system as it is newly represented by the phase m+1 model. The information that is lacking may be information about the computer, per se, or may be information which lies outside the scope of the computer model.

To illustrate this point, consider the example discussed in the previous subsection. In this case we have two phases with state sets

$$Q^1 = \{q_1, q_2, q_3, q_4\}$$

and

$$Q^2 = \{r_1, r_2, r_3, r_4\}$$

respectively. If, at time t_1, the system is in state q_3 with respect to the phase 1 model (i.e., two module failures) then, depending on which two modules failed, the state of the system with respect to the phase 2 model is either r_3 (one module failure in each duplex stage) or r_4 (two module failures in the same duplex stage). As module failures (in this example) are independent

and equally likely, a straightforward calculation reveals that $h_{33} = 2/3$ and $h_{34} = 1/3$. Transitions from states other than q_3 happen to be deterministic, and thus we obtain the following interphase transition matrix:

$$H(1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2/3 & 1/3 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Using such interphase transition matrices, we have started to investigate the development of more precise formulations of system performability in terms of the per-phase nature of the bottom model. In particular, for two-valued mission models wherein "mission success" is defined to be "success of every phase of the mission," we have succeeded in deriving an exact expression of mission reliability.

For each phase of the mission, let P(m) denote the initial-to-final state transition matrix of the m^{th} phase, i.e.,

$$P(m) = [p_{ij}(m)]$$

where

$$p_{ij}(m) = \Pr[Y_S(t_m) = j \mid Y_S(t_{m-1}) = i].$$

For each phase except the final phase, let G(m) denote the success state matrix of the m^{th} phase ($1 \leq m < k$), i.e.,

$$G(m) = [g_{ij}(m)]$$

where $g_{ij}(m) = \begin{cases} 1 \text{ if } i=j \text{ and } i \in R_m \\ 0 \text{ otherwise.} \end{cases}$

For the final phase (m=k) we define a success state vector

$$F(k) = \begin{bmatrix} f_1(k) \\ \vdots \\ f_{n_k}(k) \end{bmatrix}$$

where $f_i(k) = \begin{cases} 1 \text{ if } i \in R_k \\ 0 \text{ otherwise.} \end{cases}$

Finally let I(0) denote the initial state distribution for phase 1, that is

$$I(0) = [p_1(0), \ldots, p_{n_1}(0)]$$

where

$$p_i(0) = \Pr[Y_S(t_o)=i], \quad 1 \leq i \leq n_1.$$

Then it can be shown that mission reliability (probability of mission success) can be formulated as follows:

$$\text{Rel(mission)} = I(0)\left(\prod_{i=1}^{k-1} P(i)G(i)H(i)\right)P(k)F(k),$$

where the product operation is matrix multiplication.

Note that in the special case of a one phase mission (k=1), the expression reduces to

$$\text{Rel(mission)} = I(0)P(1)F(1).$$

Here, I(0)P(1) is a vector of final state probabilities. Multiplication by F(1) selects those states which are success states and sums their probabilities, the result being "probability of success" (relative to initial distribution I(0)).

To illustrate a less trivial application of the formula, consider once again the two-phase example for which we derived the interphase transition matrix H(1). In this case

$$\text{Rel(mission)} = I(0)P(1)G(1)H(1)P(2)F(2)$$

where we will suppose that

$$I(0) = [1\ 0\ 0\ 0]$$

(i.e., we begin with no failures) and where P(1) and P(2) are obtained by the usual methods of stationary Markov model

analysis (see [20], for example). The remaining matrices are

$$G(1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$H(1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2/3 & 1/3 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

and

$$F(2) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

If we let $T_1 = (t_1 - t_o)$ and $T_2 = (t_2 - t_1)$ be the durations of phases 1 and 2, respectively, and we iteratively compute the matrix product, beginning from the left, then for the first two terms we have:

$$A_1 = I(0)P(1) = [a_{11}\ a_{12}\ a_{13}\ a_{14}]$$

where

$$a_{11} = e^{-4\lambda T_1},$$

$$a_{12} = 4e^{-3\lambda T_1} - 4e^{-4\lambda T_1},$$

$$a_{13} = 6e^{-2\lambda T_1} - 12e^{-3\lambda T_1} + 6e^{-4\lambda T_1},$$

$$a_{14} = 1 - 6e^{-2\lambda T_1} + 8e^{-3\lambda T_1} - 3e^{-4\lambda T_1}.$$

The interpretation of a_{1i} is the probability that the final state of phase 1 is q_i (given the initial state distribution $I(0) = [1\ 0\ 0\ 0]$).

The next partial product is the result of multiplying A_1 by the success state matrix G(1) which yields:

$$A_2 = A_1 G(1) = [a_{21}\ a_{22}\ a_{23}\ a_{24}]$$

where

$$a_{21} = a_{11},$$
$$a_{22} = a_{12},$$
$$a_{23} = a_{13},$$
$$a_{24} = 0.$$

Thus vector A_2 is the same as A_1 for those states of the phase 1 model that guarantee phase 1 success. The remaining entries (those corresponding to phase 1 failure states) are 0. More precisely the interpretation of a_{2i} is the probability that the final state of phase 1 is q_i and phase 1 is a success.

The third partial product is a result of multiplying A_2 by the interphase transition matrix H(1) which yields:

$$A_3 = A_2H(1) = [a_{31}\ a_{32}\ a_{33}\ a_{34}]$$

where

$$a_{31} = e^{-4\lambda T_1},$$

$$a_{32} = 4e^{-3\lambda T_1} - 4e^{-4\lambda T_1},$$

$$a_{33} = 4e^{-2\lambda T_1} - 8e^{-3\lambda T_1} + 4e^{-4\lambda T_1},$$

$$a_{34} = 0.$$

The purpose of this operation is to describe the results of the phase 1 analysis in terms of the phase 2 model, where the interpretation of entry a_{3i} is the probability that the initial state of phase 2 is r_i and phase 1 is a success.

The fourth partial product is the result of multiplying A_3 by the transition matrix P(2) of phase 2, that is:

$$A_4 = A_3P(2) = [a_{41}\ a_{42}\ a_{43}\ a_{44}]$$

where, if $T = T_1+T_2$ then

$$a_{41} = e^{-4\lambda T},$$

$$a_{42} = 4e^{-3\lambda T} - 4e^{-4\lambda T},$$

$$a_{43} = 4e^{-2\lambda T} - 8e^{-3\lambda T} + 4e^{-4\lambda T},$$

$$a_{44} = 1 - 4e^{-2\lambda T} + 4e^{-3\lambda T} - e^{-4\lambda T}.$$

Thus the interpretation of a_{4i} is the probability that r_i is the final state of phase 2 and phase 1 is a success.

The product is completed by multiplying A_4 by the success state vector F(2) of the final phase, that is,

$$\begin{aligned} \text{Rel(mission)} &= A_4F(2) \\ &= \sum_{i\varepsilon R_2} a_{4i} \\ &= 4e^{-2\lambda T} - 4e^{-3\lambda T} + e^{-4T}. \end{aligned}$$

Since the sum is taken over all final states of phase 2 that guarantee phase 2 success, the interpretation of the sum is the probability of phase 2 success and phase 1 success, i.e., the probability of mission success (given the initial state distribution I(0) = [1 0 0 0]).

The above example serves not only to illustrate an exact computation of mission reliability but also to give an informal justification of why this method produces the desired result. As a check on the computation, we note that this simple example could be viewed equivalently as a double duplex system throughout its utilization interval and, when so viewed, yields the result obtained above. This is not to suggest that multiphase models can generally be reduced to single phase models; indeed, we

believe that the above example is quite unusual in this regard.

