AD-A116 566 HUGHES AIRCRAFT CO FULLERTON CA GROUND SYSTEMS GROUP F/G 9/2
COMBINED HW/SW RELIABILITY MODELS.(U)
APR 82 L E JAMES, J E ANGUS, J B BOWEN F30602-80-C-0085
UNCLASSIFIED HAC-FR-81-16-351A RADC-TR-82-68 NL

1 of 2
AD A
116566

(12)

RADC-TR-82-68
Final Technical Report
April 1982

COMBINED HW/SW RELIABILITY MODELS

Hughes Aircraft Company

L.E. James, J.E. Angus, J.B. Bowen and J. McDaniel

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-82-68 has been reviewed and is approved for publication.

APPROVED:

JEROME KLION
Project Engineer

APPROVED: David C. Luke

DAVID C. LUKE, Colonel, USAF
Chief, Reliability & Compatibility Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (RBET) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER RADC-TR-82-68	2. GOVT ACCESSION NO. AD-A116 766	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) COMBINED HW/SW RELIABILITY MODELS		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report Feb 80 - Feb 81
		6. PERFORMING ORG. REPORT NUMBER FR 81-16-351A
7. AUTHOR(s) L. E. James J. B. Bowen J. E. Angus J. McDaniel		8. CONTRACT OR GRANT NUMBER(s) F30602-80-C-0085
9. PERFORMING ORGANIZATION NAME AND ADDRESS Hughes Aircraft Company, Ground Systems Group P O Box 3310 Fullerton CA 92634		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 23380241
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (RBET) Griffiss AFB NY 13441		12. REPORT DATE April 1982
		13. NUMBER OF PAGES 158
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

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

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Jerome Klion (RBET)

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

Software Reliability
Hardware Reliability
Combined HW/SW Reliability
Purely Discontinuous Markov Processes
C^3 Systems
Redundancy
Maintainability

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

A general methodology is developed for combining hardware and software reliability. Based on this general methodology, a baseline combined HW/SW reliability was developed incorporating and unifying the SW reliability theory of Jelinski-Moranda, Goel-Okumoto with traditional HW reliability theory. The baseline model is computerized and includes various HW/SW failure and repair characteristics, allowance for imperfect SW debugs and modes of HW/SW interaction. Finally a HW/SW tradeoff procedure is developed using a combined HW/SW availability measure, Examples are provided to illustrate the general theory and tradeoff procedure.

TABLE OF CONTENTS

Section Zero - EXECUTIVE SUMMARY 0-1

Section One - SUMMARY OF STUDY RESULTS 1-1

Section Two - RESULTS OF LITERATURE SEARCH 2-1

Section Three - THEORETICAL FOUNDATIONS OF COMBINED HW-SW RELIABILITY MODELS 3-1

3.1 Introduction 3-1
3.2 The General Combined Hardware/Software Reliability Model 3-1
3.3 Reliability Measures 3-5

Section Four - APPLICATION OF METHODOLOGY TO SIMPLE RELIABILITY CONSTRUCTS 4-1

4.1 Analysis of Series Constructs 4-2

4.1.1 Two-State Series Case 4-3
4.1.2 Three State Series Case 4-6

4.2 Analysis of Constructs with Redundant Hardware 4-10

4.2.1 Transition Diagrams and Infinitesimal Matrices 4-10
4.2.2 Examples 4-13
4.2.3 Modifications 4-16

4.3 SW Redundancy 4-18
4.4 The Use of Other SW Models 4-19
4.5 Approximations to the HW/SW Model 4-20

4.5.1 Series HW Constructs 4-20
4.5.2 Redundant HW Constructs 4-20

Section Five - EXTENSION TO COMPLEX RELIABILITY CONSTRUCTS: C^3 SYSTEMS 5-1

5.1 Introduction 5-1
5.2 Development of the System Reliability Model 5-3

5.2.1 Mission Tasks and Operating Scenarios 5-3
5.2.2 System Reliability Model 5-5

5.3 Failure Definition 5-10

5.3.1 Derivation of Transition Rates for HW Failure 5-10
5.3.2 Derivation of Transition Rates for SW Failure 5-11

5.4 Quantifiable Maintainability Concepts 5-12

TABLE OF CONTENTS (Continued)

5.4.1 Derivation of Transition Rates for HW Repair 5-12
5.4.2 Derivation of Transition Rates for SW Repair 5-13

5.4.2.1 Introduction 5-13
5.4.2.2 General SW Maintainability Considerations 5-13
5.4.2.3 Consideration of SW Maintainability Techniques in Modeling System Restoration.................. 5-14
5.4.2.4 Selection of Quantifiable Attributes 5-16
5.4.2.5 System Compatible SW Maintainability Metrics 5-20

Section Six - RELIABILITY TRADEOFF METHODOLOGY 6-1

6.1 General Approach 6-1
6.2 HW Complexity 6-2
6.3 SW Complexity 6-2
6.4 Tradeoff Procedure 6-2
6.5 Examples 6-3

6.5.1 Series Constructs.................................... 6-3
6.5.2 Redundant HW Constructs: Five State 6-4

Section 7.0 - CONCLUSIONS AND RECOMMENDATIONS 7-1

Section 8.0 - BIBLIOGRAPHY AND REFERENCES 8-1

APPENDIX A - IMPORTANT NOTATIONS AND DEFINITIONS A-1

APPENDIX B - PURELY DISCONTINUOUS MARKOV PROCESSES B-1

APPENDIX C - A MARKOVIAN SW PROCESS C-1

APPENDIX D - DEVELOPMENT OF A COMBINED HW/SW RELIABILITY MODEL USING A MARKOVIAN SW PROCESS D-1

APPENDIX E - NUMERICAL/COMPUTATIONAL ASPECTS OF THE COMBINED HW/SW RELIABILITY MODEL............... E-1

APPENDIX F - COMPUTER PROGRAM FOR CALCULATING STATE OCCUPANCY PROBABILITIES ....:.. F-1

LIST OF FIGURES

Figure		Page
0.0-1	Typical Path of X(t)	0-3
0.0-2	Same Paths of Y(t) and X(t)	0-3
0.0-3	System Having Both HW and SW Failure	0-4
0.0-4	A(t) Versus Time	0-5
0.0-5	Reliability Block Diagram for a C^3 System	0-7
1.0-1	General Arrangement of Sections Within the Technical Report	1-3
3.2-1	Example of a System Having Both HW and SW Failure States	3-3
3.2-2	Sample Paths of Y(t), $\geq$t 0, X(t), t$\geq$0	3-4
4.1-1	Examples of Simple Reliability Constructs	4-2
4.1-2	A(t) Versus Time: ϕ = 1, c = 0.95	4-9
4.1-3	A(t) Versus Time: ϕ = 0.001, 0.01, c = 0.1,2	4-11
4.2-1	Transition Diagram for a Configuration with Redundant HW	4-12
4.5-1	Availability Approximations for Series Reliability Construct	4-21
4.5-2	Error Estimate Using $A(t) = A_{HW}(t) \cdot A_{SW}(t)$ Versus Number of HW Units	4-22
4.5-3	Availability Approximation for a Two-Standby Redundant Construct	4-23
5.1-1	General Model for Complex Reliability Constructs	5-2
5.2-1	Simplified Overview of Air Surveillance System	5-4
5.2-2	Air Surveillance Mission Sub-task Breakdown Into Computer Program Functions	5-5
5.2-3	Reliability Block Diagram for a C^3 System	5-6
5.4.2-1	Bohem's Maintainability SW Tree	5-17

LIST OF FIGURES (Continued)

Figure Page

5.4.2-2 McCall-Matsumoto SW Maintainability Tree 5-17

5.4.2-3 Hughes-Recommended SW Maintainability Tree 5-18

6.4-1 Isometric Curve of Constant A* as a Design Guide 6-3

6.5-1 Isometric Curve of A*=0.97 for the Series Example 6-4

6.5-2 Isometric Curve of A*=0.99 for the Redundant HW Example 6-5

B.1-1 Typical Path of a Purely Discontinuous Stationary Markov Process B-3

F-1 Simplified Flow Chart for Computer Program F-4

F-2 *Detailed Flow Diagram* F-5

LIST OF TABLES

Table		Page
0.0-1	Partitioning of an Air Surveillance System SW	0-8
2.1	Literature Searches	2-3
4.2-1	State Occupancy Probabilities for the Five State Example	4-15
4.2-2	A(t) for the Five-State Example	4-16
4.2-3	State Occupancy Probabilities for the Seven State Example ..	4-17
4.2-4	A(t) for the Seven State Example	4-18
5.2-1	Example Partitioning of an Air Surveillance System SW Based on Percentage Utilization	5-7
5.2-2	System Equations for Computing Combined HW/SW Reliability	5-9
5.4 2-1	Impact of Software Maintainability Techniques on System Restoration	5-15
5.4.2-2	Tailoring Scheme for Maintainability Factors.................	5-19
5.4.2-3	Suggested Maintainability Factors	5-19
5.4.2-4	Typical C^3 System Recovery Time after Software Failure	5-22
5.4.2-5	Typical C^3 System SW Correction Time for Patch Method	5-22
5.4.2-6	Typical C^3 System SW Correction Time for Recompile/Reassembly Method	5-23
5.4.2-7	Description of SW Maintainability Metrics....................	5-24

Section 0.0

EXECUTIVE SUMMARY

The quantity and importance of embedded software (SW) in modern weapon systems has increased dramatically over the last decade and, with ever increasing complexity of this SW, it has become necessary to consider the impact of the SW faults and maintenance actions on system effectiveness. Whereas in systems of several generations past the SW contribution to system unavailability could reasonably be neglected, the effects of SW faults on modern weapon systems are known to be substantial. While the state-of-the-art in hardware (HW) reliability modeling is now well founded, and with SW reliability modeling presently approaching a state where SW reliability requirements can be specified and designed toward, there is no unified technique for modeling the reliability/maintainability scenario of a system exhibiting both HW and SW faults and repairs. It is the purpose of this study to develop such a unified technique so that combined HW/SW reliability measures may be specified and analyzed to lead to a reliability design which is adequate to meet mission requirements. This unified technique will consist of developing a combined HW/SW reliability model, i.e., a mathematical model which probabilistically describes, at any point in time, the state (in terms of reliability) of a system exhibiting both HW and SW faults and repairs. What follows represents a rudimentary description of the concepts and theory behind this model.

There are two basic components in the combined HW/SW reliability model. The most fundamental of these is the SW fault correction process which represents the number of faults remaining in the SW (and hence defines the SW failure rate as a function of the proportion of faults remaining) at any point in time. Letting $X(t)$ represent the number of SW faults at time t, we assume that $X(t)$ changes as faults occur and are corrected (the implicit assumption made is that $X(t)$ takes the form of a time-homogeneous Markov process). Figure 0.0-1 shows how $X(t)$, and hence SW failure rate, changes as fault corrections take place at the random points in time t_1, t_2, ... During each interval between fault removals, SW failure rate remains constant, taking a value proportional to the number of faults remaining in the SW. Notice that in Figure 0.0-1, any number of faults can be corrected (or even inadvertently introduced) over any interval of time. The exact nature and characteristics of X(t) are embedded in the character of the particular software reliability model chosen for X(t). This is described in detail in the body of the repcrt.

The second component of the combined HW/SW reliability model is the process describing the operating condition of the system, the "system state" (in terms of which of its components are operating or not operating due to system malfunctions and restorations caused by both software and hardware) at any point in time within a given interval defined by X(t).

We represent the possible system states by the set (0, 1, 2, ..., J) where we arbitrarily assign the full-up state to "O" (implying none of the system components are in an inoperative state). The remaining states namely 1, 2, ... J can be used to represent any combination of HW/SW operational degraded or inoperative states. We represent the state of the system at time t by Y(t), and it turns out that (conditioned on t being in a time interval between successive changes in the SW failure rate) Y(t) evolves according to a time-homogeneous Markov process. Figure 0.0-2 shows the relationship between the two components. Referring to Figure 0.0-2, beginning at time 0, Y(t) evolves according to a Markov process (determined by the fixed values of the failure and repair rates and the initial value of the SW failure rate) until time t_1, when the SW failure rate jumps to a new value. From time t_1 until the next jump in SW failure rate, Y(t) evolves according to a different Markov process, different because the SW failure rate changed at time t_1. The process Y(t) continues in this fashion.

The concept of availability of a combined HW/SW system may be defined exactly as for a system possessing only HW. Namely, the availability of the system at time t is the probability that the system is capable of performing all necessary mission tasks at time t. During periods of time when the SW failure rate is constant, portions of the system (including SW) may incur malfunctions and repairs or restorations and hence, the system state Y(t) may transition from one state to another. Figure 0.0-3 provides a simple three-state example of a transition diagram when there are j faults remaining in the SW. Naturally, when the SW failure rate is high, the SW will fail more frequently. On the other hand, when enough SW faults have been removed to effect a substantial reduction in the SW failure rate, the system failures due to SW become less frequent and can ultimately be eliminated if the SW can be totally debugged. This changing nature of SW failure rate is responsible for an "availability undershoot" which is illustrated in Figure 0.0-4. When a system possesses no SW, and providing that the system state is full-up at time 0, then its availability curve begins at the value 1 at time 0 and decreases (as t increases) approaching the steady-state value. For this reason, steady state availability is often specified as a requirement. For example, for a system with only one HW unit (and no SW) the "steady state availability" is MTBF/(MTBF+MTTR), a well-known formula to reliability engineers. For this system, the time dependent availability always exceeds the "steady state" value, MTBF / (MTBF+MTTR), and so specifying

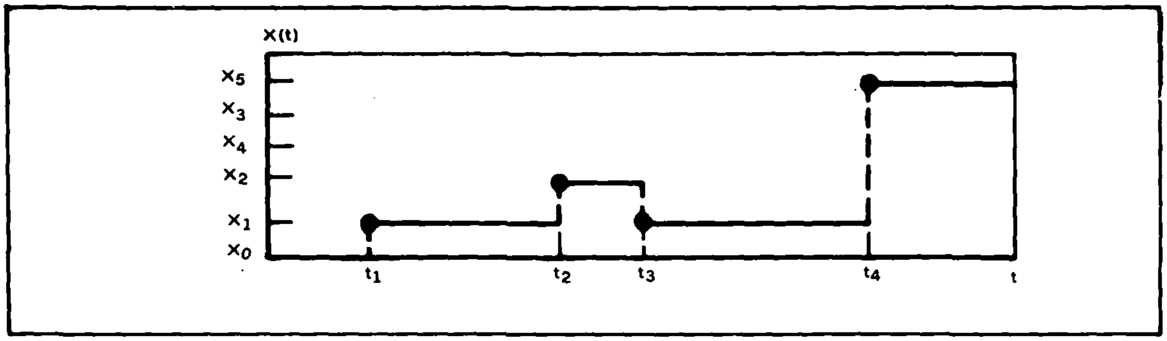


Figure 0.0-1. Typical path of X(t).

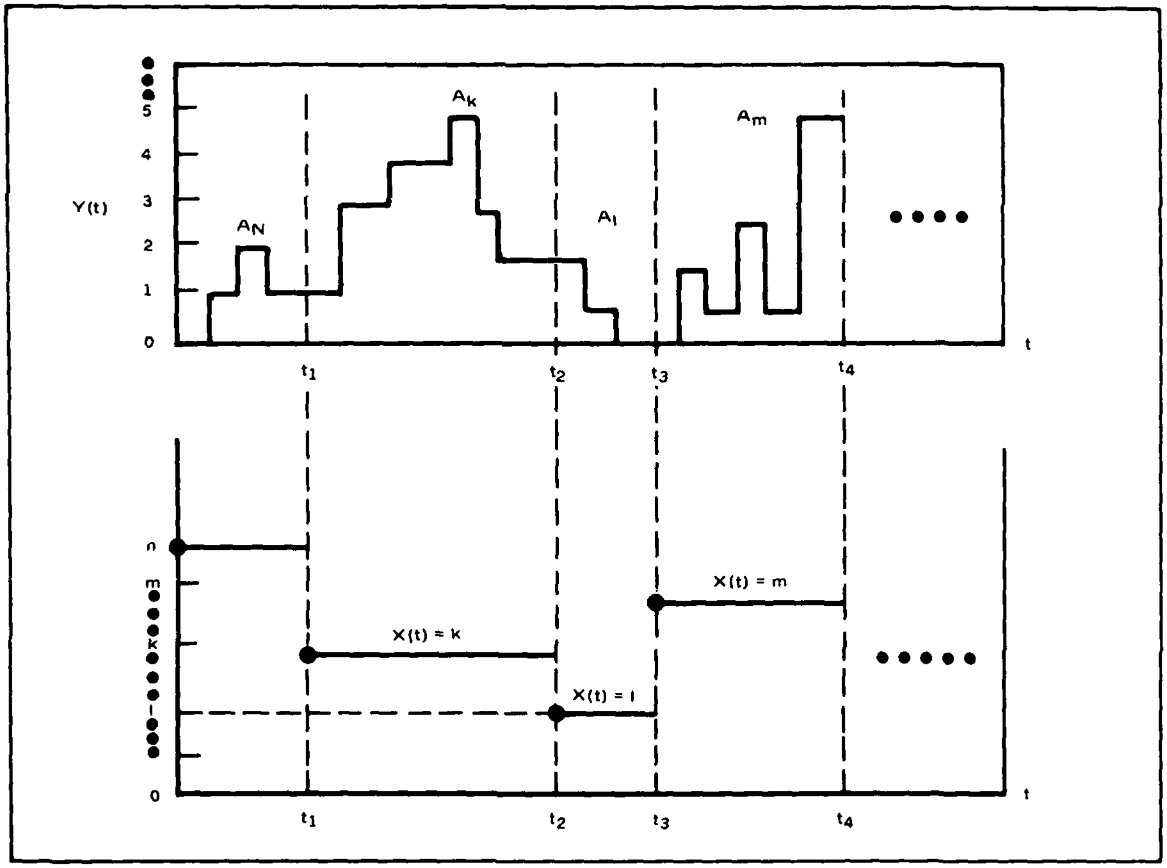
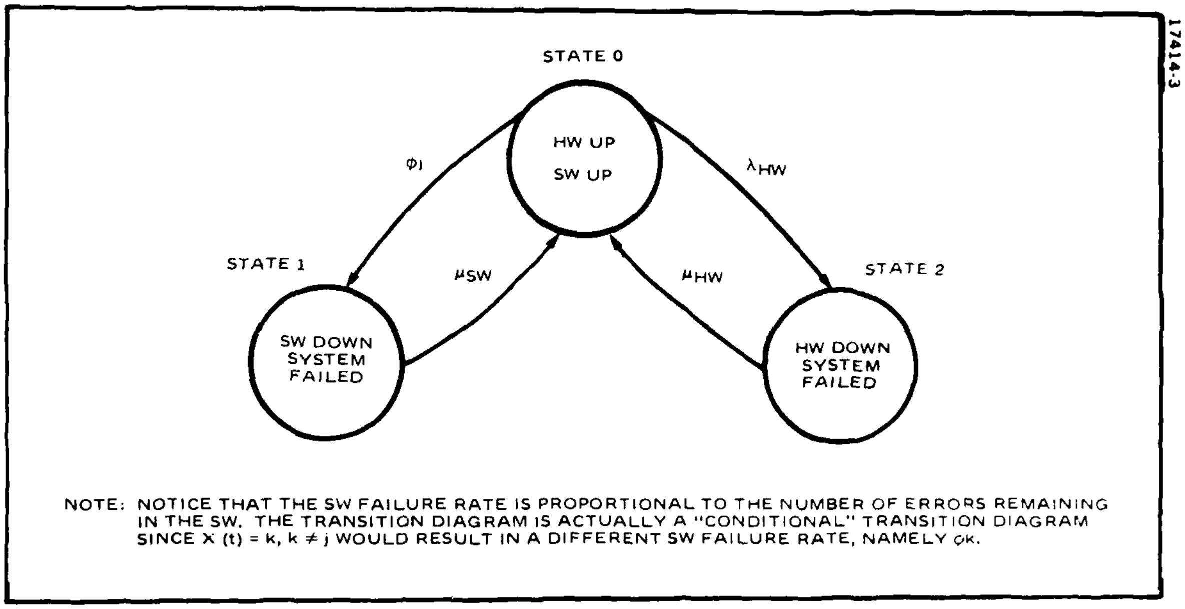


Figure 0.0-2. Sample Paths of Y(t) and X(t).



NOTE: NOTICE THAT THE SW FAILURE RATE IS PROPORTIONAL TO THE NUMBER OF ERRORS REMAINING IN THE SW. THE TRANSITION DIAGRAM IS ACTUALLY A "CONDITIONAL" TRANSITION DIAGRAM SINCE $X(t) = k$, $k \neq j$ WOULD RESULT IN A DIFFERENT SW FAILURE RATE, NAMELY ϕk.

Figure 0.0-3. Example of a System Having Both HW and SW Failure States. With $X(t) = j$ SW faults present in the system, the (constant) SW failure rate is ϕj, and the HW failure rate is λ_{HW}. The respective repair rates are μ_{HW} and μ_{SW}.