Although the above computational algorithm applies only to a restricted set of mission level models, it establishes in our opinion, the feasibility of using non-stationary Markov models as base models for performability analysis. It should also be noted that the above algorithm need not be restricted to base models where the intraphase processes are Markovian, as long as the overall sampled process (where a sample is made at the end of each phase) is Markovian. Thus, semi-Markov processes or approximate Markov processes (see [19]-[23]) can also be used to model the intraphase behavior. Thus we intend to pursue this approach for more general types of mission models via an analysis of "R-dependencies" associated with various levels of mission accomplishment. Our work in the latter regard is discussed in the section that follows.

3.3 Formulation of System Effectiveness

The central idea that underlying this research project is that the evaluation of system reliability and performance should not be treated as separate issues but, instead, as a single issue which can generally be referred to as "system effectiveness." Informally, "system effectiveness" is the extent to which the user may expect to benefit from the missions accomplished by the system in the use environment. Thus effectiveness measures for aircraft computers must quantify the extent to which a commercial air carrier may expect to benefit from missions accomplished by an aircraft computer (in conjunction with cooperating related systems

and supporting systems). As discussed in the first Semi-Annual Status Report (see [1], Section 3.3.4.2), the formulation of such measures can be naturally decomposed into two problems:

i) Formulating the probabilities of accomplishing various types and qualities of missions, and

ii) Formulating the worth (benefit) associated with accomplishing various types and qualities of missions.

As justified in the previous Status Report, we have chosen to focus our attention on the first of these two problems, i.e., the problem of formulating measures of system "performability." More precisely, a performability measure can be regarded as a special type of effectiveness measure wherein the worth of a performance is equated with the performance itself (as described by the top model). Recalling the WSEIAC definition of effectiveness (see [24]),

> System effectiveness is a measure of the extent to which a system may be expected to achieve a set of specific mission requirements. It is a function of the system's availability, dependability, and capability

it follows that performability can likewise be decomposed into measures of availability, dependability, and capability. In terms of our model hierarchy, the first two measures (availability and dependability) quantify the behavior of the bottom model. The third measure (capability) quantifies the behavior of the top model as a function of values assumed by "basic" variables of the bottom and intermediate models (see Section 3.1.2.2). Thus the capability aspect of performability invokes the entire model hierarchy and, indeed, is the reason for the hierarchy's existence.

During the reporting period, we have developed a precise notion of capability and have started to investigate its properties

in terms of a generalized notion of functional dependency. This work is reported in the subsections that follow.

3.3.1 Structure Functions and Dependency

3.3.1.1 Structure Functions

In structure-based analysis, the system to be evaluted (call it S) is regarded as a network of subsystems (components). For each subsystem S_i there is associated a set Q_i composed of the operational states of S_i. These operational states represent the various fault conditions of S_i. In the simplest case, $Q_i = \{0,1\}$ where a "0" indicates that S_i is fault-free, and a "1" indicates that S_i is faulty. The success of the system S is then related to the operational states of the subsystems S_i by a binary-valued function

$$\phi: Q_1 x Q_2 x \ldots x Q_n \rightarrow \{0,1\}$$

where

$$\phi(q_1, q_2, \ldots q_n) = \begin{cases} 0 \text{ if S is a "success" in operational state } (q_1, q_2, \ldots, q_n) \\ 1 \text{ otherwise.} \end{cases}$$

Such a function is called a structure function (see [1], [25]). (Technically, the above definition is the "dual" of the traditional definition of a structure function, since we interpret 0 (rather than 1) as "success." We find the dual definition to be more convenient when it comes time to extend the concept to multiple levels of system performance.)

The limitations of the structure function approach have been discussed elsewhere (see [1], [26]), but two points deserve reiteration. First, the fact that structure functions are binary-valued disallows adequate handling of modes of degraded perform-

ance. Second, structure-based analysis regards the network of subsystems as a combinational, or memoryless, network. This point of view does not allow the treatment of interactions between components over time. Given these limitations, a new approach is required.

3.3.1.2 Dependency

Before formally defining the concept of a capability function, consider the notion of dependency. In general, there are two modes in which one thing can depend on another [27]. In the first mode, knowing that A depends on B and knowing everything of interest about B tells everything of interest about A. This mode of dependence is exemplified by linear dependence. In the second mode, knowledge of B coupled with the knowledge that A depends on B tells us something (but not necessarily everything) about A. The best example here is the idea of statistical dependence. This is the mode of dependence which will generally occur in the study of complex systems.

It is important to note that knowledge of certain dependencies between subsystems (of the system) of interest may help in several ways. In classical reliability analysis, for example, knowledge that two subsystems fail independently allows them to be decoupled and studied separately. Probabilistically, if the failures of S_1 and S_2 are independent, then

$$P(S_1 \text{ fails and } S_2 \text{ fails}) = P(S_1 \text{ fails}) \cdot P(S_2 \text{ fails}).$$

It is not the case, however, that all forms of dependency are bad. For instance, knowledge of certain dependencies between the operational states of a system over time may allow the simplification of considering the states of the system only at specific times. Given that the appropriate forms of dependency exist, then, observation of the system can be limited without a loss of relevant knowledge. One example of this is

when assumptions are made concerning the coherence and pure-death properties of a system. If a system has these properties, then knowledge that its final state is fault-free tells that at all previous times the system was fault-free. Thus the assumption of coherence and a pure-death model is actually an assumption about the temporal dependencies inherent in a particular system. Knowing which dependencies exist may help to simplify analysis. In general, functional independence and temporal dependence appear to be simplifying factors.

3.3.1.3 Functional Dependence

In [1], the CARSRA notion of functional dependence (ϕ-dependence) was formalized as follows. Given a system S with component subsystems $S_1,\ldots,S_n$, state set $Q=Q_1xQ_2x\ldots xQ_n$ (Q_i is the state set of S_i), and structure function ϕ, let

$$R_\phi = \{q|\phi(q) = 0, q \in Q\}.$$

Then R_ϕ is the set of all success states of S relative to the structure function ϕ. For $q \in Q$, let $\xi_i(q)$ denote the value of the i^{th} coordinate of q, i.e. if $q = (q_1,\ldots,q_n)$ then $\xi_i(q) = q_i$. We define

$$D_\phi(i) = \xi_i(R_\phi) = \{\xi_i(q)|q \in R_\phi\}.$$

$D_\phi(i)$ is the projection of R_ϕ on the i^{th} coordinate. We also define, for $1 \leq j \leq n$ and $q_j \in D_\phi(j)$

$$R_\phi(j,q_j) = \{q \in R_\phi|\xi_j(q) = q_j\} \text{ and}$$

$$D_\phi(i,j,q_j) = \xi_i(R_\phi(j,q_j)).$$

Informally, $R_\phi(j,q_j)$ is the subset of R_ϕ comprised of all the elements of R_ϕ whose j^{th} coordinate is equal to q_j. This means that $D_\phi(i,j,q_j)$ is the result of first selecting all the elements

of R_ϕ whose j^{th} coordinate is q_j, and then taking the projection of this set on the i^{th} component.

For example, consider the three stage plus independent voter TMR system with binary-valued state sets which can be represented as in Figure 7.

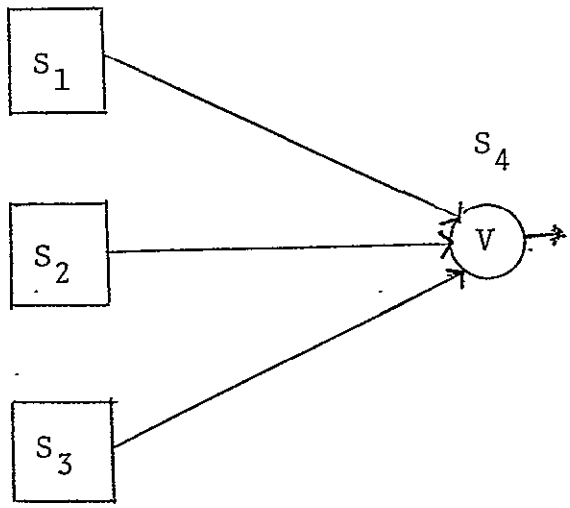


Figure 7

Then

$$R_\phi = \begin{Bmatrix} (0,0,0,0) \\ (1,0,0,0) \\ (0,1,0,0) \\ (0,0,1,0) \end{Bmatrix}.$$

Clearly, $D_\phi(1) = \{0,1\}$ and $D_\phi(4) = \{0\}$. In addition, examples of $R_\phi(j,q_j)$ are

$$R_\phi(2,0) = \begin{Bmatrix} (0,0,0,0) \\ (1,0,0,0) \\ (0,0,1,0) \end{Bmatrix}, \quad R_\phi(2,1) = \{(0,1,0,0)\}$$

so

$$D_\phi(1,2,0) = \xi_1(R_\phi(2,0)) = \{0,1\}, \text{ and}$$
$$D_\phi(1,2,1) = \{0\}.$$

The formal definition of functional dependence is stated as:

Definition: If S_i and S_j are subsystems of a system with structure function ϕ then S_i ϕ-depends on S_j if, for some state $q_j \in D_\phi(j)$, $D_\phi(i,j,q_j) \neq D_\phi(i)$.

From this definition we see that in the above example S_1 ϕ-depends on S_2 because

$$\{0,1\} = D_\phi(1) \neq D_\phi(1,2,1) = \{0\}.$$

Examination quickly shows also that S_1 ϕ-depends on S_3 and that S_2 ϕ-depends on S_3. On the other hand, since the state of S_4 is constant, S_4 is ϕ-independent of S_1, S_2 and S_3. Calculation of the possible sets shows this to be true. In fact, S_4 is "universally independent" in the sense that no other subsystem depends on it [27].

Example

Consider now the triplex system (Figure 8) discussed in [1] (see also [12], figure 8). As presented therein, the system is regarded as being composed of four subsystems S_1, S_2, S_3 and S_4 called "stages." Each stage is comprised of 3 "modules" (see [12]) and is represented by a finite-state Markov process with a transition graph as illustrated in Figure 9. Thus, the state set for stage S_i is $Q_i = \{1,2,3,4,5\}$ $(i = 1,2,3,4)$ and the structure function, in this case, is the function

$$\phi: \{1,2,3,4,5\}^4 \rightarrow \{0,1\}$$

where

$$\phi(q) = \begin{cases} 0 \text{ if, when the system is in state } q, \\ \quad \text{the voter can make use of the outputs of at least} \\ \quad 2 \text{ fault-free modules in each stage} \\ 1 \text{ otherwise.} \end{cases}$$

Here $q = (2,1,1,1)$ has the interpretation that S_1 (stage 1) has one faulty module, while S_2, S_3, and S_4 are fault-free.