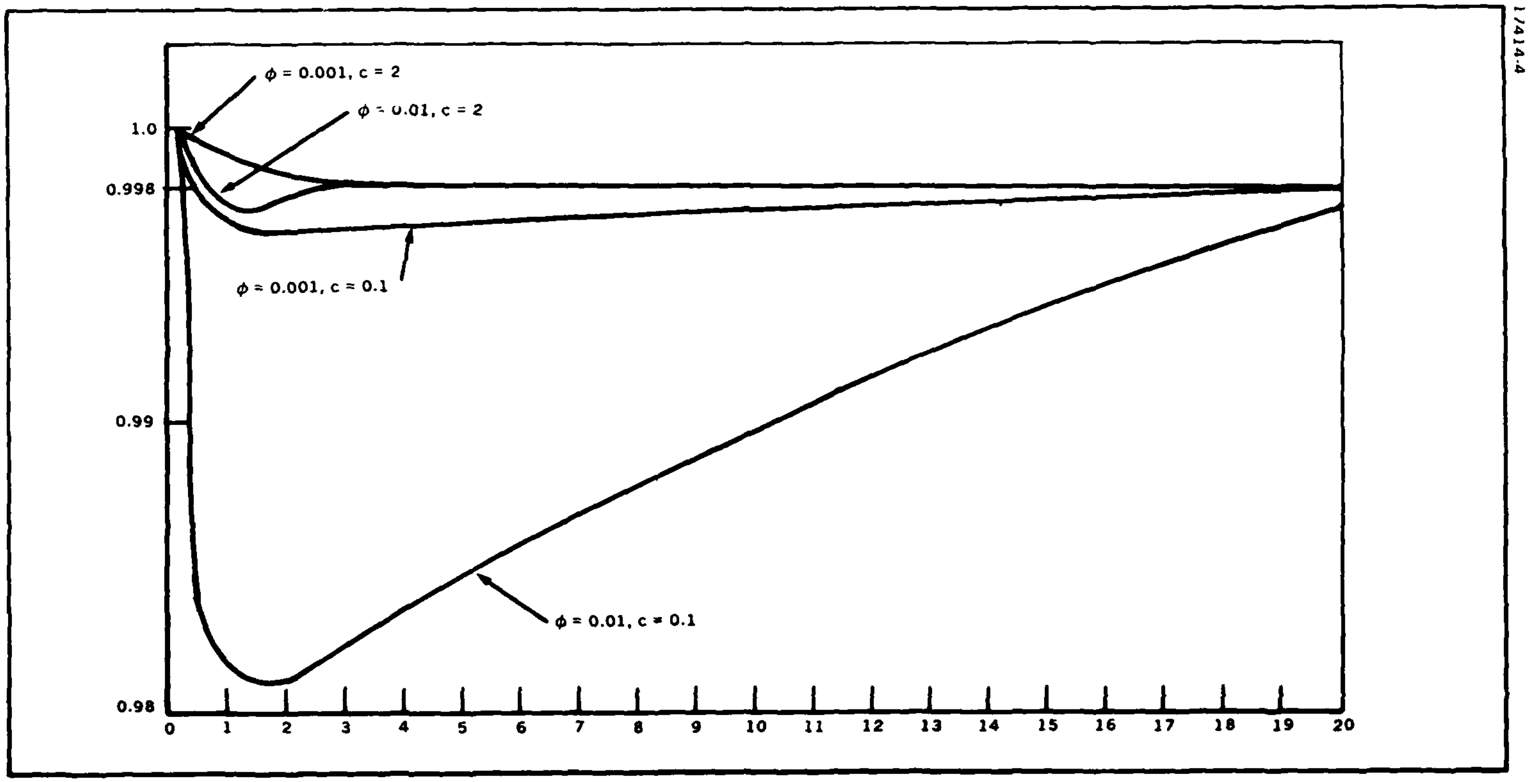


Figure 0.0-4. A(t) Versus Time

"steady-state availability" as a requirement is really the same as specifying the "worst case" value. As seen in Figure 0.0-4 however, the presence of SW can cause the availability to dip below the "steady-state" value and then approach steady-state from below. Notice in Figure 0.0-4 that each curve possesses the same "steady-state" value, and pay particular attention to the impact of the SW parameters (these parameters are explained in the report) on the size of the undershot. The implication of this is clear: for systems with substantial embedded SW, "steady-state availability" is misleading. A better measure is minimum availability.

Because of the HW/SW interfaces and interactions in a large system, HW/SW reliability modeling requires that:

a. HW/SW duty cycles be defined relative to each subsystem (i.e., a relationship between each SW module use time and subsystem hardware operating time).

b. The SW be partitioned into groups of modules which interact with specific subsystem HW items.

Thus, the combined HW/SW model must be applied to each constituent subsystem consisting of HW along with its specific SW. Figure 0.0-5 shows a typical reliability block diagram for a Command, Control, and Communications system. Table 0.0-1 gives an example of partitioning of a hypothetical Air Surveillance System's SW based on percent utilization. These data are used to determine the values of the SW parameters to be used in the combined HW/SW model (an explanation of how to use these data is given in the body of the report). Having applied the combined HW/SW reliability model to each of K constituent subsystems, the overall system reliability measures may be computed. For example, if $A_i(t)$ is the availability of subsystem i at time t as determined by applying the combined HW/SW model to subsystem i, then the system availability at time t is simply the product $A_i(t)$ x $A_2(t)$ x ... X $A_k(t)$ of the subsystem availabilities. The application of availability and other figures of merit to complex systems with embedded SW is discussed in detail.

The concepts discussed above, along with reliability tradeoff methodology between HW and SW are discussed in the body of the report.

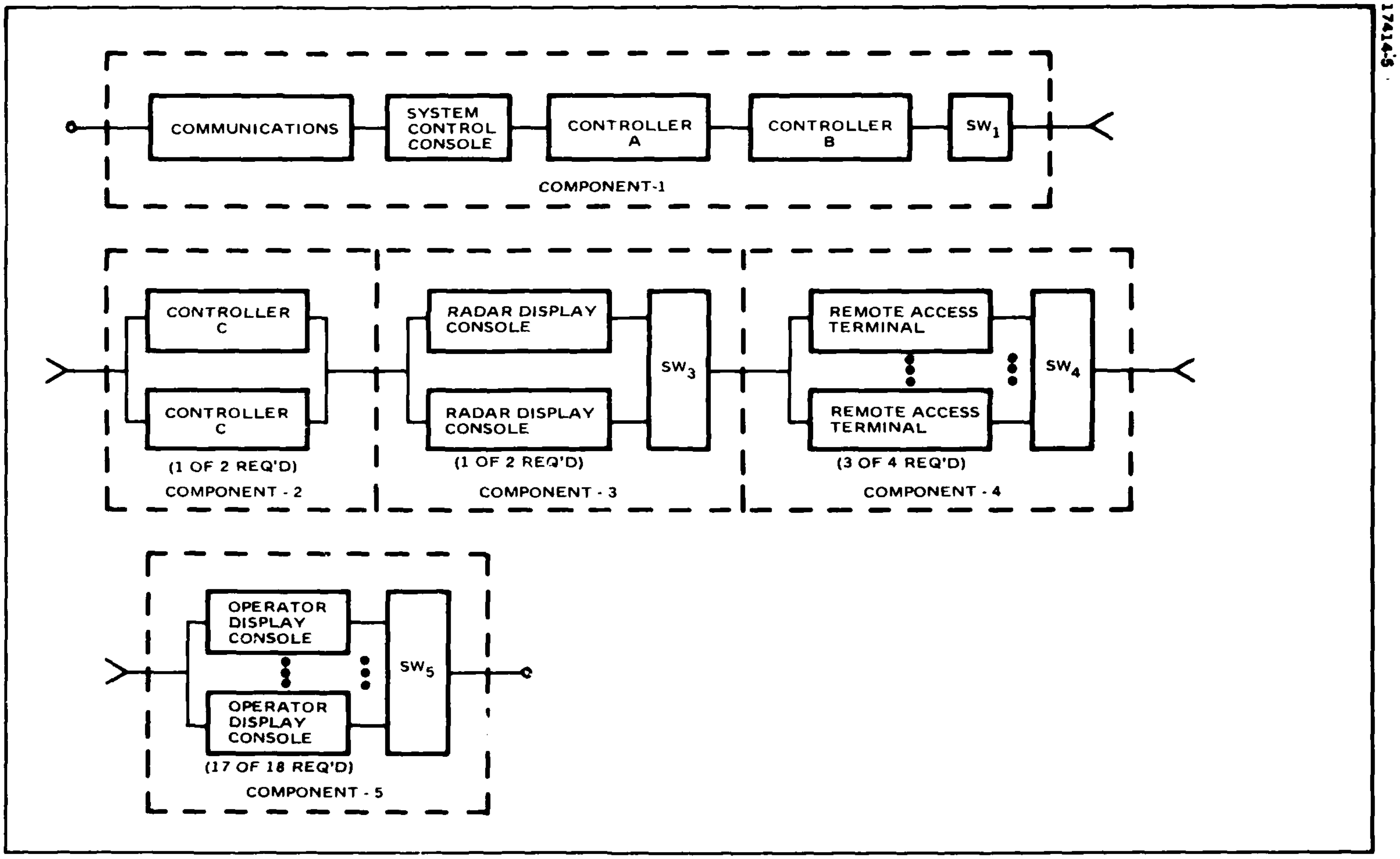


Figure 0.0-5. Reliability Block Diagram for a C^3 System

Table 0.0-1. Partitioning of an Air Surveillance System SW Based On Percentage Utilization

HW / CPCI	Communications	System Central Console	Display Controller	Peripheral Controller	Central Computer	Radar Display	Remote Access Terminal	Operator Display Console	Total
OSS	0	5	5	5	73	5	2	5	100
SUS	0	20	5	5	30	10	10	20	100
APS	20	5	0	0	10	30	5	30	100
DIS	0	15	15	15	20	15	5	15	100
DRS	0	0	0	0	75	0	25	0	100
SES	0	0	0	0	35	30	5	30	100

Section 1.0

SUMMARY OF STUDY RESULTS

The need for operational versatility in modern weapon systems has dramatically increased the importance of the embedded software (SW) used in these systems. This is particularly true for multipurpose systems which have a large number of varied human interfaces such as command, control and communications (C^3) systems. With this high SW usage, increased emphasis has been placed on SW reliability. A literature search conducted early in the study revealed that many efforts have been undertaken in recent years to develop techniques for quantifying SW reliability. As a result, SW measurement and modeling methodology is fast approaching a point where it is on par with hardware (HW) reliability methodology. A large number of SW reliability models have been developed and have, in varying degrees of success, been validated to error data. Accordingly, Hughes is currently under contract to evaluate ten of the more promising SW models using error data collected from an on-going C^3 system*.

Therefore, with the state of SW reliability methodology approaching that of HW, the next logical step is to combine the two disciplines into a common reliability methodology, which was the purpose of this study. Specifically, the objectives of this study were to 1) develop the necessary technical foundation for combining HW and SW reliability into common figures-of-merit (FOM's) which have conventional (i.e., consistent with HW) reliability interpretations, 2) develop models/procedures which apply to HW versus SW tradeoffs with respect to combined HW/SW reliability measures, and 3) define specific FOMs and tradeoff models/procedures which are applicable to C^3 systems.

Aside from the usual failure and repair processes that take place under purely HW considerations, there are three additional types of random phenomena that must be considered in deriving a combined HW/SW reliability model. These are: 1) the SW failure process, 2) the SW "repair" process (i.e., a remedial action that restores the system to an operational state without correcting the SW fault), and 3) the SW fault correction process which, if successful, affects the SW failure process. In this investigation, a general methodology for combining

*Reliability Model Demonstration Study, Contract No. F30602-80-C-0273.

traditional HW reliability models with SW reliability models has been developed around the theory of Markov Processes. Appendix B provides a brief theoretical background in Markov Processes as they apply to the HW and SW failure and repair phenomena discussed in this report.

Figure 1.0-1 provides an overview of the material presented in this report. The general flow in the main body of the material is oriented toward the reliability practitioner as much as possible with the necessary background theory and most of the more complex developments provided in the Appendices. The development of the combined HW/SW reliability model and associated reliability measures are developed in generality (Section 3.0) without specifying the nature of the SW processes. To be a useful working model, however, the SW as well as the HW processes must be well defined. This is done in Section 4.0 where the general HW/SW model is applied to some simple reliability configurations (or constructs) using SW models selected from the literature. Specifically, the SW reliability theory of Jelinski-Moranda (1972), Goel-Okumoto (1978), and standard HW reliability theory (e.g., as described in Barlow and Proschan (1965, 1975) and Kozlov and Ushakov (1970)) were incorporated in the general model to produce a working HW/SW reliability model. The details of going from the general model to this working model are provided in Appendix D. A similar development would be required for another choice of HW and SW models. Computational aspects related to the HW/SW model are presented and an experimental computer program for the model is described and documented.

The extension of this HW/SW model to more complex reliability constructs is given in Section 5.0 with an example application to a C^3 system. A system model is described based on operational-mission tasks using the simple reliability constructs of Section 3.0 as "buildng blocks". An attempt is also made at unifying failure and repair concepts based on the previously selected HW and SW models.

Finally, a HW/SW tradeoff procedure is established in Section 6 using various interpretations of the combined HW/SW availability measure. Isometric availability curves are used to specify feasible ranges of HW versus SW complexity in terms of failure rate. Example applications to specific HW/SW reliability constructs are also provided.

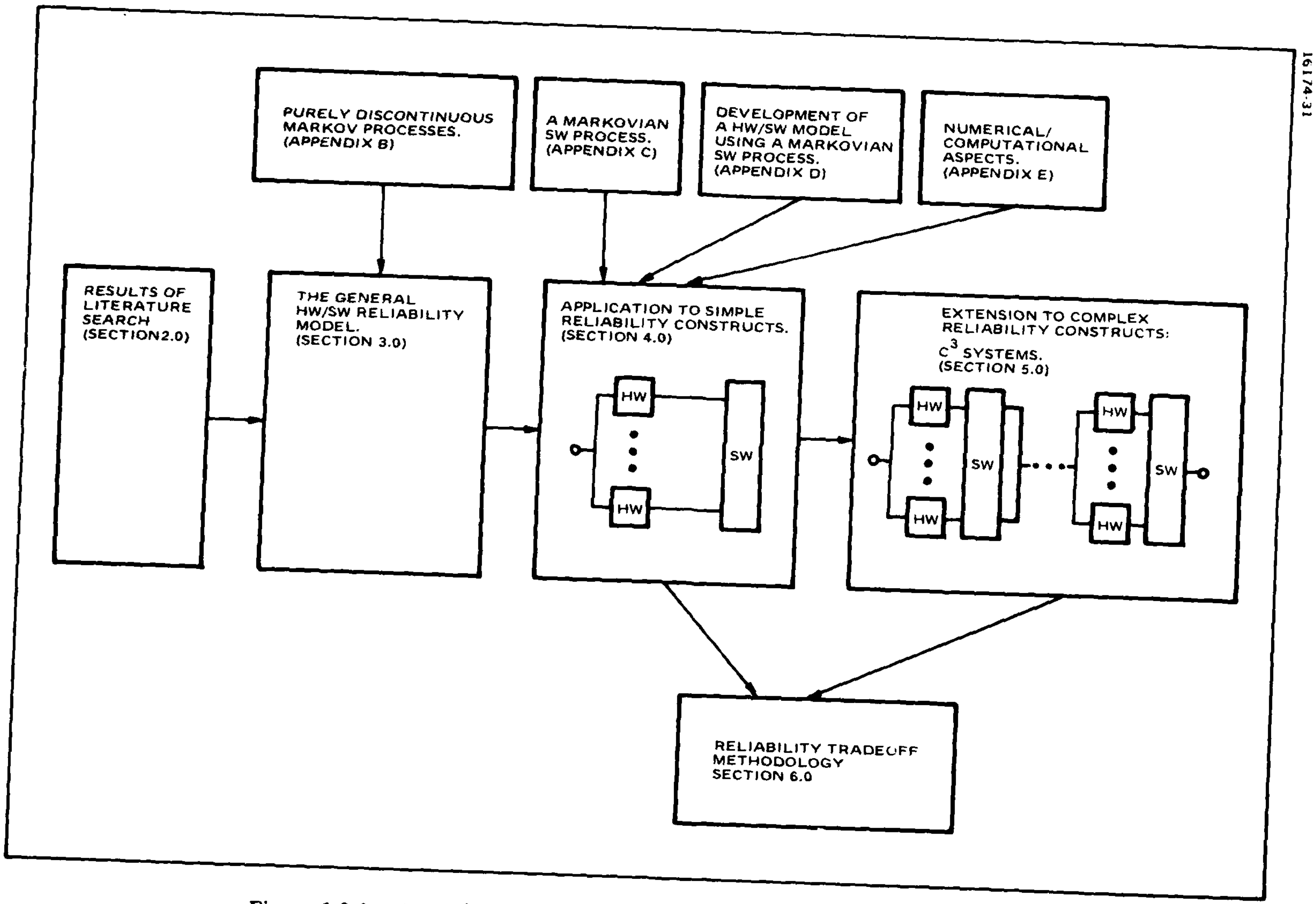

16174-31
PURELY DISCONTINUOUS MARKOV PROCESSES. (APPENDIX B)
A MARKOVIAN SW PROCESS. (APPENDIX C)
DEVELOPMENT OF A HW/SW MODEL USING A MARKOVIAN SW PROCESS. (APPENDIX D)
NUMERICAL/ COMPUTATIONAL ASPECTS. (APPENDIX E)
RESULTS OF LITERATURE SEARCH (SECTION2.0)
THE GENERAL HW/SW RELIABILITY MODEL. (SECTION 3.0)
APPLICATION TO SIMPLE RELIABILITY CONSTRUCTS. (SECTION 4.0)
HW
HW
SW
EXTENSION TO COMPLEX RELIABILITY CONSTRUCTS: C3 SYSTEMS. (SECTION 5.0)
HW
HW
SW
HW
HW
SW
RELIABILITY TRADEOFF METHODOLOGY SECTION 6.0

Figure 1.0-1. General Arrangement of Sections Within the Technical Report

Section 2.0

RESULTS OF LITERATURE SEARCH

Eight separate literature searches were conducted through various agencies in order to identify and evaluate sources of information pertaining to combined hardware/software reliability models. These searches include all journals, reports, and technical information on software reliability and combined software/hardware reliability published since 1975.

Literally thousands of references were reported in these searches, but only one precedent was discovered for mathematical modeling of systems experiencing both hardware and software failures (Costes, et. al., 1978). Many useful references to software reliability mathematical models were found along with the well-known work in hardware reliability models (e.g. Barlow & Proschan 1965, 1975; and Kozlov & Ushakov 1970). A combined list of references and bibliography containing the highlights of the literature search is presented in Section 8.0.

There are several controlling ideas in past attempts at modeling software reliability. Many authors agree that software errors manifest themselves according to a type of Poisson process. For example, Jelinski & Moranda (1971, 1972, 1975, 1976) model the software failure process by assuming a constant failure rate between bug removals. A variation on this are the models of Schick and Wolverton (1973, 1978) which assume a piece-wise continuous failure rate. Shooman (1972, 1973, 1975, 1978) proposed a model conceptually similar to that of Jelinski & Moranda with a software failure rate proportional to the number of faults remaining in the software. A continuous analog to these models is the non-homogeneous Poisson Model of Goel & Okumoto (1980) whose error correction rate is a monotone decreasing continuous function of time.

There are many apparent controversies involving past modeling attempts. One such controversy stems from the time scale involved in the models. Musa (1975) argues that time should be measured in execution time while other authors often do not make such a specification, implying universal applicability with respect to time scales. It was concluded in Schafer et. al. (1979) that the failure of many popular software reliability models to adequately fit the available software failure data was due primarily to improper control of the software testing intensity with respect to time.

Another controversy involves the applicability of models with respect to software failure definitions. There is no standard failure definition for software with regard to operational mission although it is implied in the literature that the models are applicable to software failures under any definition. This cannot be so since there are important structural differences between software "bug's" (e.g., documentation errors vs. logic errors).

Finally, there are certain features which are thought by most authors to be very important for a software reliability model to possess. The model must, of course, be mathematically tractable as were most models revealed in the literature search. Although no one model was especially simple to apply, the procedures for the most part are easily implemented on a digital computer.

Other features which a software reliability model should have are the ability to account for imperfect debugging (Goel & Okumoto, 1978, 1980), the ability to account for the phenomenon of recovering the working system without fixing the software fault (Costes, et. al. 1978), software maintenance (Trivedi & Shooman 1975, Costes, et. al. 1978), and a mechanism by which faults generated as a direct result of erroneous input data (as opposed to program bugs) can be handled (Littlewood 1979).

As mentioned earlier, the literature search turned up many references on hardware reliability modeling but most are derived on the premises set forth in the classical works of Barlow & Proschan (1965, 1975), Feller (1957, 1966), Kozlov & Ushakov (1970), Lloyd & Lipow (1962), and Mann et. al. (1974). By in large, these authors treat hardware reliability-maintainability models from the standpoint of continuous-time Markov processes. Implicit in their models is the assumption of constant failure and repair rates, the latter of which is not unreasonable in view of modern fault isolation capabilities.

Costes et. al. (1978) studied the combined hardware/software reliability problem from a semi-Markov point of view. Although their approach is straightforward and simple for simple system configurations, for a complex system, their approach becomes intractable with only steady state results being available. Indeed, one cannot even write down a complete system state diagram easily for a moderately complex system structure due to the exhaustive manner in which system states are defined. Their semi-Markov approach shows definite promise for combined hardware/software reliability models but places too much importance on extraneous system states.

TABLE 2.1 LITERATURE SEARCHES

1) "Report on OC-811, Software and Hardware Reliability,"
DDC Report Number CX3108, Defense Documentation Center.
Defense Logistics Agency, Cameron Station, Alexandria, Virginia 22314,
February 14, 1980.

2) "Report on OC-812, Software Reliability,"
DDC Report Number CX3105, Defense Documentation Center
February 14, 1980.

3) "Software & Hardware Reliability,"
Search Control Number 090869, Defense Technical Information Center,
Defense Logistics Agency, Cameron Station, Alexandria, Virginia 22314
February 14, 1980.

4) "Software Reliability,"
Search Control Number 090872, Defense Technical Information Center,
February 14, 1980.

5) "Software Reliability,"
NASA Literature Search Number 42808,
February 14, 1980.

6) "Hardware Reliability,"
NASA Literature Search Number 42809,
February 14, 1980.

7) "Software Reliability,"
NASA/RECON. On-Line Retrieval Service Bibliography,
Search Number 035,
December 20, 1979.

8) "Software/Hardware Reliability,"
Lockheed Dialog Information Retrieval Service, Dialog File 12,
February 8, 1980.

Section 3.0

THEORETICAL FOUNDATIONS OF COMBINED HW-SW RELIABILITY MODELS

3.1 INTRODUCTION

This section includes a theoretical discussion of the approach to combining HW and SW reliability models. A short section providing background information on purely discontinuous Markov processes is included in Appendix B since they are the foundation of hardware reliability models (cf. Kozlov & Ushakov, 1970) and play an integral part in the combined model. The combined HW/SW reliability model and its associated reliability measures are first derived for general SW failure and repair processes. Section 4.0 then provides a derivation from the general model based on some simplifying assumptions concerning the nature of the SW process. Example applications using simple series systems and redundant systems with single unit replications will be used to examine the effects on reliability of combining the HW and SW processes. In Section 5.0, the methodology will be extended to more complex systems involving series-parallel constructs. Specifically, the methodology will be applied to a typical command, control and communication system.

3.2 THE GENERAL COMBINED HARDWARE/SOFTWARE RELIABILITY MODEL

The general approach of combining HW and SW reliability models is based on the widely accepted propositions that: (1) the state of a hardware system is adequately described by a time-stationary Markov process (cf. e.g. Kozlov & Ushakov 1970) and (2) SW possesses a constant failure rate between SW fault corrections (Jelinski & Moranda 1972, Lipow 1974, Lloyd & Lipow 1978, Shooman 1972, Trivedi & Shooman 1975). It should be pointed out that (1) requires the assumption that the HW repair times of maintained systems are exponentially distributed which is not unreasonable in view of modern fault isolation/detection capabilities.

In combining (1) and (2) above into a HW/SW reliability model it is important to first outline the features that such a model should have. Obviously, the model should account for both HW and SW failures occurring in time and the repair thereof. The model should allow for the recovery of the SW operating system after the manifestation of a fault without necessarily removing the fault (i.e. SW startover/switchover capability). The model should allow for but not necessarily be based on an independent SW support facility which, knowing the SW contains faults, works concurrently during system operations to uncover and remove such faults before they cause substantial system down time. Such support facilities are common on systems possessing complex SW such as command control and communications (C^3) systems. Finally, when an attempt is made at correcting a SW fault, the model should allow for the possibility that this attempt is unsuccessful.