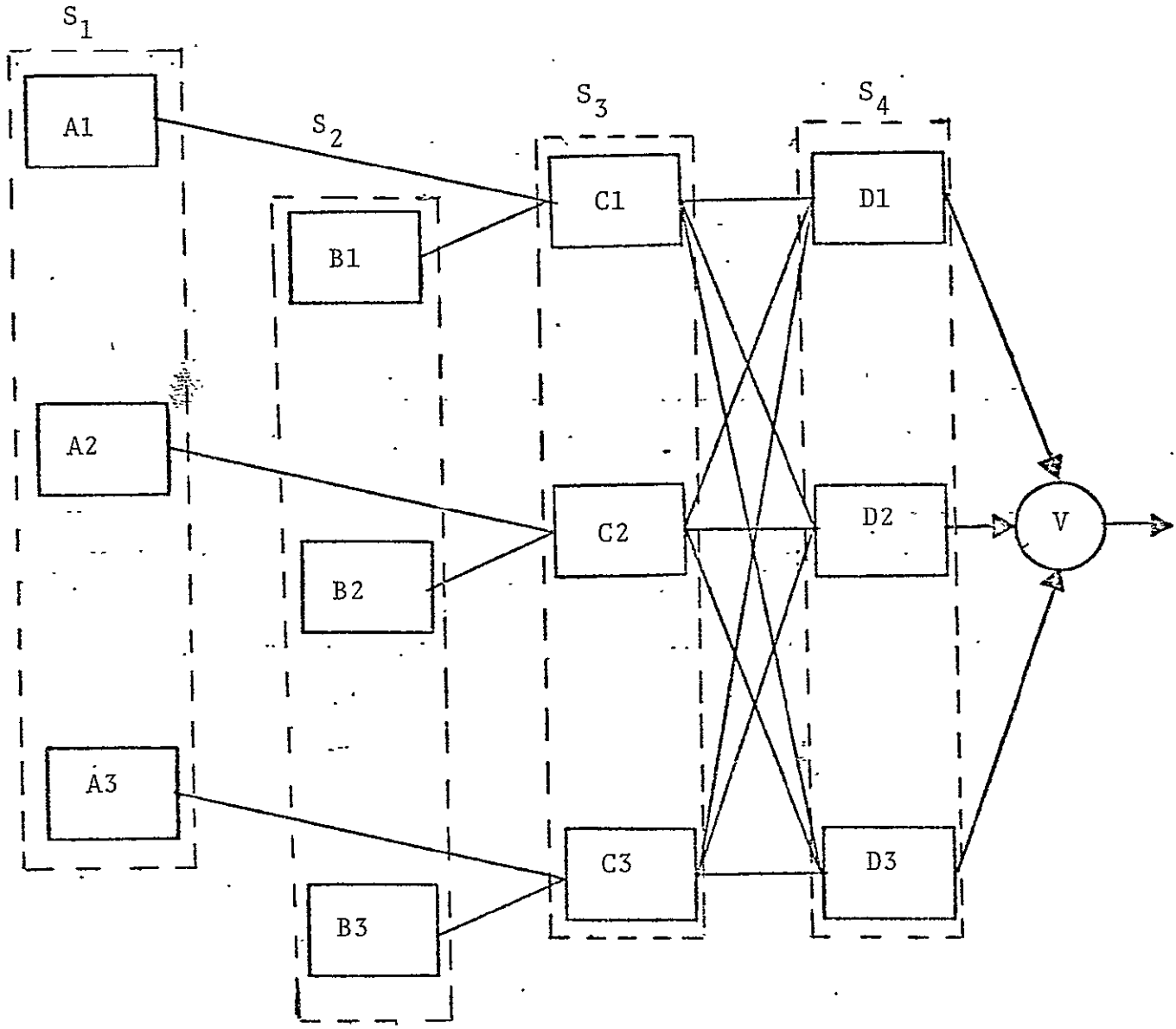


Figure 8

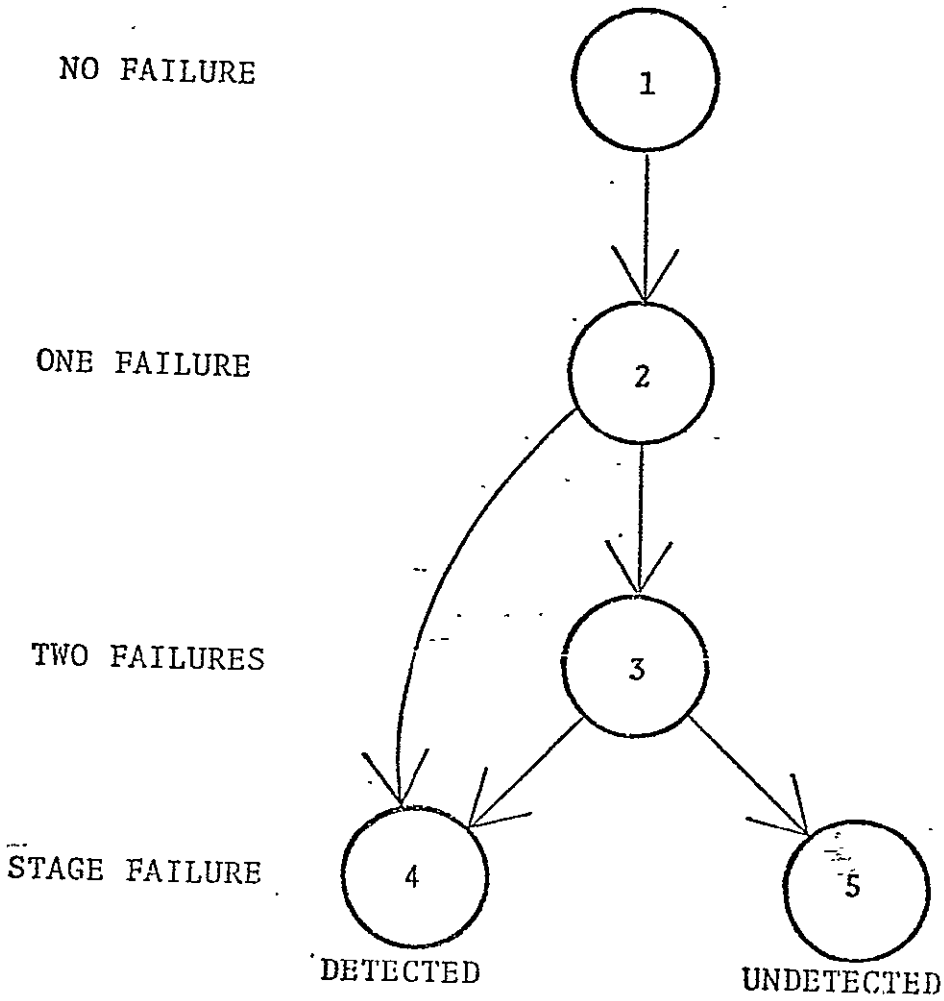


Figure 9
Stage Markov model

Some examples of the evaluation of ϕ are

$\phi(2,1,1,1) = 0,$
$\phi(3,1,1,1) = 1,$ but
$\phi(2,2,2,2) = 0$ or 1 depending on which modules are faulty.

This means that in this case, the structure function ϕ is actually a structure relation. Let

$$R_\phi = \{q \mid \phi(q) = 0 \text{ (and } \phi(q) \neq 1)\}, \; q \in Q_1 \times Q_2 \times Q_3 \times Q_4\}.$$

The set R_ϕ is composed of all the unambiguous success states of the system S relative to the structure relation ϕ. As shown in [1],

$$R_\phi = \left\{ \begin{matrix} (1,1,1,1) & , & (1,1,1,2) \\ (2,1,1,1) & , & (2,1,1,2) \\ (1,2,1,1) & , & (1,2,1,2) \\ (1,1,2,1) & , & (1\ 1,2,2) \\ (2,2,1,1) & , & (2,2,1,2) \end{matrix} \right\}.$$

It was also shown in [1] that S_1 ϕ-depends on S_3, S_2 ϕ-depends on S_3, but S_4 is ϕ-independent of all three stages S_1, S_2 and S_3.

Knowing that these dependencies are present, are there simpler ways to view the system so as to mask off the dependencies? Doing so would allow us to deal with these interactions on a lower (and possible simpler) level. For instance, on a lower level we might not need to obtain a whole set of conditional probabilities, but could instead obtain only absolute probabilities. It is possible that the absolute probabilities would inherently reflect the dependencies, without further system decomposition.

One example of such an alternative representation is shown in Figure 10. Here, S_1' is composed of stages S_1, S_2 and S_3, while S_2' corresponds to stage S_4. We can define a new state set Q_1'

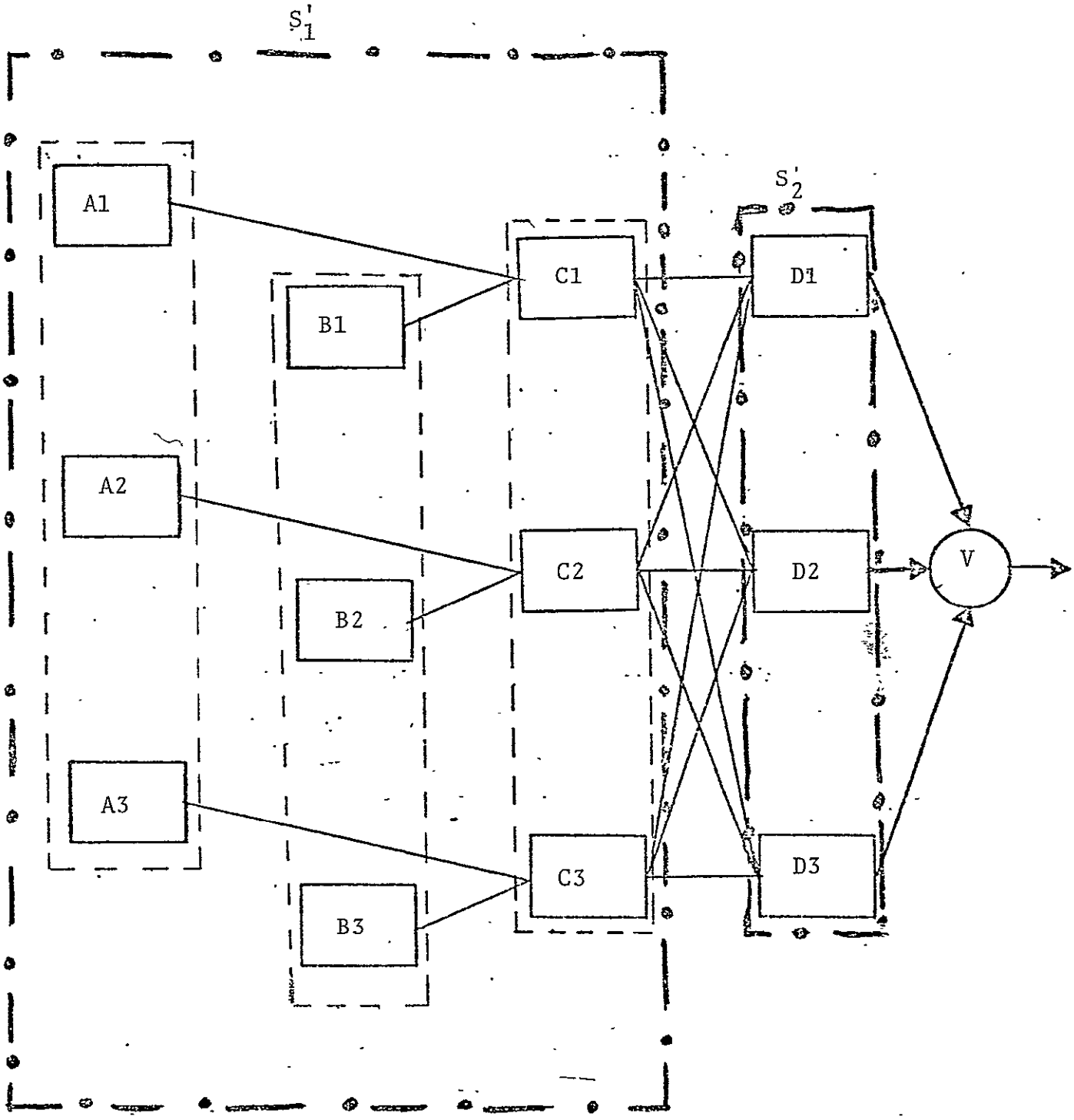


Figure 10

for S_1' which is the range of a 1-1 correspondence M, i.e., a function

$$M: Q_1 x Q_2 x Q_3 \rightarrow Q_1'$$

where, for example

(q_1, q_2, q_3)	$M(q_1, q_2, q_3)$
(1,1,1)	1
(1,1,2)	2
(1,2,1)	3
(1,2,2)	4
(2,1,1)	5
(2,1,2)	6
(2,2,1)	7
(2,2,2)	8
(3,1,1)	9
⋮	⋮
(5,5,5)	125 .