This fault correction process is characterized by a purely discontinuous time stationary Markov process $\{X(t), t \geq 0\}$ taking values in the set $\{0, 1, \ldots, N\}$ which represents the number of faults remaining in the SW. Let $0 = t_0 < t_1$, $< \ldots < t_k < \ldots$ denote the random times at which the process $X(t)$ changes values. By construction, the process $X(t)$ is a right-continuous step function. So, for t in the interval $[t_k, t_{k+1})$, $X(t)$ is constant taking the value at t_k. If the process is eventually absorbed at 0 (i.e., no SW faults remaining in the system), then the sequence of jump times will be finite, say $t_0 < t_1, < \ldots < t_M$, for some (random) integer M.

Now let the possible HW/SW system states be represented by 0, 1, ..., J where "0" represents the full-up state and 1 through J represent various HW and SW degraded or failed states and let $Y(t)$ be the state of the system at time $t \geq 0$. Because SW possesses a constant failure rate as long as $X(t)$ is constant (i.e., $X(t)$ is constant in $[t_k, t_{k+1})$), then during such periods of time, it is reasonable to model $Y(t)$, the system state at time t, as a time-stationary Markov process. Figure 3.2-1 gives an example of a system having both HW and SW failure states where A_j represents the $(J+1)x(J+1)$ transition rate (or infinitesimal matrix, see B.1-8) corresponding to the system state transition diagram when $X(t) = j$, i.e., when there are j bugs in the SW (see Figure 3.2-1).

Suppose, therefore, that the state of the system at t_k is such that $X(t_k) = j$, $Y(t_k) = \ell$. Then the conditional probability that the HW/SW system is in state m at time t given the state of the system at time t_k is:

$$P\ \{Y(t) = m\ |t_k \leq t < t_{k+1},\ X(t_k) = j,\ Y(t_k) = \ell,\ t_k = \mu,\ t_{k+1} = \nu\}$$

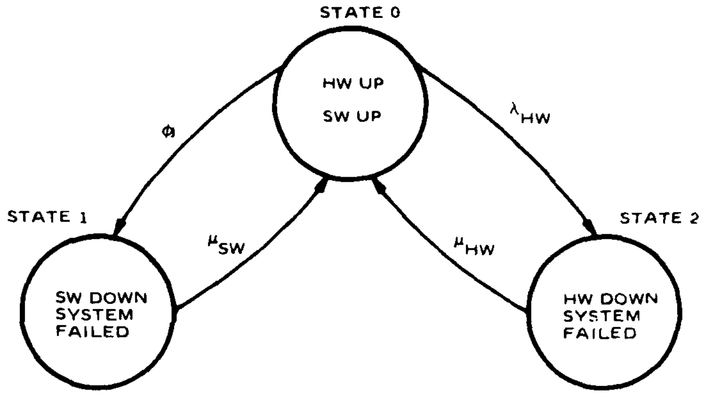
$$= p_j(t-\mu,\ \ell, m) \qquad (3.2\text{-}1)$$

where p_j is the solution to the system of differential equations (B.1-7) with A replaced by A_j, the initial condition is given by $p_j(0,\ell,\ell) = 1$, and μ and ν are the values of the random times t_k and t_{k+1}. The unconditional distribution of $Y(t)$ can now be written as:

$$P\{Y(t) = m\} = \sum_{\ell=0}^{J} \sum_{j=0}^{N} \sum_{k \geq 0} \left[\int\int_{\mu < \nu} p_j(t-\mu, \ell, m)\ P\ \left\{t_k \leq t < t_{k+1},\right.\right.$$

$$\left.\left. X(t_k) = j,\ Y(t_k) = \ell,\ t_k\ \varepsilon[\mu, \mu+d\mu),\ t_{k+1}\ \varepsilon[\nu, d\nu)\right\}\right] \qquad (3.2\text{-}2)$$

A typical sample path of $Y(t)$, $t \geq 0$, can thus be described as follows. At time $t = 0$, Y begins at state 0 (i.e., fully operational) and $X(0) = N$ (i.e., N faults are present in the SW). From time 0 until t_1, $Y(t)$ evolves according to time homogeneous Markov process with infinitesimal matrix A_N and initial value 0. At time t_1, X jumps from its initial value N to some other state $k\varepsilon\ \{0, 1, \ldots, N-1\}$. Then, for $t_1 \leq t < t_2$, $Y(t)$ evolves according to a new Markov process with infinitesimal matrix A_k and initial value $Y(t_1)$. The process continues in this fashion (see Figure 3.2-2).



(a) TRANSITION DIAGRAM OF A SYSTEM HAVING BOTH HW AND SW FAILURE STATES.

$$A_j = \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & -(\phi j + \lambda_{HW}) & \phi j & \lambda_{HW} \\ 1 & \mu_{SW} & -\mu_{SW} & 0 \\ 2 & \mu_{HW} & 0 & -\mu_{HW} \end{array}$$

(b) INFINITESIMAL MATIRX FOR HW/SW SYSTEM

NOTE: THE TRANSITION DIAGRAM (a) IS ACTUALLY A "CONDITIONAL" TRANSITION DIAGRAM SINCE X (t) = k, k ≠ j WOULD RESULT IN A DIFFERENT SW FAILURE RATE, NAMELY ϕk SIMILARLY, THE INFINITESIMAL MATRIX (b) IS ALSO CONDITIONED ON THE NUMBER OF SW FAULTS IN THE SYSTEM.

Figure 3.2-1. Example of a System Having Both HW and SW Failure States. With X(t) = j SW faults present in the system, the (constant) SW failure rate is ϕj, and the HW failure rate is λ_{HW}. The respective repair rates are μ_{HW} and μ_{SW}.

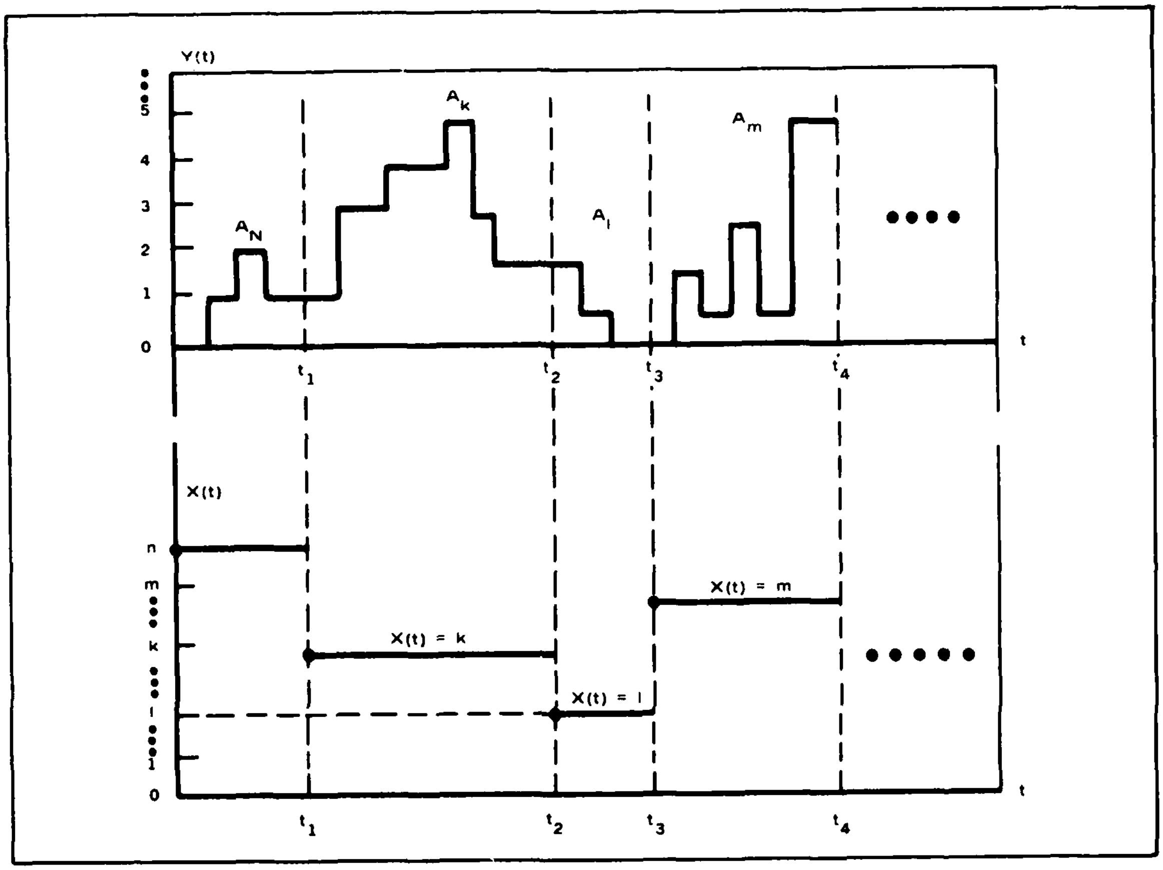
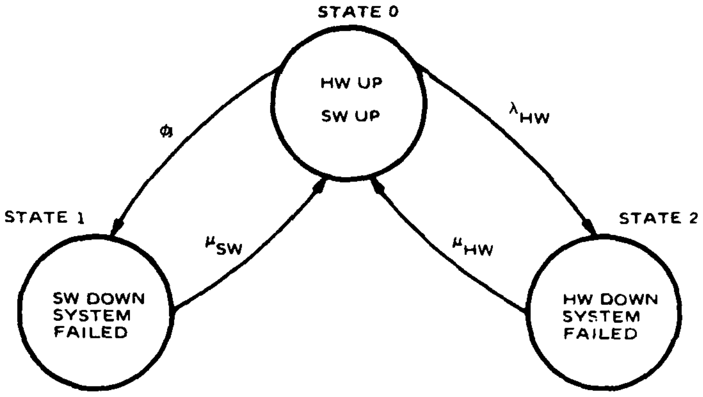


Figure 3.2-2. Sample Paths of Y(t), $t \geq 0$, X(t), $t \geq 0$

The process Y(t) is not a time-stationary Markov process in general, although it is in a sense, conditionally a Markov process. However, it should not be expected that the state of a system possessing both hardware and software faults be a time stationary Markov process because SW reliability "improves" as faults are corrected. Thus a naive approach to modelling based on strictly constant SW failure rate is not realistic.

Because of (3.2-1) and (3.2-2) and in view of the typical sample path behavior shown in Figure 3.2-2, the joint process (Y(t), X(t)) can be viewed as a Markov process with a random environment (the random environment being the Y(t)-parameter values depending on X(t)). Such models are not new. For related processes, refer to Athreya & Karlin (1971, 1971A), Kaplan (1973), Purdue (1974), Smith (1968), Smith & Wilkinson (1969, 1971), Solomon (1975), and Torrez (1978, 1979), and Cogburn & Torrez (1981).

26032-1



(a) TRANSITION DIAGRAM OF A SYSTEM HAVING BOTH HW AND SW FAILURE STATES.

$$A_j = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \left[\begin{array}{ccc} -(\phi j + \lambda_{HW}) & \phi j & \lambda_{HW} \\ \mu_{SW} & -\mu_{SW} & 0 \\ \mu_{HW} & 0 & -\mu_{HW} \end{array} \right] \end{array}$$

(b) INFINITESIMAL MATIRX FOR HW/SW SYSTEM

NOTE: THE TRANSITION DIAGRAM (a) IS ACTUALLY A "CONDITIONAL" TRANSITION DIAGRAM SINCE X (t) = k, k ≠ j WOULD RESULT IN A DIFFERENT SW FAILURE RATE, NAMELY ϕk SIMILARLY, THE INFINITESIMAL MATRIX (b) IS ALSO CONDITIONED ON THE NUMBER OF SW FAULTS IN THE SYSTEM.

Figure 3.2-1. Example of a System Having Both HW and SW Failure States. With X(t) = j SW faults present in the system, the (constant) SW failure rate is ϕj, and the HW failure rate is λ_{HW}. *The respective repair rates are* μ_{HW} and μ_{SW}.

3.3 RELIABILITY MEASURES

Equation (3.3-2) can be used to define the usual measures of reliability. The system availability at time t is defined as the probability that the system is capable of performing the mission tasks at time t. Hence, the availability, denoted by A(t), is

$$A(t) = \sum_{m\varepsilon OP} P\{Y(t) = m\} \tag{3.3-1}$$

where OP is the set of successful operating states (i.e., OP≡{m:m is an operating state}) and the summation is over all such operating states.

The system mean time to failure (MTTF) starting from an operational state is obtained by first deriving the distribution of the time to go from a given operational state to the first visit to a failed state. The general model (3.2-2) is derived under the conditions that the class of failed states for Y(t) be an absorbing class; i.e., the failed states in the transition diagrams from which the $p_j(t-u,\ell,m)$ are derived in (3.2-2) have arrows pointing to them but none pointing out of them leading to operational states. For example, the transition diagram of Figure 3.2-1 would have no return arrows from the failed states 1 and 2. This is equivalent to zeroing-out the repair rates leading from failed states to operational states. Then starting in an operational state, say state i, the timc T_{iF} until the system fails for the first time has distribution defined by

$$P\{T_{iF} \geq t \,|\, Y(0) = i\} = \sum_{m\varepsilon OP} P\{Y(t) = m \,|\, Y(0) = i\}. \tag{3.3-2}$$

Then the MTTF starting from state i is the integrated reliability function defined by

$$MTTF_i = \int_0^\infty \sum_{m\varepsilon OP} P\{Y(t) = m \,|\, Y(0) = i\}dt \tag{3.3-3}$$

Under the same conditions governing (3.3-2) and (3.3-3), the probability of failure-free operation starting in operational state i and of duration t is given by

$$P(t) = P\left\{T_{iF} \geq t \,|\, Y(0) = i\right\}. \tag{3.3-4}$$

Another measure, the non-stationary reliability coefficient, is defined as the probability that a system is operational at a moment t>0 and then operates failure-free up to the moment t + t_o, $t_o>0$ (Kozlov & Ushakov, 1970, pp. 28). Defining this quantity as $R(t,t_o)$, then

$$R(t,t_o) = P\{Y(t)\varepsilon OP,\ Y(t+t_o)\varepsilon OP\}. \qquad (3.3\text{-}5)$$

Since the process Y(t) is generally neither Markov nor time-stationary, (3.3-5) is difficult to compute. However, under certain conditions Y(t) will behave asymptotically as a time stationary Markov process (see Appendix B) so that $R(t,t_o)$ can be approximated by $R(\infty,t_o)$ to yield a "steady-state" reliability coefficient which can be shown to be equal to

$$R(\infty,t_o) = \sum_{m\varepsilon OP} P_m(t_o)\pi(m) \qquad (3.3\text{-}6)$$

where the quantities

$$\pi(m) = \lim_{t\to\infty} P\{Y(t) = m|Y(0) = k\}$$

are assumed to exist and be independent of k, and P_m (see 3.3-4)) is computed under "steady-state" conditions.

Section 4.0

APPLICATION OF METHODOLOGY TO SIMPLE RELIABILITY CONSTRUCTS

In this section some criteria will be presented for selecting models for the SW process $\{X(t), t \geq 0\}$ and these criteria will be used to show how the calculation of (3.2-2) can be accomplished.

A particular Markov model for the SW process, namely the Goel/Okumoto SW process described in Appendix C, will be adopted for the general combined HW/SW reliability model (3.2-2). With this adaptation the model will be analyzed, and expressions for the state occupancy probabilities and reliability measures defined in Section 3.2 will be derived. Computational/numerical aspects of employing the baseline model will be discussed and several specific reliability constructs will be analyzed using the baseline model. Finally, the possibilities for employing other processes for the SW model process (not necessarily those satisfying the criteria discussed earlier) will be discussed.

Employing the basic assumptions of the Goel/Okumoto model in the SW process X(t), therefore, the general HW/SW model (3.2-2) becomes

$$P\{Y(t) = n | Y(0) = i\} = e^{-Nct}\, p_N(t,i,n) + \sum_{j=0}^{N-1} \sum_{\ell=0}^{j} \sum_{m=0}^{N-\ell} \left[\binom{N}{\ell} \binom{N-\ell}{m} (-1)^{m+1}\, c(\ell+m) e^{-c(\ell+m)t} \int_0^t p_j(s,0,n) e^{-c(j-\ell-m)s} ds \right] \qquad (4.1\text{-}1)$$

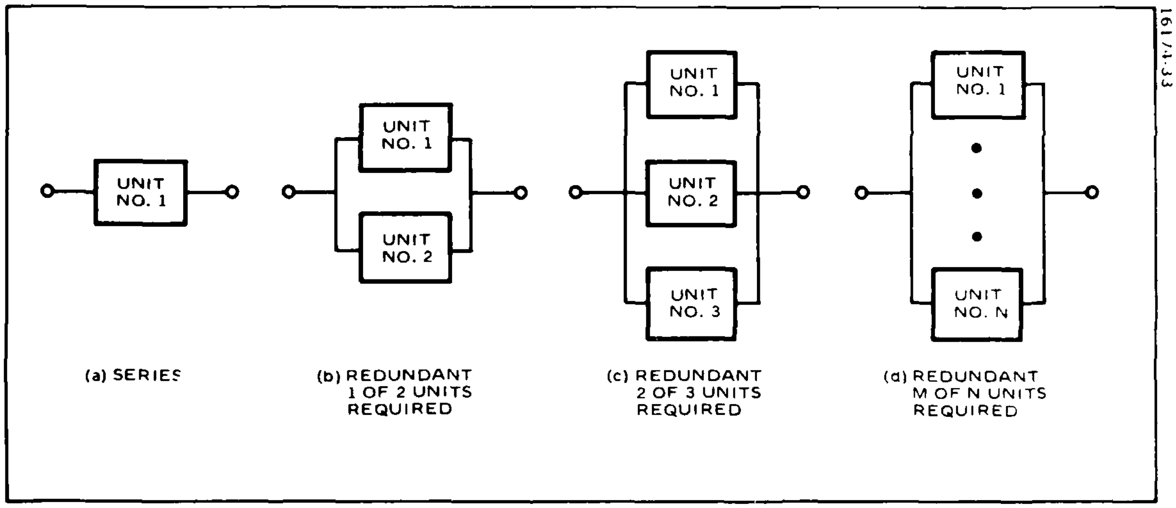
where the process Y(t), as before, takes values in the system state space $\{0, 1, \ldots, J\}$ with $Y(0) = i$ (i.e., the system starts in state i), and the process X(t) takes values in $\{0, 1, \ldots, N\}$ with $X(0) = N$ (i.e., the SW initially contains N faults). The parameter $c = \lambda p$ is the rate of fault correction where p is the probability of a "perfect" debug and λ is the rate of maintenance troubleshooting.

The detailed development of (4.1-1) from (3.2-2) using the Goel/Okumoto SW model is provided in Appendix D.

In the subsections below, (4.1-1) will be applied to some simple reliability configurations (or constructs) which involve series systems and redundant systems that are replications of a single unit. Figure 4.1-1 gives several examples of these simple reliability constructs. Complex reliability constructs (series-parallel configurations) are considered in Section 5.0.

4.1 ANALYSIS OF SERIES CONSTRUCTS

In the series examples detailed below the solutions to the Kolmogorov differential equations can be obtained in closed-form. Although in general it will be necessary to compute the reliability measures on a computer, for these cases it will not be necessary to numerically solve systems of differential equations.



(a) SERIES

(b) REDUNDANT 1 OF 2 UNITS REQUIRED

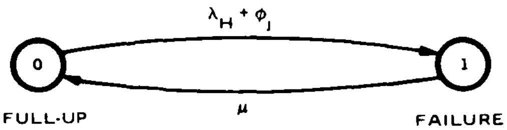
(c) REDUNDANT 2 OF 3 UNITS REQUIRED

(d) REDUNDANT M OF N UNITS REQUIRED

Figure 4.1-1. Examples of Simple Reliability Construct.

4.1.1 Two State Series Case

In this example it will be assumed that there is one HW unit and one SW "unit" in a series configuration. The SW failure rate will be ϕj where there are j faults in the SW, i.e., the SW failure rate will be $\phi X(t)$. The HW failure rate (plus any constant component of SW failure rate) will be λ_H. For this example, we will not distinguish between the different failed states (i.e. HW down, SW down) but will assume that the rate of repair from the failed state is μ, regardless of the failure type. The transition diagram is (when X(t) = j) given by



In this case, there are two distinct system states as indicated in the transition diagram (i.e., Y(t) takes values in $\{0,1\}$). The infinitesimal matrix associated with this transition diagram is

$$A_j = \begin{bmatrix} -(\lambda_H + \phi j) & (\lambda_H + \phi j) \\ \mu & -\mu \end{bmatrix} .$$

The Kolmogorov differential equations are (cf. (B.1-7)):

$$\frac{d}{dt} p_j(t,0,0) = -(\lambda_H + \phi j)\, p_j(t,0,0) + \mu p_j(t,0,1)$$

$$\frac{d}{dt} p_j(t,0,1) = -\mu p_j(t,0,1) + (\lambda_H + \phi j)\, p_j(t,0,0)$$

with initial conditions $p_j(0,0,0) = 1$, $p_j(0,0,1) = 0$. The solutions for $t \geq 0$ are

$$p_j(t,0,0) = \frac{\mu}{(\lambda_H + \phi j + \mu)} + \frac{(\lambda_H + \phi j)e^{-(\lambda_H+\phi j+\mu)t}}{(\lambda_H + \phi j + \mu)}$$

$$p_j(t,0,1) = \frac{(\lambda_H + \phi j)}{(\lambda_H + \phi j + \mu)} - \frac{(\lambda_H + \phi j)e^{-(\lambda_H+\phi j+\mu)t}}{(\lambda_H + \phi j + \mu)} \qquad (4.1\text{-}2)$$

The integration in (4.1-1) is easily performed to yield

$$\int_o^t p_j(s,0,0)e^{-c(j-\ell-m)s}ds = \frac{(1-e^{-c(j-\ell-m)t})\mu}{c(j-\ell-m)(\lambda_H + \phi j + \mu)}$$

$$+ \frac{(\lambda_H + \phi j)\left(1-e^{-[\lambda_H+\phi j+\mu+c(j-\ell-m)]t}\right)}{(\lambda_H + \phi j + \mu)[\lambda_H + \phi j + \mu + c(j-\ell-m)]}$$

and

$$\int_o^t p_j(s,0,1)e^{-c(j-\ell-m)s}ds = \frac{(\lambda_H + \phi j)\ (1-e^{-c(j-\ell-m)t})}{C(j-\ell-m)\ (\lambda_H + \phi j + \mu)}$$

$$\frac{(\lambda_H + \phi)\left(1-e^{-[\lambda_H+\phi j+\mu+c(j-\ell-m)]t}\right)}{(\lambda_H + \phi j + \mu)[\lambda_H + \phi j + \mu + c(j-\ell-m)]} \text{ .}$$

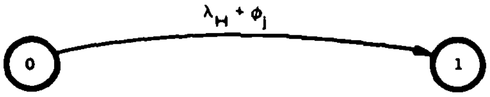
In these expressions, it should be noted that

$$\frac{1-e^{-at}}{a} = t \text{ when } a = 0.$$

Substituting these expressions into (4.1-1) with i = 0 and a fixed value for N will give expressions for $P\{\ Y(t) = q\,|Y(0) = 0\ \}$, q = 0, 1 which are easily programmed on a computer. Alternatively, the computer program in Appendix F could be used. Since there is only one operational state, the availability is

$$A(t) = P\{\,Y(t) = 1\,|Y(0) = 0\} \text{ .}$$

To compute MTTF, $P(t_o)$, and $R(\infty, t_o)$ it is necessary to make state $1 \equiv i_F$ absorbing in which case the transition diagram is now (when $X(t) = j$):



The infinitesimal matrix for this diagram is

$$A_j = \begin{bmatrix} -(\lambda_H + \phi j) & (\lambda_H + \phi j) \\ 0 & 0 \end{bmatrix}$$

and the Kolmogorov equations are then

$$\frac{d}{dt}\, p_j(t,0,0) = -(\lambda_H + \phi j)\, p_j(t,0,0)$$

$$\frac{d}{dt}\, p_j(t,0,1) = (\lambda_H + \phi j)\, p_j\,(t,0,0)$$

with initial conditions as before. The solutions are

$$p_j(t,0,0) = e^{-(\lambda_H+\phi j)t}$$

$$p_j(t,0,1) = 1-e^{-(\lambda_H+\phi j)t} \qquad (4.1\text{-}3)$$

Using (4.1-2) it is seen that

$$\lim_{t\to\infty} p_0(t,0,0) = \pi(0) = \mu/(\lambda_H + \mu) = \lim_{t\to\infty} P\{Y(t) = 0\}$$

and

$$\lim_{t\to\infty} p_0(t,0,1) = \pi(1) = \lambda_H/(\lambda_H + \mu) = \lim_{t\to\infty} P\{Y(t) = 1\}$$

which are the same results which would be obtained if there were no SW in the system.

Using (4.1-3) and since there is only one operational state, the steady state reliability coefficient is

$$R(\infty, t_o) = \frac{\mu}{(\mu + \lambda_H)} e^{-\lambda_H t_o}$$

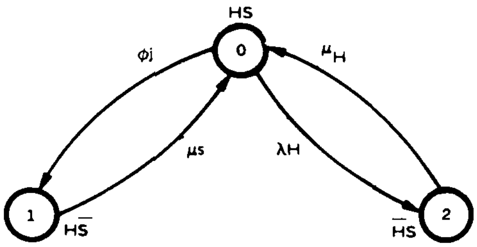
The remaining measures are computed by using

$$1 - p_j(u,0,i_F) = e^{-(\lambda_H + \phi j)u}$$

in (D.2-14) and then employing (3.3-2), (3.3-3), and (3.3-4).

4.1-2 Three State Series Case

In this example, there is again one HW unit and one SW unit in a series configuration. In this case the failed states will be: HW down, SW operational ($\overline{H}S$); SW down, HW operational ($H\overline{S}$). The full-up state will be denoted by HS. The transition diagram is (when X(t) = j) given by:



The infinitesimal matrix associated with this diagram is

$$A_j = \begin{bmatrix} -(\phi j + \lambda_H) & \phi j & \lambda_H \\ \mu_S & -\mu_S & 0 \\ \mu_H & 0 & -\mu_H \end{bmatrix}$$

The Kolmogorov differential equations are

$$\frac{d}{dt} p_j(t,0,0) = -(\phi j + \lambda_H) p_j(t,0,0) + \mu_S p_j(t,0,1) + \mu_H p_j(t,0,2)$$

$$\frac{d}{dt}\ p_j(t,0,1) = \phi j\ p_j(t,0,0) - \mu_S\ p_j(t,0,1)$$

$$\frac{d}{dt}\ p_j(t,0,2) = \lambda_H\ p_j(t,0,0) - \mu_H\ p_j(t,0,2)$$

with initial conditions $p_j(0,0,0) = 1$, $p_j(0,0,1) = p_j(0,0,2) = 0$.

Let

$$a_j = \lambda_H + \mu_H + \mu_S + \phi j$$

$$b_j = \mu_S\ (\mu_H + \lambda_H) + \phi j \mu_H$$

$$d_j = \mu_S \mu_H / b_j$$

$$r_{1j} = \frac{-a_j + \sqrt{a_j^2 - 4b_j}}{2}$$

$$r_{2j} = \frac{-a_j - \sqrt{a_j^2 - 4b_j}}{2}$$

$$C_{1j} = -[\lambda_H + \phi j + r_{2j}(1-d_j)] \Big/ \sqrt{a_j^2 - 4b_j}$$

$$C_{2j} = [\lambda_H + \phi j + r_{1j}(1-d_j)] \Big/ \sqrt{a_j^2 - 4b_j} \quad .$$

Then the solutions to the differential equations are

$$p_j(t,0,0) = d_j + C_{1j}e^{r_{1j}t} + C_{2j}e^{r_{2j}t}$$

$$p_j(t,0,1) = \frac{\phi j d_j}{\mu_S}\left(1-e^{-\mu_S t}\right) + \frac{\phi_j C_{1j}}{(\mu_S + r_{1j})}\left(e^{r_{1j}t} - e^{r_{2j}t}\right)$$

$$+ \frac{\phi j C_{2j}}{(\mu_S + r_{2j})}\left(e^{r_{2j}t} - e^{r_{1j}t}\right)$$

$$p_j(t,0,2) = \frac{\lambda_H d_j}{\mu_H}\left(1-e^{-\mu_H t}\right) + \lambda_H C_{1j}\left(e^{r_{1j}t} - e^{-\mu_H t}\right)$$

$$+ \frac{\lambda_H C_{2j}}{(\mu_H + r_{2j})}\left(e^{r_{2j}t} - e^{-\mu_H t}\right) \qquad (4.1\text{-}4)$$

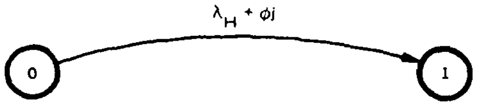
with the provision that

$$\frac{e^{at} - e^{bt}}{a-b} \equiv t$$

when a = b.

The integrations in (4.1-1) are straightforward but will not be exhibited because the notation is cumbersome. Nevertheless, (4.1-1) is now easily programmed on a computer to yield the state-occupancy probabilities and availability.

By collapsing the failed-states into one absorbing failed state $1 \equiv i_F$ the state diagram becomes:



and the solution to the Kolmogorov equations in this situation is identical to (4.1-3) with $\mu = \mu_H$ so that the last comment of Section 4.1-1 applies here also. Letting $t \to \infty$ in (4.1-4) gives

$$\lim_{t \to \infty} p_0(t,0,0) = \pi(0) = \mu_H/(\lambda_H + \mu_H) = \lim_{t \to \infty} P\{Y(t) = 0\}$$

$$\lim_{t\to\infty} p_0(t,0,1) = \pi(1) = 0 = \lim_{t\to\infty} P\{Y(t) = 1\}$$

$$\lim_{t\to\infty} p_0(t,0,2) = \pi(2) = \lambda_H/(\lambda_H + \mu_H) = \lim_{t\to\infty} P\{Y(t) = 2\}.$$

The steady-state reliability coefficient is therefore given by

$$R(\infty,t_o) = \left\{\mu_H/(\lambda_H + \mu_H)\right\} e^{-\lambda_H t_o}$$

The expressions (4.1-4) were used in (4.1-1) with

$$\phi = 1$$

$$N = 5$$

$$\mu_s = \mu_H = 2$$

$$\lambda_H = 0.004$$

$$c \equiv \lambda p = 0.95$$

and programmed on a computer (a description of the program is provided in Appendix F). A plot of availability as a function of time is pictured in Figure 4.1-2. The steady state value is (since there is only one operational state) $\mu H/(\lambda H + \mu H) = 2/2.004 = 0.9980$.

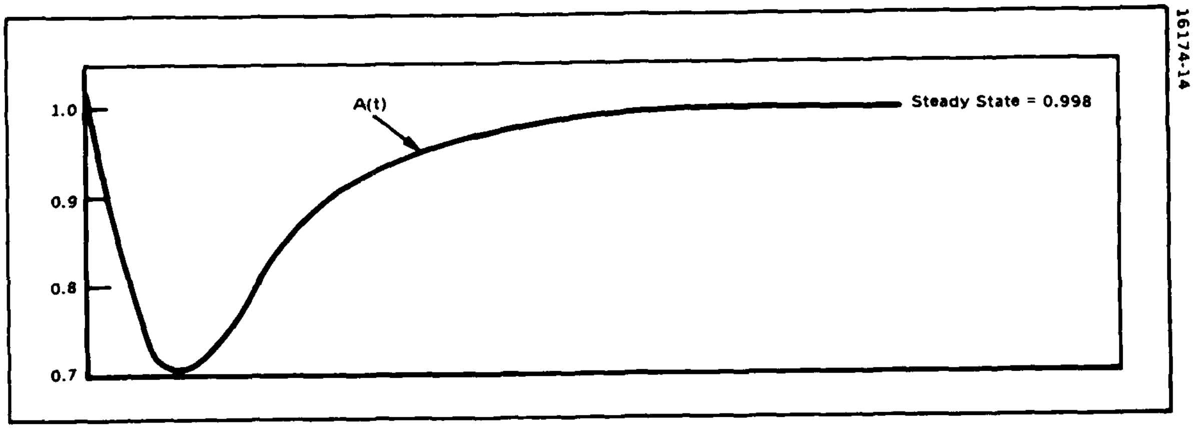


Figure 4.1-2. A(t) versus Time: $\phi = 1$, c = 0.95

The severe "undershoot" of availability beneath the steady-state is noteworthy. This behavior is typical whenever SW failures occur often enough to be important without being removed very quickly. Figure 4.1-3 shows additional plots for combinations of ϕ (effecting SW failure rate) and $c \equiv \lambda p$ (effecting the need of fault corrections). As ϕ decreases (c fixed) the undershoot becomes less severe and eventually non-existent. Of course, as ϕ decreases, this means that the faults existing in the SW do not manifest themselves often so that in the time it takes to remove the faults, the SW does not fail often enough to cause severe undershoot. The constant c determines, roughly, the rate at which faults are removed from the SW. When c is small, it causes the rise from the undershoot to be very long and slow, thus postponing steady-state conditions greatly.

Most importantly, these considerations suggest that instead of using the traditional steady-state availability as a single measure of effectiveness it would be better to consider the minimum value of availability, since each graph in Figures 4.1-2 and 4.1-3 possesses the same steady-state value.

4.2 ANALYSIS OF CONFIGURATIONS WITH REDUNDANT HARDWARE

In this section a configuration with redundant HW units interacting with SW will be discussed. A general transition diagram and infinitesimal matrix will be described and some examples will be analyzed using the computer program documented in Appendix F. Modifications to incorporate various maintenance scenarios will also be discussed.

4.2.1 Transition Diagrams and Infinitesimal Matrices

For this example it is assumed that there are N_H identical HW units, M of which are required for operation of the HW portion of the system. For now, it will be assumed that the HW standby units are operational (i.e., hot standby), and that each HW unit is accessing the SW equally. It will also be assumed that only one HW unit can be repaired at a time, and that additional HW units do not fail while the SW is down (e.g., the SW is being patched). The transition diagram corresponding to this scenario (when $X(t) = j$) is shown in Figure 4.2-1.

In Figure 4.2-1, states are assigned integer values and labeled according to the condition of the system. For example, H_kS is the state in which there are k HW units down but the SW is operational. Similarly, $H_k\bar{S}$ is the state in which there are k HW units down and the SW is down. The states "$H_k\bar{S}$" $k = 0, 1, \ldots, N_H-M$ are system failed states, while the states H_kS, $k \leq N_H-M$, are operational states, although for $k > 1$, they may be degraded operational states.

Notice that the rate of transition from H_kS to $H_k\bar{S}$ is $\phi(N_H-k)j$ since there are k less HW units accessing the SW. If it is desired to have unlimited repair of the HW (i.e., unlimited repair resources), then the value of μ_H must be multiplied by suitable constants depending on the number of HW units failed at each transition. Notice that in this case, we are tacitly assuming that all of the HW units are accessing the same SW unit, rather than employing redundant SW units.

26032-3

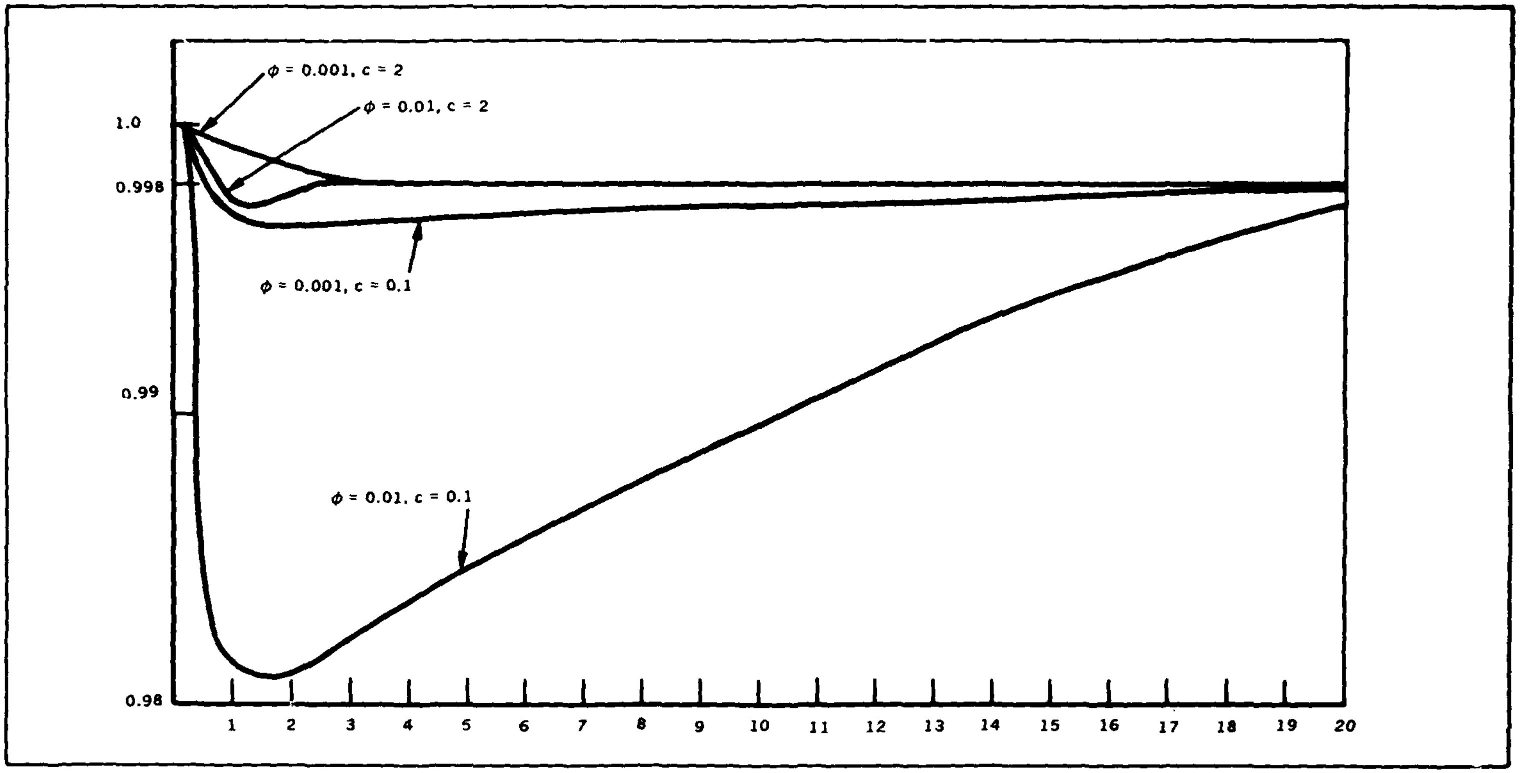


Figure 4.1-3. A(t) versus Time: ϕ = 0.001, 0.01, c = 0.1, 2

The infinitesimal matrix associated with Figure 4.2-1 is easily written down for specific cases but is cumbersome to express in general. However, it is clear what the entries should be from the figure. For example, denoting $A_j(x,y)$ to be the (x,y) entry in matrix A_j(cf, B.1-8):

$$A_j(0,1) = \phi N_H j,\ A_j(0,2) = N_H\lambda_H,\ A_j(0,0) = -(\phi N_H j+N_H\lambda),\ A_j(0,k) = 0 \text{ for } k > 2;$$

$$A_j(1,0) = \mu_S,\ A_j(1,1) = -\mu_S,\ A_j(1,k) = 0 \text{ for } k >1;$$

$$A_j(2,0) = \mu_H,\ A_j(2,1) = 0,\ A_j(2,3) = \phi(N_H-1)j,\ A_j(2,4) = (N_H-1)\lambda_H,$$

$$A_j(2,2) = -\left[\mu_H+\phi(N_H-1)j+(N_H-1)\lambda_H\right]\ .\ A_j(2,k) = 0 \text{ for } k > 4;$$

etc.

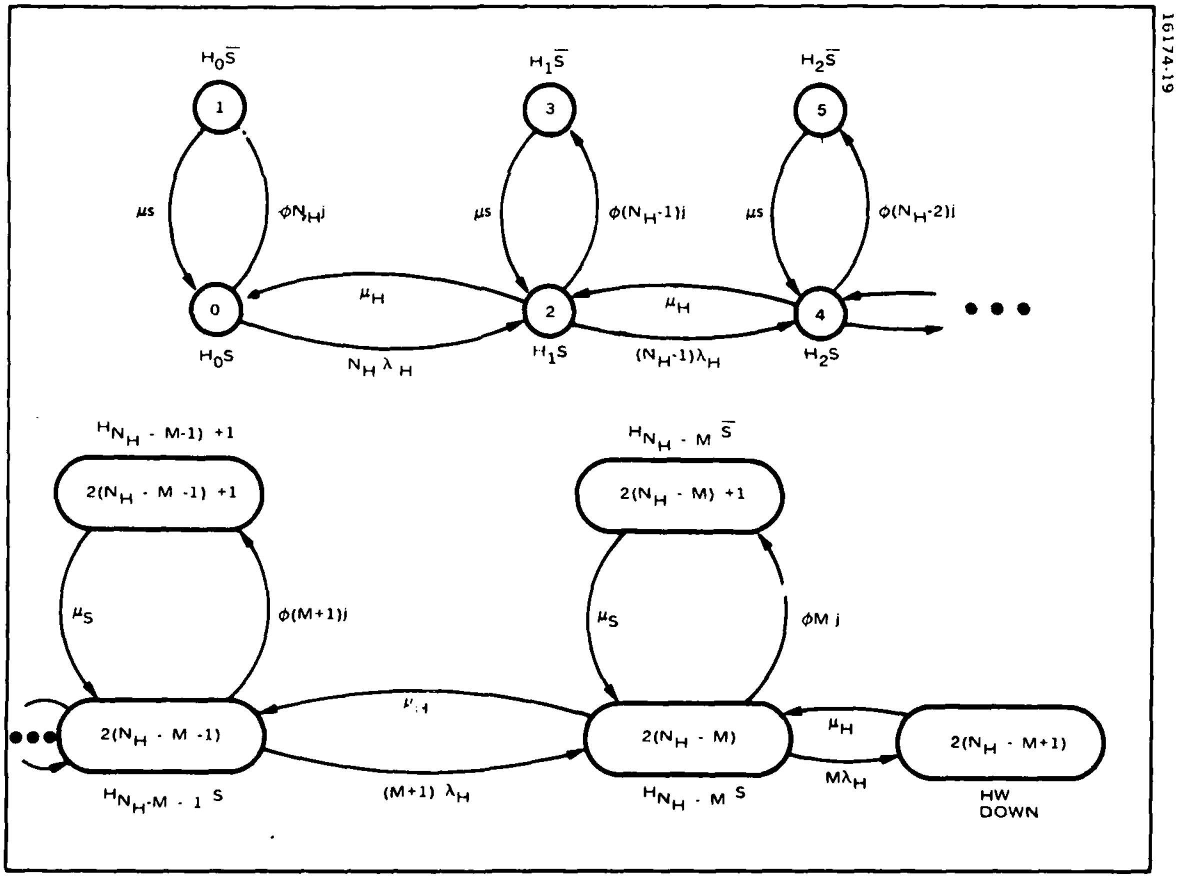


Figure 4.2-1. Transition Diagram for Configuration with Redundant HW

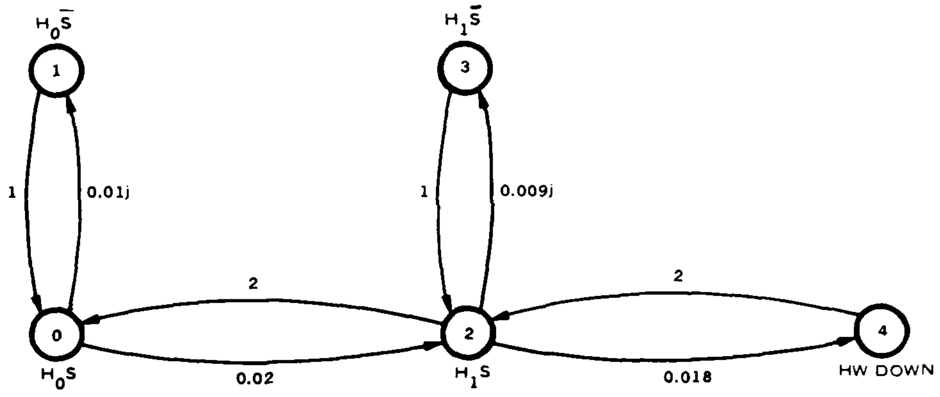
In general, to determine the value of $A_j(x,y)$ for fixed x, refer to the state x in the transition diagram. If $y \neq x$ and x has an arrow pointing away from it to y, then $A_j(x,y)$ is equal to the transition rate associated with this arrow. If there is no arrow, $A_j(x,y) = 0$. The value of $A_j(x,x)$ is then

$$A_j(x,x) = -\sum_{y \neq x} A_j(x,y).$$

4.2.2 Examples

This first example has (using the notation of Section 4.2.1) $M = 9$, $N_H = 10$, $\phi = 0.001$, $\mu_S = 1$, $\lambda_H = 0.002$, $\mu_H = 2$. The parameters for the software process $X(t)$ are c $(\equiv \lambda p) = 0.95$ and $N = 10$ SW bugs present initially.