Note that this mapping preserves all the information used in our original analysis. We can now evaluate the new structure relation

$$\phi': Q_1' x Q_2' \rightarrow \{0,1\}.$$

to get

$$R_{\phi'} = \begin{Bmatrix} (1,1) & , & (1,2) \\ (2,1) & , & (2,2) \\ (3,1) & , & (3,2) \\ (5,1) & , & (5,2) \\ (7,1) & , & (7,2) \end{Bmatrix}.$$

The relation ϕ' is defined by

$$\phi'(M(q_1, q_2, q_3), q_4) = \phi(q_1, q_2, q_3, q_4)$$

where $q_i \in Q_i$, $i = 1,2,3,4$ and $Q_2' = Q_4$.

Calculation shows that

$$D_{\phi'}(1) = \{1,2,3,5,7\},$$
$$D_{\phi'}(2) = \{1,2\},$$
$$D_{\phi'}(1,2,1) = \{1,2,3,5,7\}, \text{ and}$$
$$D_{\phi'}(1,2,2) = \{1,2,3,5,7\}.$$

From this, S_1' is ϕ'-independent of S_2'.

A second way to analyze the system is shown in Figure 11. Here, there are two stages as above, S_1'' and S_2'', but we

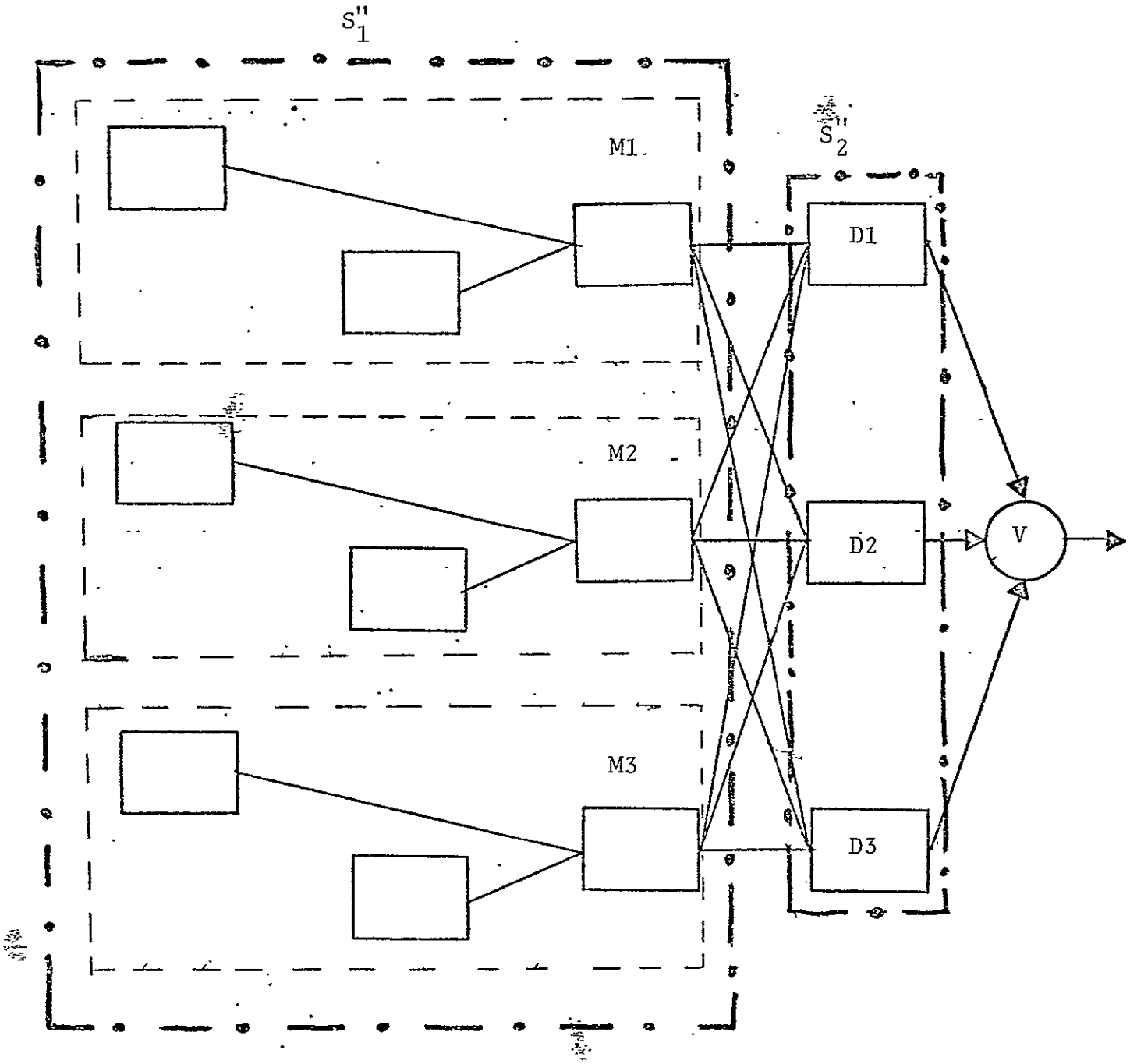


Figure 11

have not preserved the states of the originally defined stages. Instead, S_1'' is composed of the three modules denoted M1, M2, and M3, and S_1'' and S_2'' are regarded in the same fashion as in the original analysis. Hence the state sets Q_1'' of S_1'' and Q_2'' of S_2'' are

$$Q_1'' = Q_2'' = \{1,2,3,4,5\}.$$

Since the system is still regarded as TMR, the set of success states $R_{\phi''}$ are

$$R_{\phi''} = \begin{Bmatrix} (1,1) & , & (1,2) \\ (2,1) & , & (2,2) \end{Bmatrix}.$$

Again, S_1'' is ϕ-independent of S_2'', because $D_{\phi''}(1) = D_{\phi''}(2) = D_{\phi''}(1,2,1) = D_{\phi''}(1,2,2) = \{1,2\}$. Hence, this decomposition also yields independent stages in which internal dependencies are masked.

Study of ϕ-dependency during the past reporting period has shown that ϕ-dependence has the following properties:

i) ϕ-dependence is symmetric
ii) In general, ϕ-dependence is not reflexive
iii) In general, ϕ-dependence is not transitive.

It is important to note that ϕ-dependence suffers from the limitations imposed by structure-based analysis. As a result, during the past reporting period, and in conjunction with the development of a notion of a capability function, we have generalized the above notion of ϕ-dependence into what we call "R-dependence."

3.3.1.4 R-Dependence

The concept of R-dependence is an extension and generalization of the concepts involved in defining and determining ϕ-dependence. The notion of R-dependence will first be described

as a direct generalization of ϕ-dependence in terms of projections on coordinatized sets, and will then be given in an alternative mathematical formulation in terms of partitions of sets.

Let S be a system with subsystems $S_1,\ldots,S_n$. Here the notion of subsystem is extended beyond the components of the object system alone. Thus any part of the total system whose behavior influences the overall performance of the system may be considered a subsystem. For instance, weather or maintenance may be regarded as subsystems. Subsystems are represented by basic variables. Suppose that we sample the states of the subsystems at times $t_1,t_2,\ldots,t_k$ where $t_1 < t_2 < \ldots < t_k$ (due to this ordering we shall henceforth speak of times $1,2,\ldots,k$). Let Q_i^t be the set of possible operational states of subsystem S_i at time t.

Definition: Given the above conditions, a state trajectory for the system S is an nxk matrix

$$u = \begin{bmatrix} q_{11} & q_{12} \cdots & q_{1k} \\ \vdots & & \vdots \\ q_{n1} & \cdots & q_{nk} \end{bmatrix}$$

where for $1 \leq i \leq n$, $1 \leq t \leq k$, $q_{it} \in Q_i^t$. The $(i,t)^{th}$ entry of u is interpreted as the state of subsystem S_i at the t^{th} time sample. The i^{th} row of a state trajectory matrix corresponds to a state trajectory for subsystem S_j. The t^{th} column gives the state of the total system S (as represented by an n-tuple) at time t.

Let $U = \{[q_{it}] | q_{it} \in Q_i^t,\ 1 \leq i \leq n,\ 1 \leq t \leq k\}$ be the set of all state trajectories for S, and let $R \subseteq U$. The set R is the set relative to which dependency will be defined.

R may be selected by any specified criterion.

In the context of this research, the set R will generally be a "γ-induced" subset of U, that is, a set of the form $\{u|\gamma(u)=A_i,\ u\in U\}$ where A_i is a particular level of accomplishment (see Section 3.1.2.1), γ is the capability function (Section 3.3.2.2), and U, A_i, and γ are all relative to the same model hierarchy. The study of R-dependencies within such sets may yield system decompositions which ease the calculation of the probability of occurrence of the missions represented by elements of the set, i.e., the missions yielding a particular level of accomplishment. It is these calculations which are the object of this study, and which underlie the concept of "performability" or "expected performance." Thus, knowledge of existing dependencies within a system may aid us in calculating the performability of that system.

At this point, recall the development of functional (ϕ) dependence. If the system S is sampled only once, then U will be a set of n x 1 matrices, or, in other words, n-tuples. The set R would correspond to R_ϕ where the selection criterion is based on success states relative to the structure function ϕ.

If R is a set of k x n matrices and $u\in R$, define $\xi_{it}(u)=q_{it}$, i.e., $\xi_{it}(u)$ yields the $(i,t)^{th}$ element of (u). This is analogous to the projection operation on a vector.

Let $\xi_{it}(R) = \{\xi_{it}(u) | u \in R\}$. If one thinks of R as an array of matrices, then visually the projection $\xi_{it}(R)$ corresponds to selecting the $(i,t)^{th}$ element all along the third coordinate (see Figure 12).

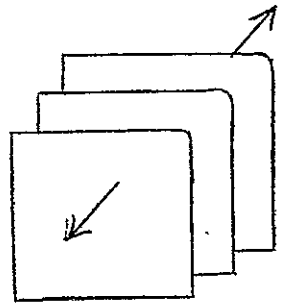


Figure 12

Now define $D(i,t) = \xi_{it}(R)$ as was done in describing ϕ-dependence. If $q \in D(j,v)$, let

$$R(j,v,q) = \{u \in R | q_{jv} = q\}.$$

The operator R(j,v,q) selects from R all those matrices whose $(j,v)^{th}$ element is q. Finally, define:

$$D(i,t\ ;\ \ell,v,q) = \xi_{it}(R(\ell,v,q)) \text{ for}$$

$$i,\ell \in \{1,\ldots,n\};\ t,v \in \{1,\ldots,k\},\ q \in Q_{\ell}^{v}.$$

Denote the fact that S_i is considered at time t by $S_i(t)$.

Definition: Let S be a system with subsystems $S_1,\ldots,S_n$ sampled at times $1,\ldots,k$. If U is the set of all state trajectory matrices of S and $R \subseteq U$, then we say $S_i(t)$ R-depends on $S_\ell(v)$ if, for some $q \in Q_{\ell}^{v}$,

$$D(i,t) \neq D(i,t;\ell,v,q).$$

Consider the following example. Let

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 2 & 2 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

$$B = \begin{bmatrix} 2 & 2 & 2 \\ 1 & 1 & 1 \\ 2 & 2 & 2 \end{bmatrix} \qquad D = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 1 & 1 \end{bmatrix}, \text{ and let}$$

$R = \{A,B,C,D\}$. R might be the accomplishment set correspond-

ing to an accomplishment A of a 3 stage system sampled at 3 times. Three different relationships will be examined. The list is meant to be illuminating, but not exhaustive.

i) Consider $S_1(3)$ and $S_1(1)$.

$D(1,3) = \{1,2\}$ but $D(1,3;1,1,2) = \{2\}$ so $S_1(3)$ R-depends on $S_1(1)$. This dependency is a temporal dependency between the states of a single subsystem.

ii) Consider now $S_1(2)$ and $S_2(3)$.

$D(1,2) = \{1,2\}$,
$D(1,2;2,3,2) = \{1\}$, and
$D(1,2;2,3,1) = \{2\}$.

Hence $S_1(2)$ R-depends on $S_2(3)$. In fact, knowing that $S_2(3)$ is in state 2 tells that $S_1(2)$ is in state 1, and knowing $S_2(3)$ is in state 1 tells that $S_1(2)$ is in state 2. Thus this dependence is also an example of the first mode of dependence discussed above. This is not true in (i) since $D(1,3;1,1,1) = \{1,2\}$ - so it is not known, given $S_1(1)$ is in state 1, which state $S_1(3)$ is in.

iii) Consider $S_2(2)$ and $S_2(3)$.

$D(2,3) = \{1,2\}$,
$D(2,3;2,2,1) = \{1,2\}$ and
$D(2,3;2,2,2) = \{1,2\}$,

so $S_2(3)$ does <u>not</u> R-depend on $S_2(2)$. We say <u>$S_2(3)$ is R-independent of $S_2(2)$</u>.

The above definitions for R-dependency follow the exposition given in the functional dependency case. However, R-dependency can be characterized directly in terms of certain partitions associated with time and state coordinates. Suppose we are given

S, R and a delimitation of the subsystems and times of interest. For instance, with S and R as above, we might wish to investigate the relationship between $S_2(3)$ and $S_1(2)$. If $u, v \in R$, let Ξ_{it} be the equivalence relation defined by

$$u \; \Xi_{it} \; v \text{ if } \xi_{it}(u) = \xi_{it}(v).$$

Hence, if $\xi_{it}(R)$ has c different elements, Ξ_{it} will partition R into c different classes. Denote this partition by R/Ξ_{it}. Clearly, there are $n \cdot k$ such partitions of R, one for each (i,t) pair.

Recall that in the definition of R-dependency, projections were repeatedly made on the $(i,t)^{th}$ element while holding the $(j,v)^{th}$ element fixed (at its various values) in order to determine the relationship between $S_i(t)$ and $S_j(v)$. If restricting the set over which a projection on the $(i,t)^{th}$ coordinate was taken restricted the possible values of elements in that projection, then dependency was said to be present. These multiple projections are a way of partitioning the set R in various ways. Realizing this, we can characterize R-dependence as follows:

Theorem: If R/Ξ_{it} and R/Ξ_{jv} are partitions of R, then $S_i(t)$ R-depends on $S_j(v)$ if and only if there exists a block $B \in R/\Xi_{it}$ and a block $B' \in R/\Xi_{jv}$ such that $B \cap B' = \phi$.

In other words, if one partitions R on the two different coordinates, then no dependencies are present if and only if each block in the first partition has a non-trivial intersection with each block in the second partition.

As an example, return to R and the matrices A,B,C,D of the R-dependence example above.

Consider:

$R/\equiv_{11} = \{\{A,C,D\},\{B\}\}$ and

$R/\equiv_{13} = \{\{A,D\},\{B,C\}\}$.

Then $\{A,D\} \cap \{B\} = \phi$, so $S_1(1)$ R-depends on $S_1(3)$.