The transition diagram (when $X(t) = j$) is:



The infinitesimal matrix corresponding to this diagram is

$$A_j = \begin{bmatrix} -(0.01j+0.02) & 0.01j & 0.02 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 2 & 0 & -(2.018+0.009j) & 0.009j & 0.018 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 2 & 0 & -2 \end{bmatrix}.$$

The computer program was used to compute the probabilities (using Equation 4.1 1)

$$P\{Y(t) = i | Y(0) = 0\}, \; i = 0, 1, 2, 3, 4$$

for t = 0, 1, ..., 15 hours and the results are reproduced in Table 4.2-1. The steady-state probabilities are

$$\pi(0) = 0.99001079$$

$$\pi(1) = 0$$

$$\pi(2) = 0.00990011$$

$$\pi(3) = 0$$

$$\pi(4) = 0.00008910$$

which are derived by solving the equations (c.f. Equations D.2-15)

$$\begin{cases} \sum_{x=0}^{4} \pi(x)\, A_0(x,y) = 0, \; y = 0, 1, \dots, 4 \\ \sum_{x=0}^{4} \pi(x) = 1. \end{cases}$$

The "successful" operational states are $\{0,2\}$ so that the availability A(t) is computed by adding the probabilities for states 0 and 2 for each t. The availability for t = 0, 1, 2, ..., 8 hours are shown in Table 4.2-2.

The next example has (using the notation of 4.2.1) $M = 8$, $N_H = 10$, $\phi = 0.005$, $\mu_S = 1$, $\lambda_H = 0.002$, $\mu_H = 1$. The parameters for the SW process are $c\ (\equiv \lambda p) = 0.95$, and $N = 10$ SW bugs initially. The transition diagram (when $X(t) = j$) is:

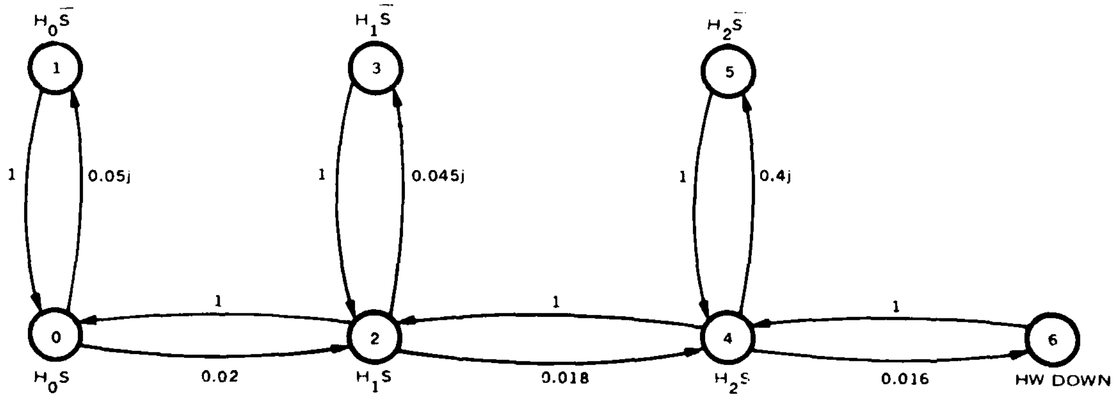


TABLE 4.2-1. STATE OCCUPANCY PROBABILITIES FOR THE FIVE STATE EXAMPLE

	State				
Time	0	1	2	3	4
0.	0.100000D+01*	0.0	0.0	0.0	0.0
1.	0.989678D+00	0.727575D-02	0.302031D-02	0.184385D-04	0.742655D-05
2.	0.989760D+00	0.514215D-02	0.505607D-02	0.212464D-04	0.207453D-04
3.	0.989853D+00	0.306042D-02	0.702866D-02	0.175062D-04	0.408468D-04
4.	0.989927D+00	0.156190D-02	0.843919D-02	0.109223D-04	0.606695D-04
5.	0.989973D+00	0.711086D-03	0.923609D-02	0.559289D-05	0.746972D-04
6.	0.989995D+00	0.300635D-03	0.961948D-02	0.252502D-05	0.825525D-04
7.	0.990004D+00	0.121714D-03	0.978646D-02	0.105871D-05	0.863253D-04
8.	0.990008D+00	0.481361D-04	0.985513D-02	0.426250D-06	0.879730D-04
9.	0.990010D+00	0.188124D-04	0.988252D-02	0.168047D-06	0.886534D-04
10.	0.990010D+00	0.731019D-05	0.989327D-02	0.655696D-07	0.889256D-04
11.	0.990011D+00	0.283310D-05	0.989746D-02	0.254596D-07	0.890327D-04
12.	0.990011D+00	0.109667D-05	0.989908D-02	0.986346D-08	0.890745D-04
13.	0.990011D+00	0.424293D-06	0.989971D-02	0.381747D-08	0.890907D-04
14.	0.990011D+00	0.164118D-06	0.989995D-02	0.147684D-08	0.890970D-04
15.	0.990011D+00	0.634756D-07	0.990005D-02	0.571229D-09	0.890994D-04

*0.100000D+01 = 0.100000 x 10^1

TABLE 4.2-2. A(t) FOR THE FIVE-STATE EXAMPLE

t	A(t)
0	1.0000
1	0.9927
2	0.9948
3	0.9969
4	0.9984
5	0.9992
6	0.9996
7	0.9998
8	0.9999

Steady State availability is $\pi(0) + \pi(2) = 0.9999119$

The difference between this example and the previous example besides different parameter values is that here there is one more redundant HW unit (i.e. only 8 of 10 are required whereas before, 9 of 10 were required).

The infinitesimal matrix corresponding to this diagram is

$$A_j = \begin{bmatrix} -(0.05j+0.02) & 0.05j & 0.02 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -(1.018+0.045j) & 0.045j & 0.018 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -(1.016+0.04j) & 0.04j & 0.016 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 \end{bmatrix}$$

The successful operating states for this configuration are $\{0, 2, 4\}$.

The state occupancy probabilities are shown in Table 4.2-3, and the availability, A(t), is given in Table 4.2-4 for t = 1, 2, ..., 9 hours.

The parameter values for these examples have been selected more or less arbitrarily to illustrate the combined HW/SW reliability methodology. Due to the large number of entry variables for this type of model, any type of generalized table of availabilities is clearly impractical to generate.

4.2.3 Modifications

The redundancy model of Figure 4.2-1 can be modified in many ways to incorporate different maintenance and/or operation scenarios. One such modification, which will be discussed in more detail in Sections 5.3 and 5.4, is providing for simultaneous repair of more than one failed HW unit at a time. As it stands, Figure 4.2-1 assumes that only one HW unit at a time is repaired. To accommodate "unlimited" repair, it is necessary to multiply μ_H at each transition accordingly. For example, the rate going from state 2 to state 0 remains at μ_H, while the rate going from 4 to 2 becomes 2 $_H$ since

TABLE 4.2-3. STATE OCCUPANCY PROBABILITIES FOR THE SEVEN STATE EXAMPLE

	State						
Time	0	1	2	3	4	5	6
0.	0.100000D+01	0.0	0.0	0.0	0.0	0.0	0.0
1.	0.961336D+00	0.352307D-01	0.332400D-02	0.100018D-03	0.892746D-05	0.222040D-06	0.191890D-07
2.	0.968589D+00	0.249663D-01	0.628607D-02	0.127491D-03	0.308124D-04	0.499997D-06	0.117657D-06
3.	0.974966D+00	0.148560D-01	0.998326D-02	0.117087D-03	0.765642D-04	0.703589D-06	0.451063D-06
4.	0.978734D+00	0.755880D-01	0.134828D-01	0.806941D-04	0.141574D-03	0.661297D-06	0.114561D-05
5.	0.980208D+00	0.342651D-02	0.161093D-01	0.448392D-04	0.208903D-03	0.458748D-06	0.212348D-05
6.	0.980501D+00	0.144266D-02	0.177669D-01	0.215302D-04	0.263978D-03	0.257468D-06	0.315089D-05
7.	0.989498D+00	0.582294D-03	0.186936D-01	0.942500D-05	0.302064D-03	0.125453D-06	0.402497D-05
8.	0.980265D+00	0.229812D-03	0.191706D-01	0.390304D-05	0.325476D-03	0.557920D-07	0.466441D-05
9.	0.980162D+00	0.897024D-04	0.194028D-01	0.156563D-05	0.338725D-03	0.234290D-07	0.508294D-05
10.	0.980102D+00	0.348327D-04	0.195115D-01	0.617085D-06	0.345791D-03	0.950000D-08	0.533803D-05
11.	0.980070D+00	0.134946D-04	0.195612D-01	0.240957D-06	0.349402D-03	0.377286D-08	0.549296D-05
12.	0.980055D+00	0.522271D-05	0.195834D-01	0.936338D-07	0.351189D-03	0.148051D-08	0.556191D-05
13.	0.980047D+00	0.202044D-05	0.195932D-01	0.362967D-07	0.352052D-03	0.577070D-09	0.560340D-05
14.	0.980044D+00	0.781479D-06	0.195975D-01	0.140533D-07	0.352462D-03	0.224101D-09	0.562458D-05
15.	0.980042D+00	0.302244D-06	0.195994D-01	0.543791D-08	0.352653D-03	0.868553D-10	0.563515D-05

*0.100000D+01 = 0.100000×10^1

state 4 entails two failed HW units with repair occurring simultaneously. The other μ_H's are revised similarly. In addition, there is the possibility of making the SW repair states μ_S dependent on other factors. For example, devoting maintenance personnel to HW repairs may take resources and manpower away from SW repair efforts so that it may be desired to reduce the SW repair rate depending on how many HW units are undergoing repair.

Another desirable modification is to allow for cold standby HW units. In this case the transition rates to go from 0 to 2, 2 to 4, etc will be changed to $M\lambda_H$ since at any point in time prior to the HW down state, there are M units operating. Also, since cold units cannot access the SW, the transition rates leading to SW failed states will be changed to $\emptyset Mj$.

TABLE 4.2-4. A(t) FOR THE SEVEN STATE EXAMPLE*

t	A(t)
0	1.0000
1	0.9647
2	0.9749
3	0.9850
4	0.9924
5	0.9965
6	0.9985
7	0.9994
8	0.9998
9	0.9999

Finally, it is not necessary nor desirable always to include all HW failed states into one down state. Figure 4.2-1 can be modified so that a new state for each number of failed HW units is added. If, for example, HW continues to operate while the system is down, additional down states H_kS and $H_k\overline{S}$, $k = N_H\text{-}M+1$, $N_H\text{-}M+2$, ..., N_H can be added so that the probabilities of being in these states can be computed.

4.3 SW REDUNDANCY

The concepts involved in SW redundancy are not entirely analogous to those in HW redundancy. For example, there would be no use in employing identical SW modules for back-up purposes since identical SW will produce identical SW failures. Also, there is no such thing as SW "hot" standby since routines cannot generally execute simultaneously on the same computer. If there were redundant computers and memories in the HW portion of the system, then the concept of "Hot" SW standby would be meaningful but hardly useful because of what was stated before: identical SW modules exhibit identical bugs.

*The steady state availability is $\pi(0) + \pi(2) + \pi(4) = 0.999994$. Equations D.2-15 are used to compute $\pi(0)$, $\pi(2)$, and $\pi(4)$.

One useful type of SW redundancy is the use of back-up SW routines which are entered when error conditions are encountered in the executing routine. Such back-up routines would perform alternate operations which do not raise error conditions and which allow completion of the task in an adequate, although possibly degraded, fashion. This type of SW redundancy belongs in the realm of fault-tolerant computing and to model it would require application of the methodology in this study at the module or routine level in the SW. A separate process X(t) would be needed to keep track of the number of faults in each SW routine at time t. Since it is not likely that there are independent maintenance teams devoted to each SW module or routine, the various processes X(t) would be correlated and the problem would be intractable. This type of internal SW redundancy is best handled by reducing the constant of proportionality used in computing the SW failure rates (e.g. reduce $\emptyset$).

4.4 THE USE OF OTHER SW MODELS

The selection of a Markov model for X(t) was made for reasons of tractability and because it had desirable properties (e.g., Markov structure, $X(t) \to 0$, $t \to \infty$, allowance for imperfect debugging, etc). There is no reason why other models could not be adopted provided of course that X(t) takes non-negative integer values and provided X(t) is a step function. In removing the Markov structure associated with X(t) the probabilistic analysis of the sample paths becomes exceedingly difficult so it would be advisable (and reasonable) to adopt such structure. Outside of this necessity, most other criteria may be dropped if so desired. In addition, there is no reason why the SW failure rate need be directly proportional to X(t) (c.f. Section 4.1.1, for example). Other dependencies including polynomial and exponential could be used.

If a different process for X(t) is adopted, simulation techniques may be necessary in order to compute state occupancy probabilities. This procedure is straight-forward if Markov structure is maintained for X(t). Such a simulation would entail generation of sample paths of Y(t) and computing statistics relating to desired measures. The procedure for generating a sample path for Y(t) is described by first generating the jump-times associated with X(t). The value X(t) takes in each interval between jumps is determined from the associated parameters for X(t) (described in Appendix B). The value of X(t) = j in turn determines the matrix A_j which describes the evolution of Y(t) during the interval when X(t) = j. It should be pointed out that in many cases, the probabilities being estimated in such a simulation are likely to be very close to 1 or 0 so that the number of replications required will be large, resulting in large computer time expenditures.

4.5 APPROXIMATIONS TO THE HW/SW MODEL

Even for a simple series system, equation (4.1-1) is difficult to evaluate without the use of a computer. For complex reliability constructs involving high SW error content, computer costs may become excessive because of the precision required in some of the calculations. Using the baseline model computer program (see Appendix F) as a standard, two methods of approximating availability were examined: 1) HW and SW availabilities computed separately and then combined (i.e., $A(t) \cong A_{HW}(t) \cdot A_{SW}(t)$)and 2) HW and SW failure and repair rates lumped (i.e., $\lambda \cong \lambda_{HW} + \lambda_{SW}$ and $\mu \cong \frac{\lambda_{HW}}{\lambda} \mu_{HW} + \frac{\lambda_{SW}}{\lambda} \mu_{SW}$)

4.5.1 Series HW Constructs.

For the first method, error is maximum at the minimum availability point (i.e., where $A(t) = A^*$). The error then diminishes slowly and finally approaches zero as $t \rightarrow \infty$. The lumped parameter approximation is significantly simpler than the method of approximation only when the SW failure rate is considered constant (i.e., the number of SW errors present remain fixed). In this case, the error is maximum at steady-state.

Figure 4.5-1 compares these two methods with the "exact" calculation provided by equation (4.1-1). A maximum error of 1.5% for the first method occurs at eight hours. For the lumped parameter method however, the maximum error approaches 14.7% for large t. If SW is a significant factor in the system, therefore, and it is necessary to make an approximation to the availability, then

$$A(t) \cong A_{SW}(t) \cdot A_{HW}(t) \qquad (4.5\text{-}1)$$

is clearly a better choice. On the other hand, if SW is not significant, lumping the parameters for HW and SW may be an adequate approximation and would be easier to compute.

4.5.2 Redundant HW Constructs

As in the series HW case, the approximation represented by (4.5-1) has a maximum error at the minimum availability point and decreases for large t. This error increases almost linearly as the number of HW units increase. Figure 4.5-2 illustrates a redundant system with two standby HW units. As the total number of units increases to 20 (i.e., 18 of 20 units required for successful operation), the error in the availability computation for the example shown increases to 40%

This error is also a function of the HW and SW failure rates as Figure 4.5-3 indicates. In this comparison, a seven-state contruct is used (i.e., 5 of 7 units required) and the unit failure rates for HW and SW have both been reduced. The maximum error in this example is only about 3% which compares to 8% error for the corresponding construct in the Figure 4.5-2 example.

Lumping the SW rates with HW rates for redundant HW would have essentially the same effect as in the series system case. This method of approximation only makes sense in situations where the SW can be considered redundant in the same way that the corresponding HW units are redundant. However, it is not clear in what sense the SW could be considered redundant.

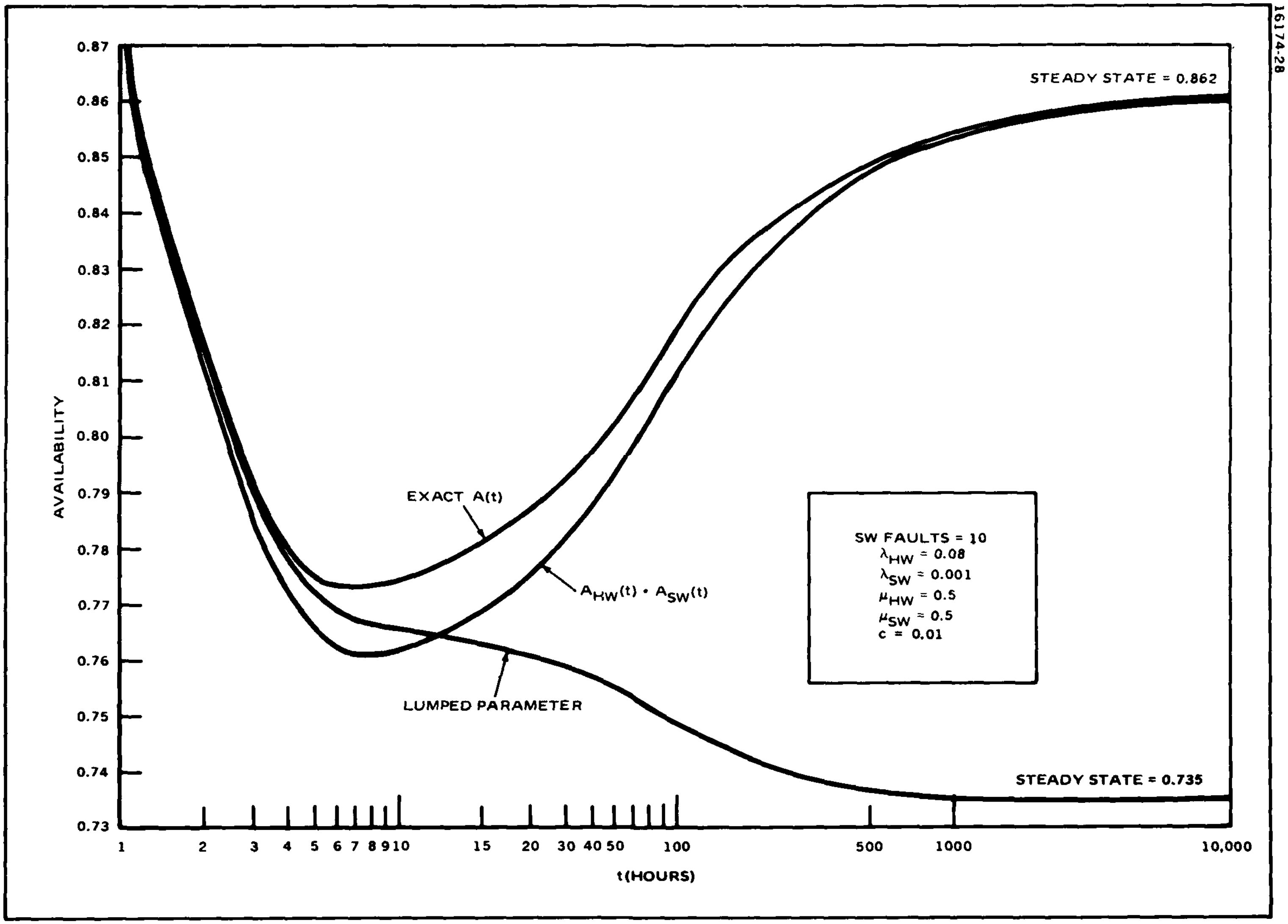


Figure 4.5-1. Availability Approximations for Series Reliability Construct

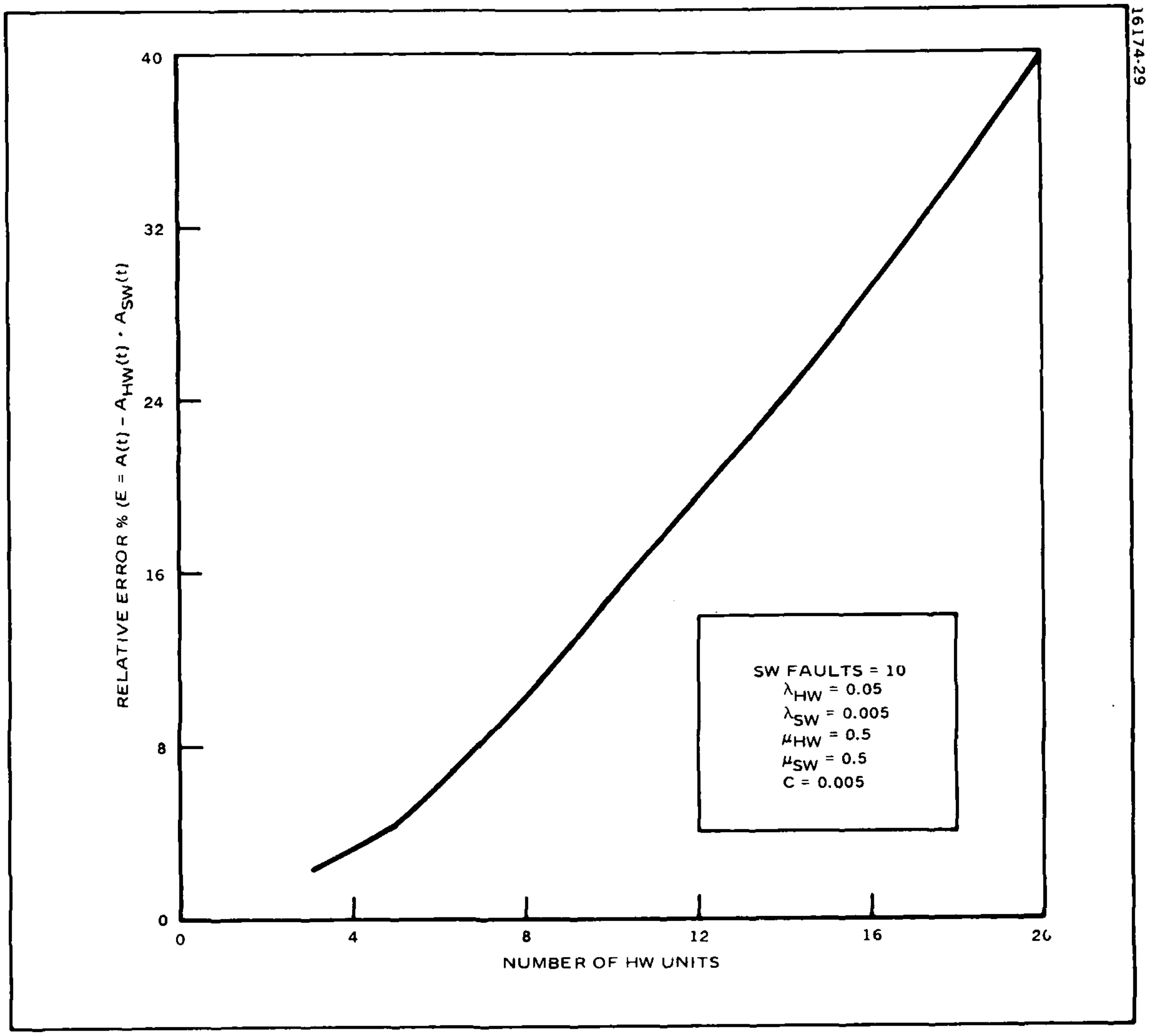


Figure 4.5-2. Error Estimate Using $A(t) \cong A_{HW}(t) \cdot A_{SW}(t)$ Versus Number of HW Units for a Two-Standby System

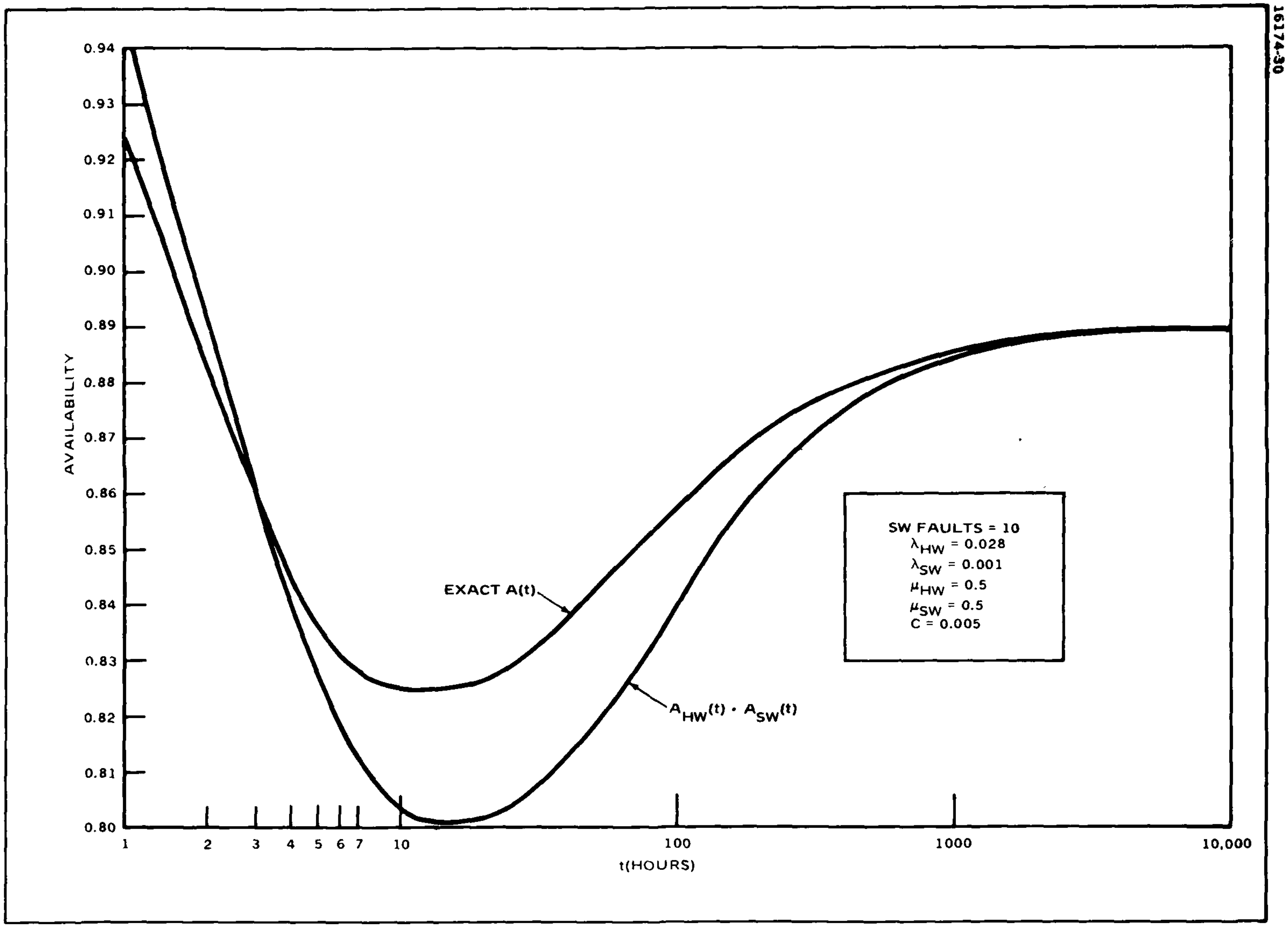


Figure 4.5-3. Availability Approximation for a Two-Standby Redundant Construct

Section 5.0

EXTENSION TO COMPLEX RELIABILITY CONSTRUCTS: C3 SYSTEMS

5.1 INTRODUCTION

The basic HW/SW reliability methodology developed in Section 3 and applied to simple reliability constructs in Section 4 will now be extended to more complex reliability constructs. Figure 5.1-1a illustrates a general reliability construct consisting of a series-parallel configuration of HW units. Each series "component" in the configuration shown in the figure is a construct of the type already discussed in Section 4. An equivalent "series" version of the general construct is given in (b). As will be seen later, complex system models will be developed around this equivalent "series" configuration.

Because of the added complexity of including SW in the reliability measures, more attention must be given to mission tasks and their HW/SW manifestations than in previous reliability methodologies where only the HW portion of the system was modeled. Moreover, for maintained systems consideration must also be given to how SW is maintained during system operation (i.e., the immediate impact of SW failures on system restoration which is analogous to the HW MTTR) as well as the fault correction process which impacts the SW failure rate. The term "mission" is used in a general sense to include both extended or ongoing operations as well as short term or one-shot operations. Depending on the purpose of the mission, therefore, the duration may be measured in hours, days, months or on-going (continuous operation) for the total life of the system. Accordingly, the selection of figures of merit (FOM's) used to measure system reliability depend on the purpose and duration of the mission. In the case of an airborne system, for example, the mission duration is typically measured in hours and, therefore, the probability of failure-free operation during the conduct of a mission is an appropriate FOM. On the other hand, the FOM for a ground-based system is better represented by the mean-time-to-failure (MTTF) or the availability.

The development of a system model, therefore, starts with 1) the statement of the system reliability requirements defined by the appropriate FOM's and 2) definition of mission tasks and the associated SW and HW required for implementation. Detailed reliability constructs of the system are then developed based on the definition of system/subsystem, alternate states of

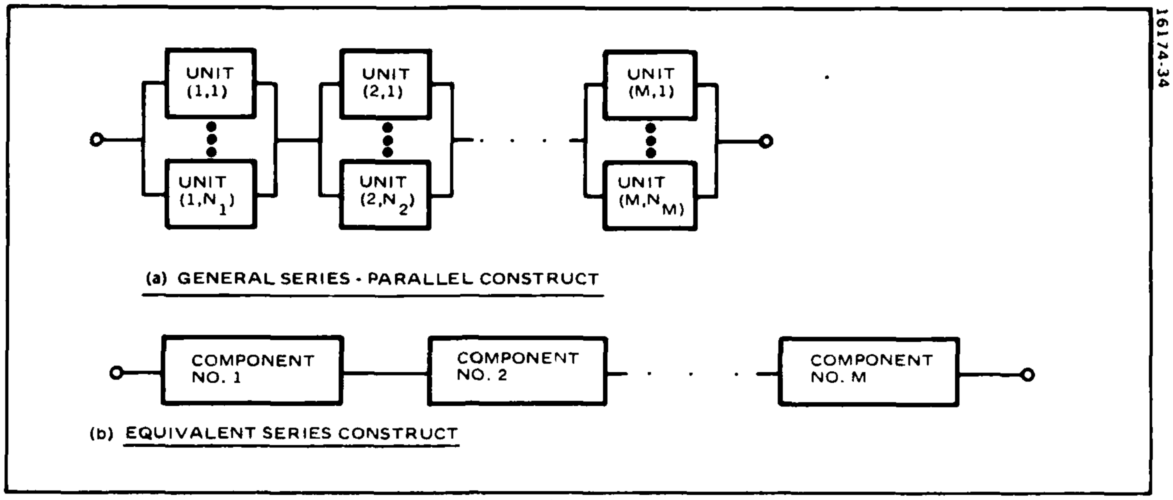


Figure 5.1-1. General Model for Complex Reliability Constructs

successful operation, degraded operation, types of redundancy, and the repair policy. The development of a combined HW/SW reliability model for a typical command, control and communications (C^3) system is illustrated in Section 5.2.

The combined HW/SW reliability theory developed in Section 3 allows treatment of the failure/repair criteria in a "normal" manner. This treatment is normal in the sense that unit HW failure and repair rates are independent of the SW rates and conversely. Thus, a reliability engineer would not be required to collect data and estimate the value of some "combined" HW/SW metric in order to compute a combined HW/SW system reliability. However, the reliability engineer will be required to select suitable SW models and estimate certain SW parameters whereas he was previously concerned only with the HW parameters. The procedures for determining unit failure rates and repair rates are given in Sections 5.3 and 5.4, respectively.

5.2 DEVELOPMENT OF THE SYSTEM RELIABILITY MODEL

5.2.1 Mission Tasks and Operating Scenarios

Mission tasks are defined based on a specified operational requirement defined, for example, to meet a potential threat. For a C^3 system, the operational requirement might be "air surveillance coverage over a specified region" and typical mission tasks would be detection, identification, track, interceptor control, etc. Figure 5.2-1 gives a simplified overview of the major HW and SW elements contained in this type of system. The numbers in the boxes represent the number of HW units.

Detailed HW and SW requirements are derived from the parameter requirements and constraints of each mission task. In the case of HW, these requirements result in defining the type of computer (i.e., processing capability, memory size, etc), the number and kind of peripheral devices, and communications equipment to external sensors. These HW units, however, do not by themselves accomplish mission tasks. They are "driven" by human operators or other stimuli through imbedded SW.

For SW, there is a more direct relationship between mission tasks and computer program components (or functions). The SW for C^3 systems is normally partitioned into functionally-oriented sets, computer program configuration items (CPCI's). For example, an air surveillance system currently in production at Hughes includes six sets as follows: operating system set (OSS), applications set (APS), support set (SUS), system exercise set (SES), data reduction set (DRS), and diagnostic set (DIS). The OSS includes such functions as confidence checking and startover, and the APS contains the functions for conducting the primary mission tasks, such as active correlation and flight plans. For an air surveillance mission, the primary function of the APS is, therefore, to establish the position of radar-reported targets and provide system operators with the capability to classify them and direct their disposition. Figure 5.2-2 provides a typical breakdown of the air surveillance mission into mission subtasks which relate directly to computer program functions. The processing is distributed into a central computer and three mini-computers (programmable peripheral controllers). Target detection data enters the system (via a communications interface) from various sensors (e.g., remote radar sites). The

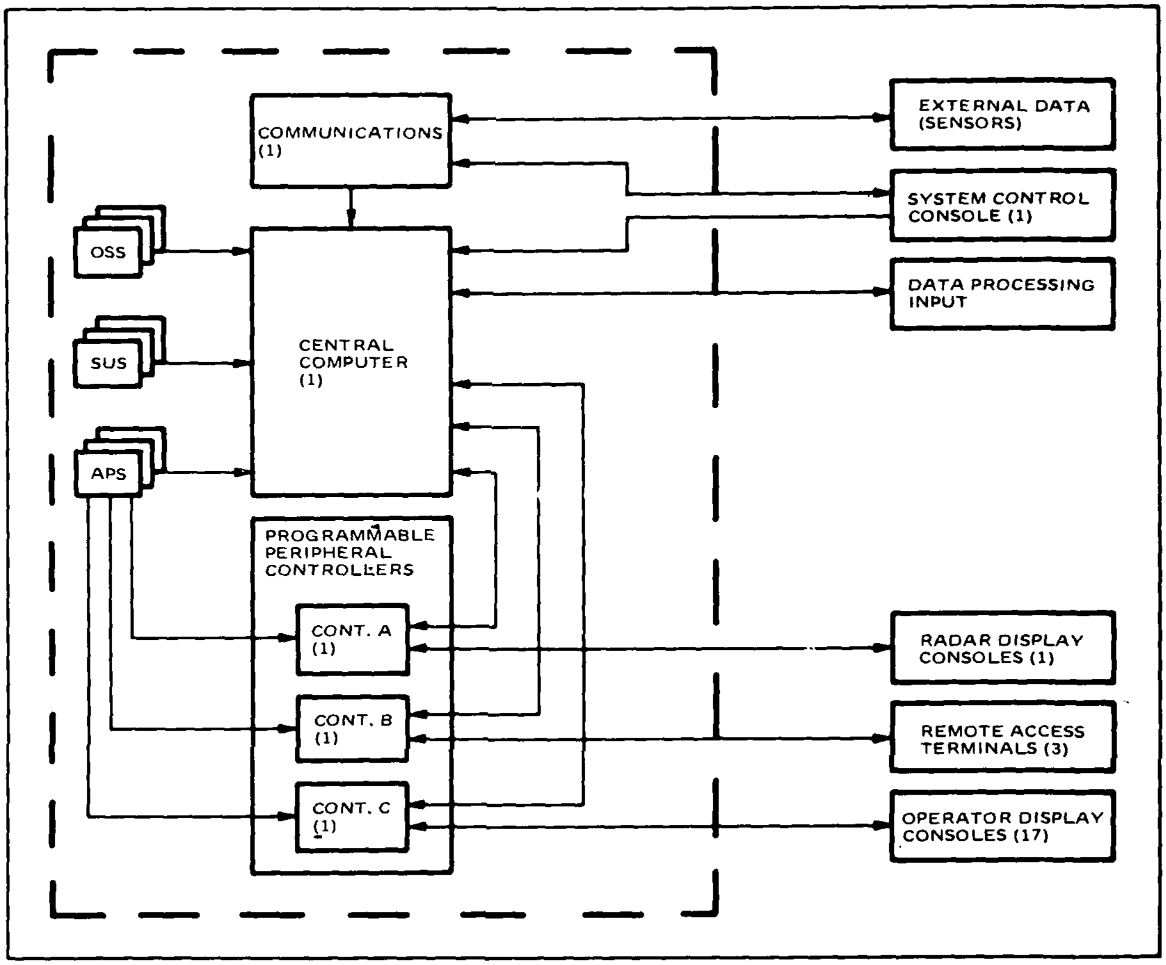

16174-26
COMMUNICATIONS (1)
OSS
SUS
APS
CENTRAL COMPUTER (1)
PROGRAMMABLE PERIPHERAL CONTROLLERS
CONT. A (1)
CONT. B (1)
CONT. C (1)
EXTERNAL DATA (SENSORS)
SYSTEM CONTROL CONSOLE (1)
DATA PROCESSING INPUT
RADAR DISPLAY CONSOLES (1)
REMOTE ACCESS TERMINALS (3)
OPERATOR DISPLAY CONSOLES (17)

Figure 5.2-1. Simplified Overview of Air Surveillance System. SW is executed via external data, direct computer input, console operators and remote access terminals.

target height is determined by the height operator using the HEIGHT function and identified using the HEIGHT and FLIGHT PLAN functions. If the target is determined to be hostile, the weapons director will dispatch the appropriate interceptor for closer investigation using the INTERCEPT CONTROL function.

Some of these mission tasks are performed on a near continuous basis (e.g., tracking air traffic) while other tasks (e.g., interceptor control) are to be performed infrequently. The fact that some mission tasks are exercised more frequently than others generally has very little effect on HW units since they are not usually dedicated to single tasks and many tasks are being conducted simultaneously. However, the effect on SW is direct and can be substantial: if a mission task is not conducted the corresponding SW is not executed and, therefore, cannot "fail." Therefore, for a given operating scenario (i.e., mission task loading during peak operating hours, peacetime conditions, battle conditions, etc), the SW functions that support the various mission tasks are "duty cycled." As in the case of HW, this duty cycle should

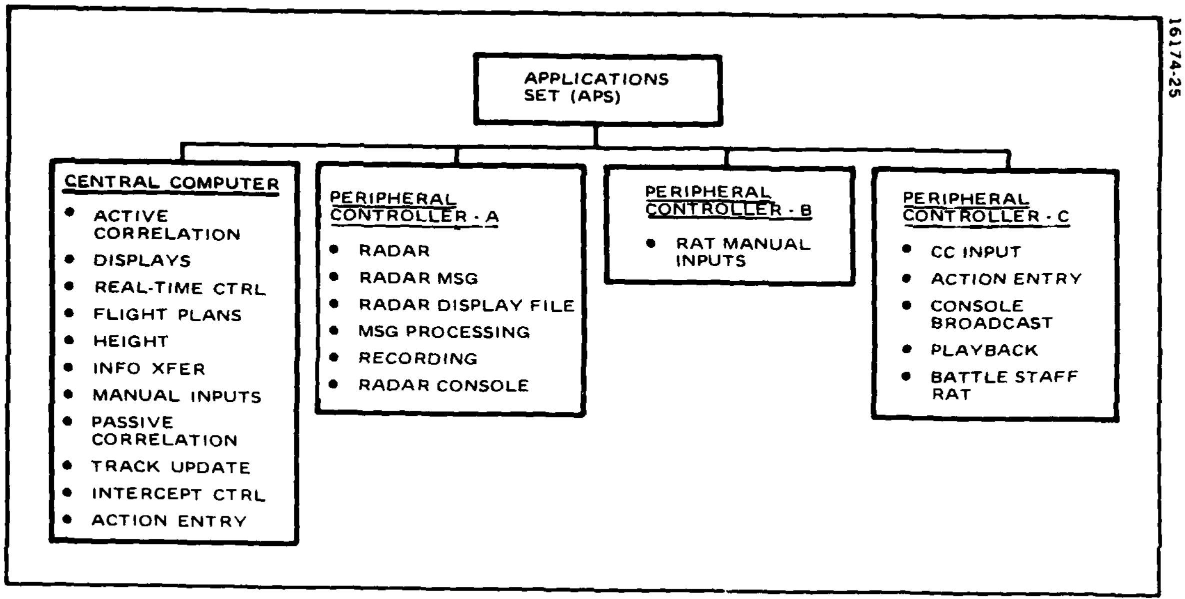


Figure 5.2-2. Air Surveillance Mission Sub-task Breakdown Into Computer Program Functions

be taken into account in allocating SW failure rates. This is discussed in more detail in Section 5.3.2.

5.2.2 System Reliability Model

With reference to Figure 5.2-1, a reliability block diagram can be constructed as shown in Figure 5.2-3. HW redundancy has been added for purpose of illustration. The block diagram shows five "system components" which relate to the reliability construct described in Figure 5.1-1: Component 1 is a series construct with SW shown as a separate series block; Component 2 is all HW and contains some unit redundancy; finally, Components 3 through 5 contain HW redundancy which interacts with SW in some known way. The system SW shown in the figure has been partitioned by the "accessing" HW units based on relative utilization during a typical mission. Note that the programmable controllers do not access SW even though they may contain memory modules, mag tape units, etc. The controller SW is accessed only by the display consoles (e.g., a human operator calls up a SW routine via the display switch action which executes the program), remote terminals and, indirectly, by external stimulus through the communications interface (e.g., automatic tracking updates to the console operators). A certain amount of care must be exercised in partitioning the SW among the accessing units within a component as well as between components. Table 5.2-1 illustrates an example partitioning of the air surveillance system represented in Figure 5.2-1. The numbers in the table represent the percentage of SW utility by each HW unit based on, for example, a system sizing and timing analysis. Thus, of the total time that the APS is executed during the performance of a typical or average mission, communications is responsible for 20%, the system control console for 5%, the central computer for 10% and the display consoles for a total of 65% (note that the table rows sum to 100% utility).

16174-35

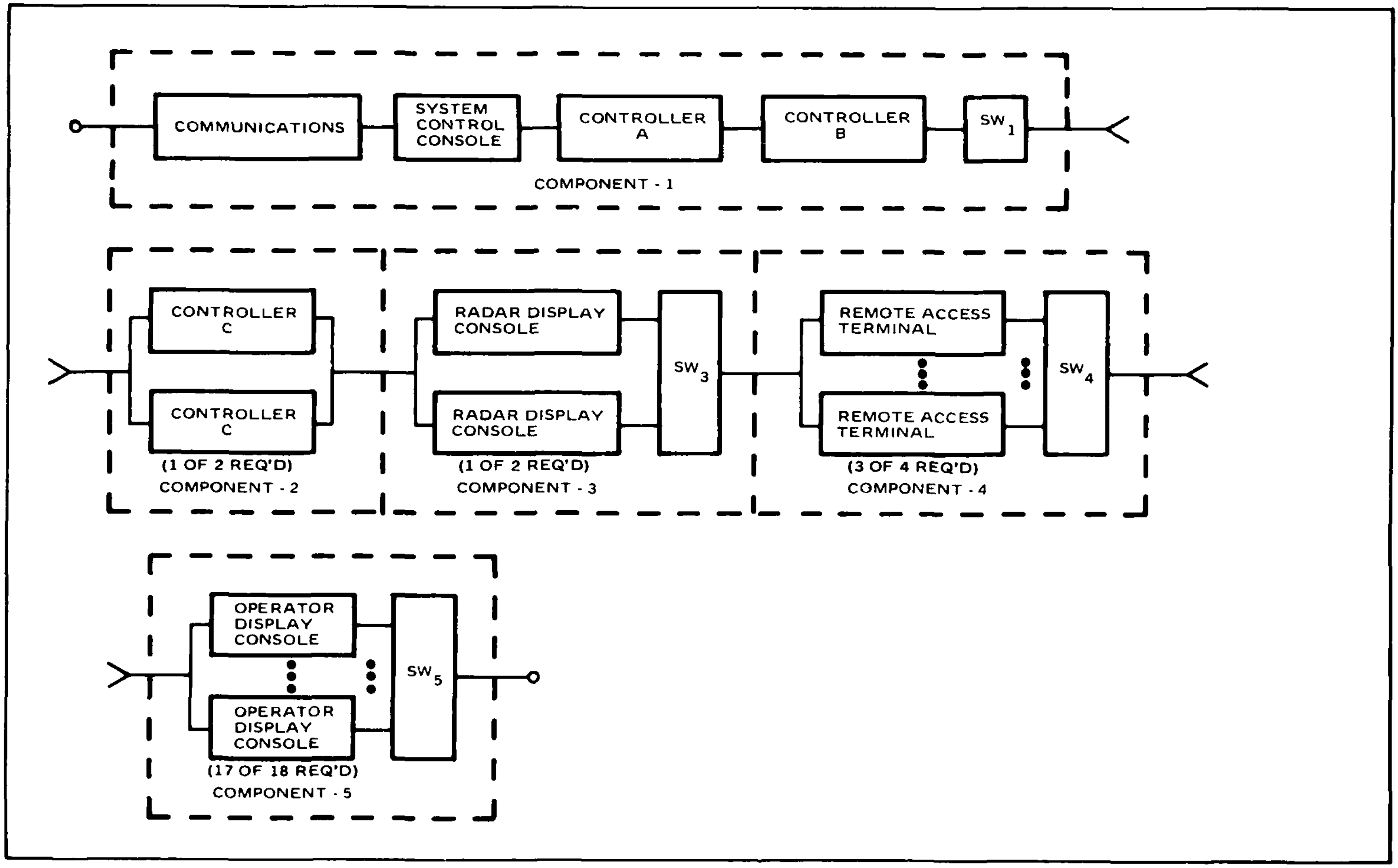


Figure 5.2-3. Reliability Block Diagram for a C^3 System

TABLE 5.2-1. EXAMPLE PARTITIONING OF AN AIR SURVEILLANCE SYSTEM SW BASED ON PERCENTAGE UTILIZATION

HW CPCI	Communications	System Central Console	Display Controller	Peripheral Controller	Central Computer	Radar Display	Remote Access Terminal	Operator Display Console	Total
OSS	0	5	5	5	73	5	2	5	100
SUS	0	20	5	5	30	10	10	20	100
APS	20	5	0	0	10	30	5	30	100
DIS	0	15	15	15	20	15	5	15	100
DRS	0	0	0	0	75	0	25	0	100
SES	0	0	0	0	35	30	5	30	100

SW partitioning is only necessary when there is interaction between HW and SW such that a HW failure changes the SW failure rate (this is discussed in more detail in Section 5.3). Otherwise, the total SW subsystem can be considered as a separate series component in the reliability block diagram. In Section 5.3, this partitioning is used to compute the necessary SW failure rates for each component as a function of the model parameters for each CPCI.