Similarly $R/\equiv_{23} = \{\{A,D\},\{B,C\}\}$ and

$R/\equiv_{12} = \{\{A,D\},\{B,C\}\}$

but $\{A,D\} \cap \{B,C\} = \phi$ so $S_2(3)$ R-depends on $S_1(2)$.

Thirdly, we see

$R/\equiv_{23} = \{\{A,D\},\{B,C\}\}$

$R/\equiv_{22} = \{\{A,B\},\{C,D\}\}$

so

$\{A,D\} \cap \{A,B\} = \{A\}$,
$\{A,D\} \cap \{C,D\} = \{D\}$,
$\{B,C\} \cap \{A,B\} = \{B\}$, and
$\{B,C\} \cap \{C,D\} = \{C\}$.

As this exhausts all the possibilities, we see that $S_2(3)$ is R-independent of $S_2(2)$.

3.3.2 Capability

3.3.2.1 Definition and Role

As discussed in Section 3.3, we are focusing our attention on a "performability" view of system effectiveness where a measure of performability can be decomposed into measures of availability, dependability, and capability. In terms of the model hierarchy, the availability and dependability measures quantify the behavior of the bottom level model. A capability measure quantifies the behavior of the top model as a function of values assumed by the "basic" variables of the bottom and intermediate models. (Recall that a variable is "basic" if it cannot be

expressed as a function of lower level variables.) Thus the capability aspect of performability combines all aspects of the model hierarchy.

The capability measures we have studied during the reporting period are essentially three-fold extensions of the structure functions discussed in Section 3.3.1.1. We call these measures "capability functions." In the notion of a capability function, the concept of a structure function has been extended in the following three ways:

i) The subsystems (components) of a system may be characterized by multiply valued state sets,

ii) The accomplishment set $\mathcal{A}$ may contain more than two elements, and

iii) The capability function is defined over a set of state trajectories.

These extensions, together with a formal definition of a capability function, are described in the following subsection.

3.3.2.2 The Capability Function

Viewing a "capability function" as a three-fold extension of the notion of a structure function, the first extension allows the state sets of the subsystems to have more than two elements. This permits characterization of degraded performance in the system's subsystems, i.e., removes the requirement that a subsystem be considered either "all on" or "all off." For example, this allows us to more accurately describe the state of a component which is constructed on the triple modular redundancy (TMR) principle. The second extension is to allow capability functions to be multivalued. Thus the range of a capability

function will generally be taken to be an "accomplishment set" or a set of mission characterizing quantities which serve to quantify some aspect of a system's effectiveness in carrying out a prescribed mission. This allows us to go beyond a simple "success-failure" characterization of the system's performance of its mission. The third extension is to take the argument of a capability function to be a state trajectory rather than single state vectors as is the case with structure functions. This permits one to investigate the behavior of the system over time. A capability function for a system which is sampled only at one time, for which each subsystem has only two states, and for which the accomplishment set has two elements, reduces directly to a structure function.

Viewed in the context of the model hierarchy, a capability function is a formal expression of how the state trajectories of the base model (bottom model plus higher level basic variables) relate to mission outcomes (and thereby mission accomplishments) at the mission level.

Suppose we have a system S which is decomposed into n subsystems (basic variables) $S_1,\ldots,S_n$. Let the system be sampled at times $t_1,\ldots,t_k$ where $t_i < t_{i+1}$, $1 \leq i \leq k-1$. In Section 3.3.1.4 we define a state trajectory for the system S to be the $n \times k$ matrix u where u_{ij} corresponds to the state of subsystem S_i when sampled at time t_j $(1 \leq i \leq n,\ 1 \leq j \leq k)$. Let $\mathscr{A}$ be an accomplishment set for the set of mission performed by the system S.

Definition: Let U be the set of all state trajectories for some system S. Let $\mathscr{A}$ be an accomplishment set over S.

Then a capability function is a function γ such that

$$\gamma: \; U \rightarrow \mathscr{A} .$$

The relationship between the model hierarchy and capability functions will be demonstrated in the following section. In general, one will attempt to define a capability function by formulating the transformations between levels of the hierarchy, and then composing the transformations in order to tie the hierarchy together. One area of concentration during the next reporting period is concerned with the study of these transformations, introduction of basic variables and the subsequent capability functions which are derived.

3.3.2.3 The Capability Function and the Model Hierarchy

As was noted in the previous section, the model hierarchy provides a framework which supports the capability function γ. By using the hierarchy of models to move from level to level we can reduce the problem of formulating the capability function to the problem of formulating the values of composite variables at level i in terms of model variables (both basic and composite) at level i+1. If one thinks of moving up through the hierarchy, the basic step is "jump to the next level and incorporate basic variables." A precise description of this process is one of the goals established for the next reporting period. The remainder of this section is devoted to examples of this procedure.

In Section 3.1.3 a hierarchy was elaborated down to the Intermediate 2 level. This hierarchy forms the basis for the following discussion. Since this hierarchy does not have a bottom model associated with it, we cannot show a capability function per se. However, the three levels which have been elaborated can be used to show the manner in which a trajectory

might be mapped up through such a modeling scheme. Before proceeding, the reader may wish to review the development in Section 3.1.3. Figure 13 shows the various matrix variables discussed, together with an interpretation of their meaning.

Consider the Intermediate 2 trajectory

$$\mathbf{x}^1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ ¢ & ¢ & ¢ & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

where the symbol '¢' denotes an element whose value is of no concern. This trajectory describes a mission (at the Intermediate 2 level) in which the autoland computations are in a 'failed' state before and during landing. However, no other complications are encountered. Thus in moving to Intermediate 1, the active control and autoland will be represented by

$$\mathbf{y}_c^1 = \begin{bmatrix} 0 & 0 & 0 \\ ¢ & 1 & 1 \end{bmatrix}$$

indicating that active control is good but autoland has failed sometime during cruise and landing. We now incorporate the weather variable, which is represented by (¢, α, ¢) to get

$$\mathbf{y}^1 = \begin{bmatrix} 0 & 0 & 0 \\ ¢ & 1 & 1 \\ ¢ & \alpha & ¢ \end{bmatrix}, \quad \alpha \in \{0,1\}.$$

This yields two qualitatively different matrices, depending on the value of α. Recall that $\alpha = 0$ indicates good weather at the beginning of landing while $\alpha = 1$ indicates weather which calls for a Category III type landing.

Top Level

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}$$

z_1 Safety

z_2 Diversion

z_3 Fuel consumption

Intermediate 1

$$\mathbf{y} = \begin{bmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \\ \hline y_{31} & y_{32} & y_{33} \end{bmatrix}$$

(Phase 1, Phase 2, Phase 3)

y_{11} y_{12} y_{13} Active Control } Composite variables

y_{21} y_{22} y_{23} Autoland } Composite variables

y_{31} y_{32} y_{33} Weather } Basic variable

Intermediate 2

$$\mathbf{x} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} \\ x_{31} & x_{32} & x_{33} & x_{34} & x_{35} \\ x_{41} & x_{42} & x_{43} & x_{44} & x_{45} \end{bmatrix}$$

(Phase 1, Phase 2, Phase 3)

x_{11} x_{12} x_{13} x_{14} x_{15} Active control computations

x_{21} x_{22} x_{23} x_{24} x_{25} Fuel computations

x_{31} x_{32} x_{33} x_{34} x_{35} Autoland computations

x_{41} x_{42} x_{43} x_{44} x_{45} Internal computations

x_{42} Beginning

x_{43} Middle

x_{44} End

Figure 13

Description of Matrix Variables from Section 3.1.3

First consider the matrix

$$\mathbf{y}^1 = \begin{bmatrix} 0 & 0 & 0 \\ \not\in & 1 & 1 \\ \not\in & 0 & \not\in \end{bmatrix}.$$

In this case, the weather was good, there was no diversion, and fuel consumption was low (this comes from active control = (0,0,0)), so

$$\mathbf{z}^1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$ which results in an A_1 level of accomplishment. Contrarily, the matrix

$$\mathbf{y}^{1'} = \begin{bmatrix} 0 & 0 & 0 \\ \not\in & 1 & 1 \\ \not\in & 1 & \not\in \end{bmatrix}$$

indicates a faulty autoland system plus bad weather, so a diversion is required. In this case,

$$\mathbf{z}^{1'} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix},$$

indicating the third level of accomplishment A_3. Note that the mapping from Intermediate 1 to the mission level depends on the value of the weather variable at the end of the cruise phase.