The equations for computing various system level FOM's for series-parallel configurations are given in Table 5.2-1. Most of these FOM's can be recognized as standard HW reliability measures. Accordingly, they reduce to the standard HW case when the SW is removed. Implicit in the formulation of these FOM's is the assumption of operational independence between the components (i.e., when the system is in a failed state undergoing repair the remaining units of the system remain in an operating state). Even when this assumption is incorrect, it provides a reasonable approximation for modern repairable electronic systems since the time to repair is much shorter than the time between system failures. Moreover, for large C^3 systems, a complete shut-down of all equipment does not generally happen whenever a system failure occurs. In any case, the equations given in the table provide a close (conservative) approximation to the extent that system components do not operate independently. For non repairable systems, equations 5.2.2-1 and 5.2.2-3 provide exact solutions. The interpretations of equation 5.2.2-1 through 5.2.2-5 are the same as those found in HW reliability theory (e.g., Barlow and Proschan, 1965, Kozlov and Ushakov, 1970). Equations 5.2.2-6 and 5.2.2-7 are new and are useful in measuring the unique affects of SW on system availability.

The individual terms in each equation represent the various components of the system as defined previously. For components which do not contain any SW, the computations are carried out using standard HW reliability formulae. For components which contain SW, equations have been derived using a Markov SW model which are provided in Section 4. The approximations given for equations 5.2.2-1, 5.2.2-2 and 5.2.2-3 are good when the component repair times are small relative to the component times to failure (i.e., $\tau_X << \theta_X$). Two complications make the computation of the mean-time-between-failures (MTBF) intractable: 1) the probability distribution laws of the system components can be arbitrary (because of redundancy possibilities) and 2) the SW failure rate changes in time. For SW-dominated systems, however, the mean-time-to-failure (θ) is a conservative approximation to the MTBF because of the improving SW.

TABLE 5.2-2. SYSTEM EQUATIONS FOR COMPUTING COMBINED HW/SW RELIABILITY

System Figures of Merit (FOM's)	Definition
(5.2.2-1) $\theta = \int_{o}^{\infty} \prod_{X=1}^{K} P_X(t)\, dt \cong \left(\sum_{X=1}^{K} \theta_X^{-1} \right)^{-1}$ where: K = Number of system components $\theta_X = \int_{o}^{\infty} P_X(t)\, dt$ $P_X(t)$ = Probability of failure-free operation of Component X for t hours.	Mean time to failure (MTTF)
(5.2.2-2) $\tau \cong \theta \sum_{X=1}^{K} (\tau_X / \theta_X)$ where: τ_X = Mean-time-to-repair Component X.	Mean time to system restoration
(5.2.2-3) $P(t) = \prod_{X=1}^{K} P_X(t) \cong e^{-t/\theta}$	Probability of failure-free operation for t hours.
(5.2.2-4) $A(t) = \prod_{X=1}^{K} A_X(t)$	Availability of the system at time t.
(5.2.2-5) $A_{ss} = \mathrm{Lim}_{t \to \infty} A(t)$	Steady state availability of the system
(5.2.2-6) $A^* = \min_{t} A(t);$ $\prod_{X=1}^{K} \min A_X(t) \le A^* \le \min_{X} \min_{t} A_X(t)$	Minimum availability (i.e., the minimum undershoot) See figure 4.1-3.

TABLE 5.2-2. SYSTEM EQUATIONS FOR COMPUTING COMBINED HW/SW RELIABILITY (Continued)

System Figures of Merit (FOM's)	Definition
(5.2.2-7) $\bar{A}_T = \frac{1}{T}\int_0^T A(t)dt$	Average availability for a mission duration T.

5.3 FAILURE DEFINITION

5.3.1 Derivation of Transition Rates for HW Failure

The failure rates of the units in Figure 5.2-3 (e.g., the computer, a display console, controller A, etc) are assumed to be constant (i.e., an exponential time to failure model is assumed). If they were not (i.e., contained redundancy), then another level of detail in the system model would be required in order to reach units of constant failure rate. With this assumption, therefore, these HW failure rates are derived in the normal manner based on, for example, applying MIL-HDBK-217, to part counts under specified electrical and environmental stresses. When redundant units are utilized in the system (e.g., the operator display consoles of component 5), the transition rates to the various states (e.g., operating, degraded and failed) are functions of the hardware failure rate and the current operating state of the system. Thus, if the failure rate of a single operator display console (ODC) is λ_{ODC} then the rate at which component 5 transitions from 18 consoles to 17 consoles (one console down) is $18\lambda_{ODC}$, and from 17 to 16 consoles (the failed state of component 5) is $17\lambda_{ODC}$. In the general rate of transition from state to state is dependent on the number of units and the type of redundancy (i.e., whether the standby units are fully operational, partially operational or non-operational. The HW transition rate $\lambda_{HW}(i)$ from the ith state to the i+1 state is given below for these three cases:

$$(5.3.1\text{-}1)\quad \lambda_{HW}(i) = \begin{cases} (N_H - i)\ \lambda_u,\ 0 \le i \le N_H - M, \text{ fully operational} \\ [M + (N_H - M - i)\nu]\,\lambda_u,\ 0 \le i \le N_H - M, \text{ partially operational.} \\ M\lambda_u,\ 0 \le i \le N_H - M, \text{ non-operational} \end{cases}$$

where N_H is the total number of units available; M is the number of units required for successful operation; λ_u is the failure rate of a single unit; ν represents the fraction of λ_u at which the standby units are operating; and i is the number of HW units that are down.

In equation (5.3.1-1), i = 0 represents the full-up state for the component and i = N_H-M+1 represents the failed state. The values for $\lambda_{HW}(i)$ provide the state-by-state failure transitions from full-up to failure used in the reliability constructs of system components (see Section 4.0).

5.3.2 Derivation of SW Transition Rates for Failure

The following position is taken on the meaning of a SW "failure:"

1) Failure occurs as a result of a latent SW fault which manifests itself by causing one or more units in the system to malfunction, or by transmitting erroneous information to a user.

2) Failure occurrences are random in the sense that SW is executed during the conduct of a mission in a random manner. Thus, the external stimuli of an operational environment (e.g., target identification, tracking information) is considered random phenomena, and SW is exercised by this stimuli via the peripheral HW units.

SW fault causes stem from various error sources (i.e., logic, data definition, data handling, interface, computational, etc). Fault manifestations also have varying effects on mission performance, such as critical, major, or minor (nuisance). Therefore, not all fault manifestation result in a system failure. If information is collected on SW error data for purpose of estimating the SW failure rate parameters (discussed below), only error data classified as failure-causing (e.g., critical or major) should be used.

The SW failure rate model assumed for application to specific reliability constructs in Section 4.0 is a Poisson type. Other SW models are, of course, possible and the methodology is not affected by another selection other than for parameter changes. A basic parameter of most of the current SW models is the initial number of faults (or bugs), N_o. Using various estimating techniques, the value of N_o can be determined from error historical data. Attempts have also been made to estimate N_o based on a number of SW complexity metrics (e.g., cyclomatic number, module fan-in and fan-out, syntactical constructs and manpower) using descriptive types of information (Fitzsimmons, 1978; Schneider, 1980; Winchester, 1978; et al). The baseline SW failure rate (λ_{SW}) has the following form:

$$\lambda_{SW}(j) = \phi(N_o - j) \qquad (5.3.2\text{-}1)$$

where N_o is defined above and ϕ is a constant of proportionality (representing the failure rate of a single fault) also determined from error data. Thus, $\lambda_{SW}(j)$ is proportional to the number of bugs remaining in the SW after j bugs have been corrected. If we only consider the number of bugs remaining in the SW, say N, then 5.3.2-1 can be written:

$$\lambda_{SW}(N) = \phi N. \qquad (5.3.2\text{-}2)$$

The SW failure rate is partitioned according to how the various program sets (e.g., APS, OSS and SUS) are exercised by the system HW units. For example, using the SW partition illustrated in Table 5.2-1 the failure rate of the operator display console component (λ_{SW_5}) is:

$$\lambda_{SW_5} = 0.05\, N_{OSS}\, \phi_{OSS} + 0.2\, N_{SUS}\, \phi_{SUS} + 0.3\, N_{APS}\, \phi_{APS}$$
$$+ 0.15\, N_{DIS}\, \phi_{DIS} + 0.3\, N_{SES}\, \phi_{SES}$$

where N_X and ϕ_X parameters are estimated from error data collected on the individual CPCI's and the table values represent the fraction of the total time each CPCI is exercised by the operator display console (Component 5).

In addition to the partitioning of SW across system components, the SW "duty cycle" within a system component must be considered. For example, if the error data on CPCI X used to estimate N_X and ϕ_X is collected under test conditions, then ϕX must be adjusted to represent field operating conditions. Similarly, SW dependence on the state of the HW should also be considered. This dependency is unique to each application and can be very complex. For series components, the SW can always be modeled as an independent series box as in Section 4.1. For system components which contain redundant HW units, two extreme situations can be modeled which will provide bounds for most cases are: 1) failure of a HW unit within the component does not effect the SW (i.e., SW loading remains constant) so that the SW could be treated as a separate "unit" in series with the HW, and 2) failure of a HW unit reduces the loading on the SW resulting in a proportionate decrease in the SW failure rate.* Thus, if the values of the SW parameters N and ϕ are based on the loading intensity of a single HW unit within a component containing N_H units (of which M are required for successful operation), then the SW transition rates for the i^th^ system state (i.e., i HW units are in repair) are given by:

$$(5.3.2\text{-}3) \quad \underset{SW}{\lambda(N,i)} = \begin{cases} \phi_X N_X (N_H - i),\ 0 < i < N_H - M, \text{ HW state dependent (operating standby)} \\ \phi_X N_X, \text{ HW state independent or a series system.} \end{cases}$$

where N_H-M represents the total number of redundant standby units. HW state dependent transition rates for partial standby units and non-operating standby units are similarly applicable as defined in equation (5.3.1-1).

5.4 QUANTIFIABLE MAINTAINABILITY CONCEPTS

5.4.1 Derivation of Transition Rates for HW Repair

The time to repair distribution is assumed to be exponential. This assumption is necessary for tractability. However, in cases where the time-to-failure is much larger than the time to repair (typical of modern repairable electronic systems), this assumption approximates the case where the time-to-failure distribution is exponential and the time to repair distribution is arbitrary (Kozlov and Ushakov, 1970).

* (An example of this is a functionally redundant unit)

The expected repair time of individual HW units is generally based on prediction data, built-in test capability and maintainability concepts unique to the HW usage. The HW transition rate from a state of i HW units down to the state with i-1 HW units down is dependent on the number of units in repair and the number of repairmen servicing these units. Bounds are obtained by the two extremes:

$$(5.4.1\text{-}1) \quad \mu(i)_{HW} = \begin{cases} i\,\mu_{UHW}, & 1 \leq i < N_H\text{-}M, \text{ unlimited repairmen.} \\ \mu_{UHW}, & 1 \leq i \leq N_H - M, \text{ single repairman.} \end{cases}$$

where μ_{UHW} is the estimated unit repair rate and N_H-M represents, as before, the number of standby units. Thus, in the unlimited repairmen case every failed unit is assigned a repairman working at a rate of μ_{UHW}, and, in the single repairman case the rate is constant regardless of the number of units failed. Most of the actual repair situations for C^3 systems will fall within these two extremes.

5.4.2 Derivation of Transition Rates for SW Repair

5.4.2.1 Introduction

Although there are similarities between SW and hardware maintainability (such as fault isolation), the term maintainability as applied to SW must be used carefully. A synonym, repairability is the ability to restore something to its state before failure. In a strict sense, SW is not repaired, because when SW fails it is corrected. SW maintainability entails the ability to change a computer program and/or data to a new state, and does not entail the ability to restore to a previous state. Therefore, when the term "repair" is used in connection with SW, it should always be taken in this later context.

Of the three classes of SW maintenance: 1) corrective, 2) adaptive, and 3) perfective, only corrective is applicable to the quantitative assessment of a software failure's contribution to system availability and operational readiness. The other two classes can be considered applicable only in the sense that such maintenance activities can introduce faults which produce failures that in turn require corrective maintenance.

5.4.2.2 General SW Maintainability Considerations

The scope of this task was to investigate and develop SW maintainability concepts that would in turn lead to the development of a combined HW/SW system availability measure. The investigation started with a literature search. Several proposed hierarchies of SW maintainability attributes were found. However, care must be taken in determining those attributes that directly influence SW maintainability. As Feuer and Fowlkes (1979) point out: "identifying program characteristics that may coincidently vary with a fundamental property tells us little about the property." The approach to selecting quantifiable SW maintainability attributes was to use the McCall and Matsumoto (1980) hierarchy (see Figure 5.4.2-2) as a starting point. The SW maintainability criteria were scrutinized for associated metrics that were coercible.

Some attributes, such as Quantity of Comments were eliminated to develop a final exhaustive set of SW maintainability attributes and metrics. Instead a set was compiled that includes attributes directly representative of SW maintainability characteristics, and associated criteria that are easy to measure and convert into SW transition rates for correction.

Some (such as Yau, 1980) contend that SW maintainability cannot be predicted, but that it can be measured as the SW proceeds through the development phases. Yau views maintainability as including the resistance to both logical and performance ripple effects, and has developed an algorithm for measuring the resistance (or stability) based on the number of logical and performance attributes changed. Curtis, et al. (1979), report on experiments where metrics applied during the coding phase did predict the difficulty in performing maintenance activities.

During the investigation of SW maintainability concepts, Hughes has concentrated on those activities that have associated quantifiable factors. The factors of interest are those that can be included in system metrics (e.g., mean-time-to-repair) or contribute to system-level figures of merit for design attribute tradeoffs.

5.4.2.3 Consideration of SW Maintainability Techniques in Modeling System Restoration

In developing metrics that model restoration times during mission operation, one must consider all known SW maintenance techniques. Certain quick-fix techniques such as patching - although not condoned as a good configuration control practice - are certainly effective in restoring a critical C^3 system function during operation. Table 5.4.2-1 summarizes common software maintenance techniques, and indicates the general effect on system availability and operational capability.

Startover and switchover are common techniques for attempting to restore a system, especially in environments where maintenance facilities are not collocated. The risk with this technique - although causing minimal downtime as listed in Table 5.4.2-1 - is that the software fault may still remain in a subtle form or that the restart does not completely resynchronize the system or restore critical data. A more effective method for startover is to collect data that will support troubleshooting, prior to restarting. This method increases the probability of correcting a fault that otherwise may recur during a critical mission.

If startover and switchover capabilities are not provided or unsuccessful then a reload and initialization of the software subsystem is usually performed. It is at this point, before the reload, that extra time should be taken to execute selected on line fault isolation and generate memory printouts for off line troubleshooting. A typical reload can be completed in four minutes while an initialization can take five to ten minutes depending on the interface configuration.

TABLE 5.4.2-1. IMPACT OF SOFTWARE MAINTAINABILITY TECHNIQUES ON SYSTEM RESTORATION

Restoration Technique	System Availability	Operational Capability
• Automatic Startover	Downtime (<1 min.)	No degradation
• Automatic Switchover	Downtime (<2 min.)	No degradation
• Semiautomatic Startover	Downtime (<3 min.)	Suspect if no safe data
• Manual Reload	Reload downtime (<5 min.)	No degradation
• Fallback to previous version/release	Reload downtime (<5 min.)	Lacks latest revisions
• Patch	Unavailable during problem analysis and patch implementation and test (<13 hr.)	No degradation
• Selective Recompilation/Reassembly	Possible unavailability during problem analysis, source correction, recompilation, test and reload (<15 hr.)	No degradation
• Complete Recompilation/Reassembly	Unavailable during problem analysis, source correction, recompilation, system generation, test, and reload (<17 hr.)	No degradation
• Load new application program	Reload downtime (<5 min.)	No degradation

Patching is the correction of a program in machine code, regardless of the language in which the program was originally coded. In most systems patches may be entered into the system through the computer console. After a patch is implemented it may be tested online by using the patching technique to vary input data and to force certain path executions. A successful simple patch can be implemented in approximately one hour (see Table 5.4.2-4).

Recompilation, reassembly, and link-loading often occur on a "host" computer other than the one on which the program is to be run. This is done to take advantage of support facilities which large computing systems possess and small, typically real-time, computers do not. Large computing systems often have prohibitive turnaround time, however most C^3 systems provide for dedicated support computers which are collocated with the operational configuration, in order to reduce the recompilation turnaround time. A recompilation correction on an unsupported computer could cause downtime from two and a half to seventeen hours depending on the complexity of the fault.

For completeness, the related topics of configuration management and interactive versus batch processing should be mentioned since these areas impact the SW fault correction process. Although patching is the recommended technique for a quick fix in an operational environment, it can cause software configuration control problems, and thus slow down fault corrections during subsequent source-level maintenance because the octal patch form of the correction usually does not directly correspond to the source form. Octal is a commonly used three-bit pattern that is convenient for machines with a basic word size that is divisible by three. Examples of problem causes are: failure to document the correction in patch form; and the introduction of faults during the conversion of patch to source corrections. Interactive maintenance is becoming increasingly popular during SW development. For example, the Source Code Control System of the Programmers Workbench provides responsive change and selective recompile capabilities under a protective mechanism against unauthorized users. Such systems - although designed for timesharing SW development - could be adapted to C^3 operations by providing an emergency dedicated mode that could be invoked from a remote site. The resulting computer program could be automatically transmitted to the site after being ιested at the support location via the remote terminal. Such a technique with its attendant reliability and configuration management might be competitive in responsiveness with the on-site patching technique.

5.4.2.4 Selection of Quantifiable Attributes

A prerequisite for measuring SW maintainability is the determination of those attributes that characterize maintainability. A search of the literature revealed several candidate sets of attributes. Boehm (1973, et al., decomposed maintainability into three attributes and nine subattributes as shown in Figure 5.4.2-1.

McCall and Matsumoto (1980) decompose maintainability into five criteria as shown in Figure 5.4.2-2, and then into eleven metrics. Yau proposes stability as the most critical SW maintainability factor where stability is further decomposed into functional and performance subfactors. Most of the candidate SW maintainability factors can be further defined in terms of quantifiable subfactors that can be either measured or estimated during early phases of SW development.

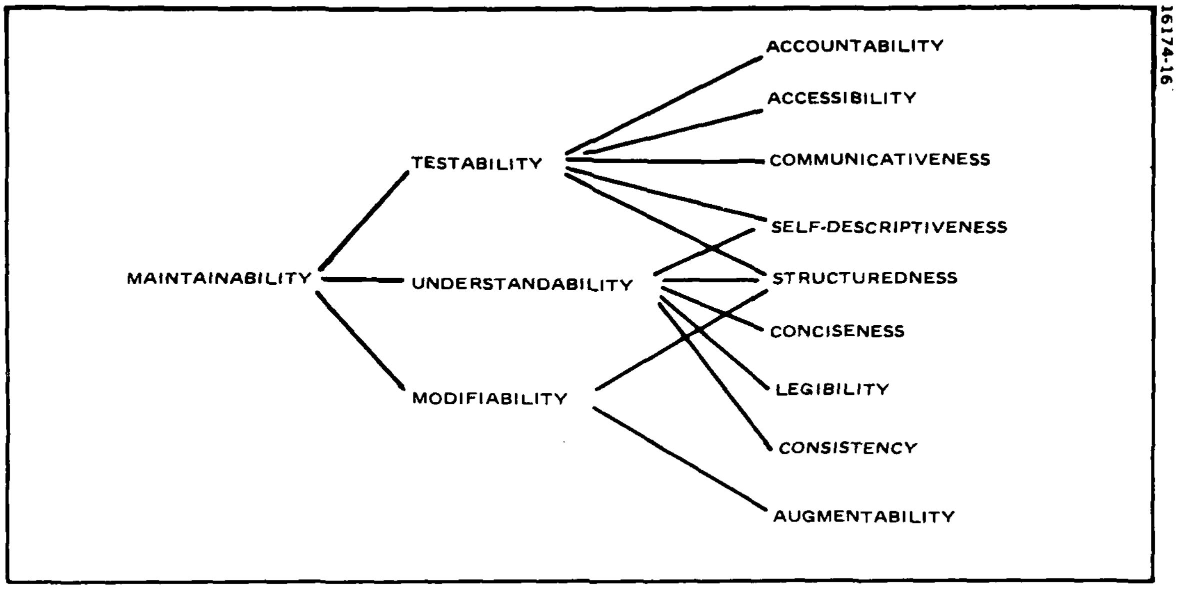


Figure 5.4.2-1. Boehm's SW Maintainability Tree

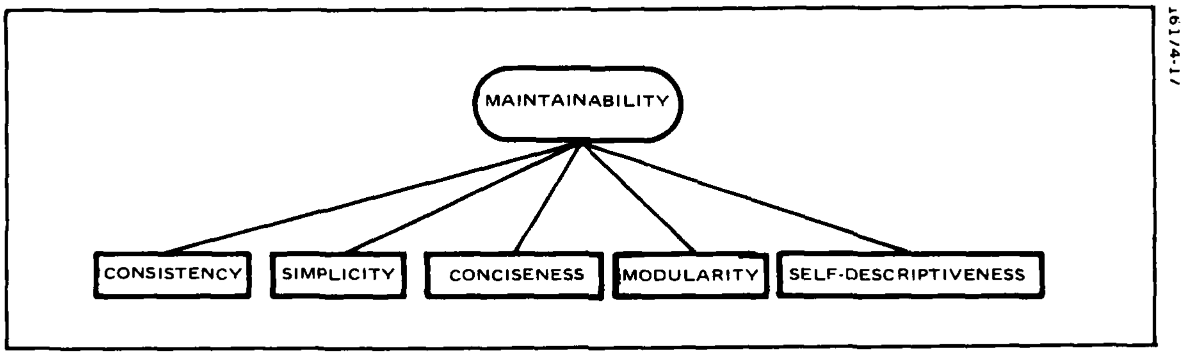


Figure 5.4.2-2. McCall-Matsumoto SW Maintainability Tree

Hughes suggests the decomposition of SW maintainability into three attributes, as shown in Figure 5.4.2-3, that are representative of the time-consuming activities of problem analysis and resolution. Self-descriptiveness is an attribute of software that characterizes the extent to which the source code (including comments and prologue) of a computer program contains enough information for a software engineer to maintain it. Complexity is an attribute of software that characterizes the degree of difficulty a software engineer may encounter -- in terms of syntactical structures, control and data flow, module interconnections, and entropy -- when tasked to modify an existing computer program. Modularity is an attribute of software that characterizes the extent to which a portion (module) of a computer program can be modified without affecting other portions of the program. The associated factors are listed in Table 5.4.2-3. They were tailored from the GE/RADC set (McCall, 1980, Vol. II) based on later findings by GE/RADC (McCall 1980, Vol. I) and on the emphasis placed on the complexity attribute by Hughes. The tailoring scheme is explained in Table 5.4.2-2.