A second example begins with the matrix

$$\mathbf{x}^2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ \not\in & \not\in & \not\in & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Both active control computations and internal computations remain error-free throughout the mission. The fuel regulation computations were accomplished only in the final part of the cruise

phase, and autoland computations fail during landing. Thus, this matrix maps into the matrix

$$\mathbf{y}_c^2 = \begin{bmatrix} 1 & 1 & 1 \\ \not{\alpha} & 0 & 1 \end{bmatrix}$$

on the Intermediate 1 level. The active control trajectory indicates fuel regulation difficulties, while the autoland trajectory indicates a failure after the landing phase has begun.

As above, we incorporate the weather variable to get

$$\mathbf{y}^2 = \begin{bmatrix} 1 & 1 & 1 \\ \not{\alpha} & 0 & 1 \\ \not{\alpha} & \alpha & \not{\alpha} \end{bmatrix}.$$

For $\alpha = 0$, the failure of the autoland system does not affect the mission quality. In terms of mission variables we see that $z_1 = 0$ (no fatalities), $z_2 = 0$ (no diversion is necessary) and $z_3 = 1$ (there is high fuel consumption due to failure of fuel regulation). Hence

$$\mathbf{z}^2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad \text{so an}$$

accomplishment level of A_2 is achieved. For $\alpha = 1$, we see a condition where the autoland system fails while landing the aircraft. By our assumptions a (fatal) crash ensues. Knowing this,

$$\mathbf{z}^{2'} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

which indicates the fifth level of accomplishment.

The above examples show the way in which a capability function may pass from level to level. It is clear that a capability function is a function of both bottom model state trajectories and the trajectories of other basic variables

(such as weather) which are inserted at higher levels. One thrust of our ongoing research is aimed at developing a more concise representation of capability functions, together with the investigation of the properties of and relationships between R-dependence, capability, and performability.

4. REFERENCES

[1] "Models and Techniques for Evaluating the Effectiveness of Aircraft Computing Systems," Semi-Annual Status Report No. 1, NASA Grant NSG 1306, November, 1976.

[2] "Models and Techniques for Evaluating the Effectiveness of Aircraft Computing Systems," Proposal for extension of NASA Grant NSG 1306, submitted to NASA Langley Research Center and subsequently approved, May 1977.

[3] J. H. Wensley, M. W. Green, K. N. Levitt, and R. E. Shostak, "The Design, Analysis, and Verification of the SIFT Fault-Tolerant System," 1976 International Conference on Software Engineering, San Francisco, California, pp. 458-469, 1976.

[4] T. B. Smith, A. L. Hopkins, et al., Interim Report to NASA Langley Research Center, Contract No. NAS1-13782, September 1976.

[5] J. D. Esary and H. Ziehms, "Reliability of Phased Missions," Reliability and Fault Tree Analysis, SIAM, Philadelphia, Pennsylvania, pp. 213-236, 1975.

[6] H. S. Winokur, Jr., and L. J. Goldstein, "Analysis of Mission-Oriented Systems," IEEE Trans. on Reliability, Vol. R-18, No. 4, pp. 144-148, November 1969.

[7] G. R. Burdick, J. B. Fussell, D. M. Rasmuson, and J. R. Wilson, "Phased Mission Analysis: A Review of New Developments and an Application," IEEE Trans. on Reliability, Vol. R-26, No. 1, pp. 43-49, April 1977.

[8] J. L. Bricker, "A Unified Method for Analyzing Mission Reliability for Fault Tolerant Computer Systems," IEEE Trans. on Reliability, Vol. R-22, No. 2, pp. 72-77, June 1973.

[9] R. S. Ratner, E. B. Shapiro, H. M. Zeidler, S. E. Wahlstrom, C. B. Clark and J. Goldberg, "Design of a Fault Tolerant Airborne Digital Computer," Vol. II, Stanford Research Institute, Final Report, NASA Contract NAS1-10920, October 1973.

[10] B. C. Kuo, Automatic Control Systems, Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1975.

[11] Y.-W. Ng and A. Avižienis, "A Model for Transient and Permanent Fault Recovery in Closed Fault-Tolerant Systems," Proc. 1976 International Symp. on Fault-Tolerant Computing, Pittsburgh, Pennsylvania, pp. 182-188, June 1976.

[12] C. J. Masreliez and B. E. Bjurman, "Fault Tolerant System Reliability Modeling/Analysis," presented at the AIAA Guidance and Control Conference, San Diego, California, August 1976.

[13] B. E. Bjurman, G. M. Jenkins, C. J. Masreliez, K. L. McClellan and J. E. Templeman, Airborne Advanced Reconfigurable Computer System (ARCS), Final Report, NASA Contract NAS1-13654, Boeing Commercial Airplane Company, Seattle, Washington, August 1976.

[14] M. D. Beaudry, "Performance Related Reliability Measures for Computing Systems," Proc. 1977 International Conference on Fault-Tolerant Computing, Los Angeles, California, pp. 16-21, June 1977.

[15] R. Troy, "Dynamic Reconfiguration: An Algorithm and its Efficiency Evaluation," Proc. 1977 International Conference on Fault-Tolerant Computing, Los Angeles, California, pp. 44-49, June 1977.

[16] F. P. Mathur, "On Reliability Modeling and Analysis of Ultra-Reliable Fault-Tolerant Digital Systems," IEEE Trans. on Computers, Vol. C-20, No. 11, pp. 1376-1382, November 1971.

[17] W. G. Bouricius, W. C. Carter, D. C. Jessup, P. R. Schneider, and A. B. Wadia, "Reliability Modeling for Fault-Tolerant Computers," IEEE Trans. on Computers, Vol. C-20, No. 11, pp. 1306-1311, November 1971.

[18] "Reliability Model Derivation of a Fault-Tolerant, Dual, Spare-Switching, Digital Computer System," Equipment Development Laboratory, Raytheon Co., Final Report, NASA contract NAS1-12668, 25 March 1974.

[19] D. R. Cox, "A Use of Complex Probabilities in the Theory of Stochastic Processes," Proc. Cambridge Philosophical Society, Vol. 51, pp. 313-319, 1955.

[20] D. R. Cox and H. D. Miller, The Theory of Stochastic Processes, John Wiley and Sons, Inc., New York, pp. 257-259, 1965.

[21] C. Singh, "Reliability Modelling and Evaluation in Electric Power Systems," University of Saskatchewan, Saskatoon, Canada, Ph.D. Thesis, 1972.

[22] J.-C. Laprie, "Reliability and Availability of Repairable Structures," Digest 1975 International Symposium on Fault-Tolerant Computers, Paris, pp. 87-92, June 1975.

[23] C. Singh, R. Billinton and S. Y. Lee, "The Method of Stages for Non-Markov Models," IEEE Trans. on Reliability, Vol. R-26, No. 2, pp. 135-137, June 1977.

[24] Weapon System Effectiveness Industry Advisory Committee, Final Report AFSC-TR-65-6 (AD-467 816), January 1965.

[25] R. E. Barlow and F. Proschan, Statistical Theory of Reliability and Life Testing, Holt, Rhinehart, and Winston, Inc., New York, 1975.