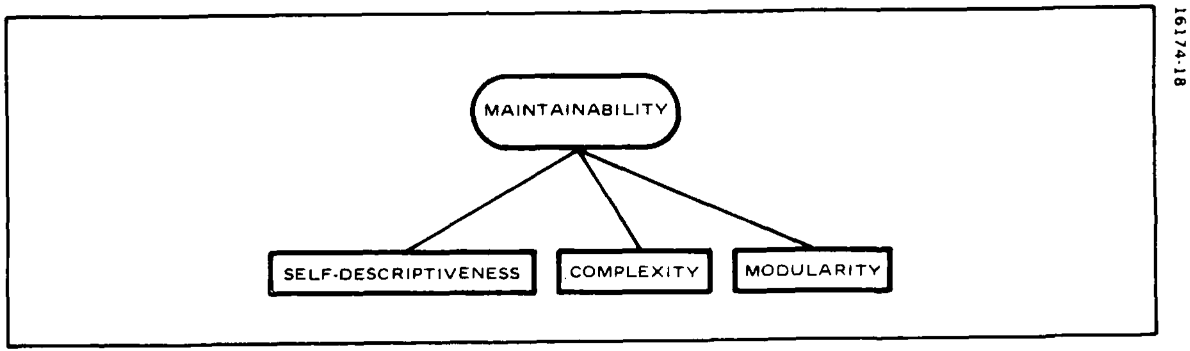


Figure 5.4.2-3. Hughes-Recommended SW Maintainability Tree

TABLE 5.4.2-2. TAILORING SCHEME FOR MAINTAINABILITY FACTORS

GE/RADC Factors*	Action	Justification
Consistency Checklists (CS.1.2)**	Deleted	Not predictive
Design Structure Measure (SI.1)	Retainod	Designated CO.1
Structure Language Check (SI.2)	Deleted	Not generally applicable***
Complexity Measure (SI.3)	Modified	Redefined and designated CO.2
Coding Simplicity Measure (SI.4)	Retained	Designated CO.3
Stability Measure (MO.1)	Deleted	Too difficult to measure***
Modular Implementation Measure (MO.2)	Retained	--
Quantity of Comments (SD.1)	Deleted	Not sensitive enough
Effectiveness of Comments Measure (SD.2)	Retained	--
Descriptiveness of Implementation Language Measure (SD.3)	Retained	--
Conciseness Measure (CO.1)	Deleted	Too difficult to calculate

*McCall (1980, Vol. II)
**Abbreviation references in parentheses relate to definitions in McCall (1980, Vol. I)
***McCall (1980, Vol. I)

TABLE 5.4.2-3. SUGGESTED MAINTAINABILITY FACTORS

- Design Structure (CO.1)
- Use of Structured Language (CO.2)
- Data and Control Flow Complexity (CO.3)*
- Logical Complexity (CO.4)*
- Effectiveness of Comments (SD.2)
- Descriptiveness of Implementation Language (SD.3)
- Modular Implementation (MO.2)

*Measured at intramodule level (see Table 5.4.2-7).

5.4.2.5 System Compatible SW Maintainability Metrics

The system downtime resulting from a SW failure is dependent on the restoration technique called for (see Table 5.4.2-1) and the maintainability characteristics of the SW. If the system can be restored by startover or switchover techniques, the system downtime is not significant (usually less than 5 minutes). If, however, the SW requires some design correction (i.e., a SW patch or recompile/reassemble) for system restoration, the system downtime can be very significant. The amount of downtime in this case is greatly dependent on the maintainability characteristics of the SW.

The mean system restoration time, τ''_{SW}, due to a SW failure is defined by:

$$\tau_{SW} = \alpha_1 \tau'_{SW} + \alpha_2 \tau''_{SW} + \alpha_3 \tau'''_{SW} \qquad (5.4.2\text{-}1)$$

where:

α_1, α_2, α_3 = relative frequencies of occurrence of startover/switchover, patch, and recompile/reassemble, respectively.

τ'_{SW} = average time for startover/switchover.

τ''_{SW} = average time for patch.

τ'''_{SW} = average time for recompile/reassemble.

Table 5.4.2-4 provides some typical values for τ'_{SW} applicable to C^3 systems. These recovery times are based primarily on the systems operating configuration and are independent of the maintainability characteristics of the SW.

Values for τ''_{SW} and τ'''_{SW} are based on characteristics of SW maintainability. A promising approach to developing a quantitative, predictive SW maintainability measure is derived from the suggested maintainability factors of Table 5.4.2-3. These maintainability factors are aggregated at the CPCI or functional level. Then the resulting CPCI values for the entire SW subsystem are averaged (weighted by probability of execution) to arrive at a system level SW maintainability figure of merit (FOM). This FOM is then translated to a SW correction time based on the restoration technique employed for τ''_{SW} and τ'''_{SW}.

The SW maintainability FOM, δ_M, for an aggregate of n CPCI's is defined by:

$$\delta_m = \sum_{K=1}^{n} F_{T_K} \cdot p(E_K) \qquad (5.4.2\text{-}2)$$

where F_{T_K} is the combined maintainability measure for the K^th CPCI and $p(E_K)$ is the relative frequency of CPCI execution during a typical or average mission scenario. The maintainability measure is averaged across CPCI and module level factors as follows:

$$F_{T_K} = \frac{1}{2}\left[\frac{1}{m_K}\left(\sum_{j=1}^{m_K} F_{m_{jK}}\right) + F_{c_K}\right] \qquad (5.4.2\text{-}3)$$

where:

$F_{M_{jK}}$ = $1/2\ (CO_2 + CO_3)$

F_{c_K} = $1/3\ (SD_3 + CO_1 + MO_2)$

m_K = Number of modules in the K^{th} CPCI

CO_1, CO_2, CO_3, SD_3 and MO_2 = **Maintainability Factors unique to the jth module in the kth CPCI (see Table 5.4.2-3)**

Finally, the FOM given in (5.4.2-2) can be converted to SW correction times, τ''_{SW} and τ'''_{SW}, by scaling values of δ_M to individual SW activities according to the correction method based on user experience. Tables 5.4.2-5 and 5.4.2-6 provide ranges of values for these activities based on actual SW maintenance experience on C^3 projects developed at Hughes. The term "regression testing" listed in the table is an activity involving both methods of SW correction which is concerned with assuring that satisfactory SW performance already attained and tested is not perturbed by implementation of the correction. The following relationships convert δ_M values to SW correction time based on the correction method:

$$\tau''_{SW} = 14.5\text{-}13\ \delta_M \qquad \text{(for patch correction)} \qquad (5.4.2\text{-}4)$$

$$\tau'''_{SW} = 19\text{-}16.6\ \delta_M \qquad \text{(for recompile/reassemble)} \qquad (5.4.2\text{-}5)$$

The relative frequencies α_1, α_2 and α_3 in (5.4.2-1) are determined from the system operating configuration and user experience. Typically, for C^3 systems developed at Hughes these values are 0.6, 0.3 and 0.1, respectively. The corresponding transition rate for system restoration due to a SW failure is defined in terms of τ_{SW}:

$$\mu_{SW} = \tau^{-1}_{SW} \qquad (5.4.2\text{-}6)$$

It must be emphasized that the relationships of (5.4.2-4) and (5.4.2-5) were derived from Hughes experience and may be different for each SW contractor. It should also be noted that the method of combining module-level metric values with CPCI-level metric values deserves further study. Some researchers propose a percentile form rather than the normalized form used here (refer to Table 5.4.2-7), while others propose multiplying module-level values rather than summing. It was not within the scope of this study to investigate the accuracy of alternate methods of combining metrics. Consequently the straightforward methods of normalization and summation were selected.

TABLE 5.4.2-4. TYPICAL C^3 SYSTEM RECOVERY TIME AFTER SOFTWARE FAILURE

Technique	Recovery Time Range (min)
Automatic startover	0.5 - 1.0
Automatic switchover	1.0 - 2.0
Semiautomatic startover	2.0 - 3.0
Manual reload	3.0 - 5.0
Reinitialization (including course and fine synchronization)	5.0 - 10.0
Average	2.3 - 4.2

TABLE 5.4.2-5. TYPICAL C^3 SYSTEM SW CORRECTION TIME FOR PATCH METHOD

Activity	Correction Time Range (hr)
Locate software fault	0.5 – 8.0
Design correction	0.25 – 2.0
Implementation correction in machine representation form	
● Find unused storage area	0.08 – 0.25
● Select instruction to patch	0.08 – 0.25
● Implement patch	0.08 – 0.5
Test correction	0.25 – 1.0
Regression test	0.25 – 1.0
Total	1.5 – 13.0

TABLE 5.4.2-6. TYPICAL C^3 SYSTEM SW CORRECTION TIME FOR RECOMPILE/REASSEMBLY METHOD

Activity	Correction Time Range (hr)
Locate software fault	0.5 - 8.0
Design correction	0.5 - 4.0
Implement correction in source form	0.5 - 2.0
Recompile/Reassemble	0.1 - 0.5
Test correction	0.25 - 0.5
Integrate into new version	0.25 - 1.0
Regression test	0.25 - 1.0
Total	2.4 - 17.0

TABLE 5.4.2-7. DESCRIPTION OF SW MAINTAINABILITY METRICS

Metric	Applicable Phase	
	Design	Coding
SD.2 EFFECTIVENESS OF COMMENTS		
(1) Modules have standard formated prologue		X
$1 - \frac{\text{\# modules that violate rule}}{\text{total \# modules}}$		
(2) Comments set off from code in uniform manner		X
$1 - \frac{\text{\# modules that violate rule}}{\text{total \# modules}}$		
(3) All transfers of control and destinations commented		X
$1 - \frac{\text{\# modules that violate rule}}{\text{total \# modules}}$		
(4) All machine dependent code commented		X
$1 - \frac{\text{\# modules that violate rule}}{\text{total \# modules}}$		
(5) All non-standard HOL statements commented		X
$1 - \frac{\text{\# modules that violate rule}}{\text{total \# modules}}$		
(6) Attributes of all declared variables commented		X
$1 - \frac{\text{\# modules that violate rule}}{\text{total \# modules}}$		
(7) Comments do not just repeat operation described in language		X
$1 - \frac{\text{\# modules that violate rule}}{\text{total \# modules}}$		
SD.3 DESCRIPTIVENESS OF IMPLEMENTATION LANGUAGE		
(1) High order language used		X
$1 - \frac{\text{\# modules with direct code}}{\text{total \# modules}}$		
(2) Variable names (mnemonic) descriptive of physical or functional property represented		X
$1 - \frac{\text{\# modules that violate rule}}{\text{total \# modules}}$		
(3) Source code logically blocked and indented		X
$1 - \frac{\text{\# modules that violate rule}}{\text{total \# modules}}$		

TABLE 5.4.2-7. DESCRIPTION OF SW MAINTAINABILITY METRICS (Continued)

Metric	Applicable Phase	
	Design	Coding
(4) One statement per line		X
$1 - \frac{\#\text{ continuations + multiple statement lines}}{\text{total }\#\text{ lines}}$		
CO.1 DESIGN STRUCTURE		
(1) Design organized in top down fashion	X	X
(2) Module processing not dependent on prior processing	X	
(3) Modules have single entrance, single exit		X
$\left(\frac{1}{\#\text{ entrances}} + \frac{1}{\#\text{ exits}}\right)$		
(4) Compartmentalization of data base	X	
$\left(\frac{\text{size}}{\#\text{ files}}\right)$		
CO.2 DATA AND CONTROL FLOW COMPLEXITY		
(1) Module Size Profile		X
(2) Cyclomatic Number, $V_{(G)}$	X	X
(3) Variable Liveness		
$\left(\frac{\#\text{ live variables}}{\#\text{ possible live variables}}\right)$		X
(4) Logical Stability (Yau 1980)		X
(5) Module flow top to bottom		X
CO.3 LOGICAL COMPLEXITY		
(1) Negative Boolean or complicated compound Boolean expressions used		X
$\left(1 - \frac{\#\text{ of above}}{\#\text{ executable statements}}\right)$		
(2) Jumps in and out of loops		X
$\left(\frac{\#\text{ single entry/single exit loops}}{\text{total }\#\text{ loops}}\right)$		
(3) Loop index modified		X
$\left(1 - \frac{\#\text{ loop indices modified}}{\text{total }\#\text{ loops}}\right)$		
(4) Module is not self-modifying		X
$\frac{\#\text{ constructs}}{\text{total lines of code}}$		

TABLE 5.4.2-7. DESCRIPTION OF MAINTAINABILITY METRICS (Continued)

Metric	Applicable Phase	
	Design	Coding
(5) Number of statement labels $\left(1 - \frac{\#\text{ labels}}{\#\text{ executable statements}}\right)$		X
(6) Nesting level $\left(\frac{1}{\text{max nesting level}}\right)$		X
(7) Number of branches $\left(1 - \frac{\#\text{ branches}}{\#\text{ executable statements}}\right)$		X
(8) Number of GOTOs $\left(1 - \frac{\#\text{ GOTO statements}}{\#\text{ executable statements}}\right)$		X
(9) Variable mix in a module $\left(\frac{\#\text{ internal variables}}{\text{total }\#\text{ variables}}\right)$		X
(10) Variable density $\left(1 - \frac{\#\text{ variables}}{\#\text{ executable statements}}\right)$		X
MO.2 MODULAR IMPLEMENTATION		
(1) Hierarchical structure $\left(1 - \frac{\#\text{ violations of hierarchy}}{\text{total }\#\text{ modules}}\right)$	X	X
(2) Controlling parameters defined by calling module $\frac{\#\text{ control variables}}{\#\text{ calling parameters}}$	X	X
(3) Input data controlled by calling module	X	X
(4) Output data provided to calling module	X	X
(5) Control returned to calling module	X	X
(6) Modules do not share temporary storage	X	X

Section 6.0

RELIABILITY TRADEOFF METHODOLOGY

6.1 GENERAL APPROACH

The general HW/SW reliability tradeoff problem can be described as the selection of the appropriate mix of hardware and software complexity to achieve desired reliability characteristics. Such tradeoff studies are of course, restricted to mission tasks/functions which can be accomplished using either HW or SW or mixtures of both (such functions are, for example, graphic display generation functions and peripheral controllers).

Other tradeoff studies are possible. For example, increasing BIT capability by increasing SW complexity will reduce MTTR and hence, increase availability thus allowing the possibility of reducing HW complexity (e.g. eliminate some redundancy) while still maintaining adequate availability. This is not a direct tradeoff of HW and SW complexity, but rather a tradeoff between MTBF and MTTR. Such tradeoffs have been analyzed before but not from the standpoint of a model which adequately combines both HW and SW failures. The primary interest in this section, however, is in the tradeoff between HW and SW complexity for the purpose of performing specific mission tasks.

To perform a given tradeoff analysis it is necessary to fix the basic system configuration (e.g. with respect to HW redundancy) as it applies to mission tasks not related to the tradeoff task. Also, as seen earlier, steady state availability is not an adequate figure of merit for evaluating performance of systems possessing both HW and SW failures because of the possible undershoot of availability below steady state. A better measure is the minimum availability

$$A^* = \min_{t \geq 0} A(t) \qquad 6.1\text{-}1$$

where A(t) is the system availability as a function of time (c.f. 3.3-1). An alternate measure which can be us̲ed if a particular mission length T is important is the average availability over the mission length defined by

$$\bar{A}_T = T^{-1} \int_0^T A(t)\, dt. \qquad 6.1\text{-}2$$

If T is large, however, the average availability measure will simply reduce to the steady state availability. With all other quantities fixed, then the tradeoff can be performed by computing A* or $\bar{A}_T$ for varying combinations of HW and SW or, if there is a fixed availability requirement, an isometric curve relating HW complexity to SW complexity to achieve the fixed requirement can be generated.

6.2 HW COMPLEXITY

HW complexity is defined in terms of total HW failure rate usually derived from piecepart counts along with generic classification and use of, for example, MIL-HDBK-217. To perform a tradeoff analysis, it is necessary to fix all HW failure rates which are not involved in the mission task for which HW and SW are being traded. For further discussion on the development of HW failure rates, see Section 5.3.1.

6.3 SW COMPLEXITY

For the baseline model the SW failure rate is directly proportional to the number of faults remaining in the SW. Since the number of faults is changing randomly in time it does not make sense to use SW failure rate in the same sense as HW failure rate. However, assuming the constant of proportionality ϕ to be fixed, the SW failure rate when the system is delivered is ϕN where N is the number of faults in the SW upon delivery. Thus, HW failure rate can be traded against the <u>initial</u> SW failure rate ϕN. It must be emphasized here that ϕ is fixed and SW failure rate decreases in units of ϕ as faults are corrected.

The constant ϕ can be found either from historical SW debugging data (e.g., estimated as in Schafer, et. al. 1979) or known from projects employing similar SW. The value of N has been shown to be related to SW complexity metrics referred to in Section 5.3.2.

The SW fault correction rate $c = \lambda p$ must also be fixed for the tradeoff analysis since it is based on the user's SW support plans. The value of λ is based on the maintenance philosophy and intensity with which effort is made in troubleshooting and correcting faults. The value of p, the probability of a successful debug, can be estimated from historical SW debugging data which is dependent on factors such as the skill-level of the user's SW maintenance personnel and SW complexity metrics.

6.4 TRADEOFF PROCEDURE

The first step in performing a HW/SW tradeoff analysis is to define the system structure in terms of HW/SW reliability configuration, maintenance policies, etc. The mission tasks for which HW and SW are to be traded should be identified at this stage.

The next step is to estimate the static parameters needed to complete the model such as ϕ, λ, p, HW/SW repair rates, and HW failure rates. The range of interest (and/or feasibility) for the initial number of SW faults, N, and λ_H, the failure rate associated with the HW portion of the tradeoff, should also be defined.

Finally, using the baseline model A* (or $\bar{A}_T$) can be computed for values of (λ_H, ϕ_N) in the specified range. Alternately, A* (or $\bar{A}_T$) can be fixed at a required level and values (λ_H, ϕ_N) can be found (by exercising the model) which yield the specified A* (or $\bar{A}_T$) and an isometric curve can be generated. The latter approach is useful both as a design tool and as a tool to be used in selecting HW/SW complexity requirements (see Figure 6.4-1).

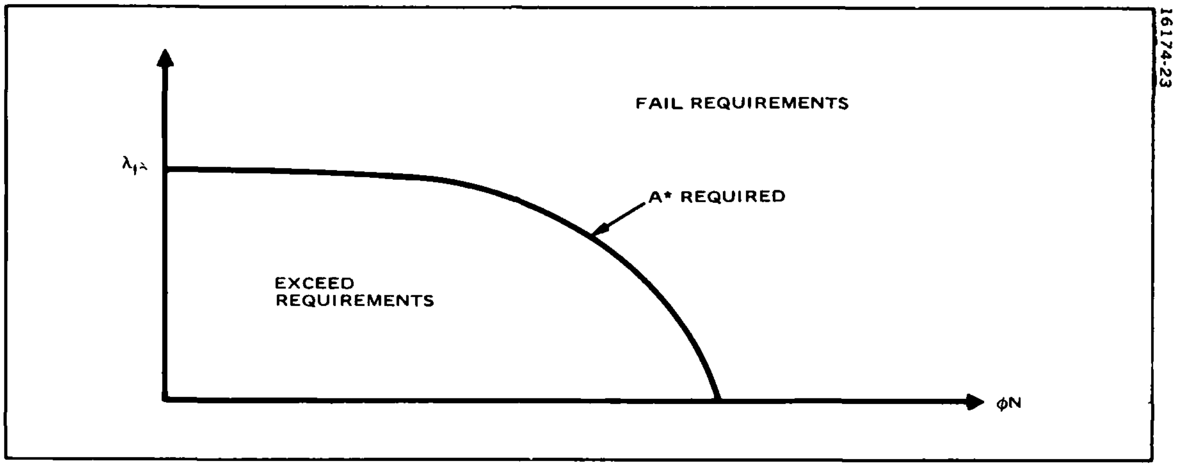


Figure 6.4-1. Isometric Curve of Constant A* as a Design Guide

6.5 EXAMPLES

6.5.1 Series Constructs

In the example of Section 4.1-2, there were three states: Full-up (HS); HW up, SW down ($H\bar{S}$); HW down, SW up ($\bar{H}S$). For this example, $\phi = 0.01$, $c = \lambda p = 0.1$, and $\mu_S = \mu_H = 2$. The values of ϕ_N and λ_H were varied to achieve A* (approximated using the program in Appendix F) equal to 0.97. Figure 6.5-1 shows the resulting isometric curve. In generating Figure 6.5-1, the computer program was run for N = 1, 2, ..., 10 SW bugs, initially, and varying λ_H until the minimum value of A(t), t = 1, 2, ... (= probability of being in state "0" since this is the only operational state) reached 0.97 to 3 decimal places. A similar curve could be generated for $\bar{A}_T$ by running the computer program as before but using:

$$\bar{A}_T \approx \frac{1}{T+1} \sum_{t=0}^{T} A(t)$$

as an approximation to (6.1-2).

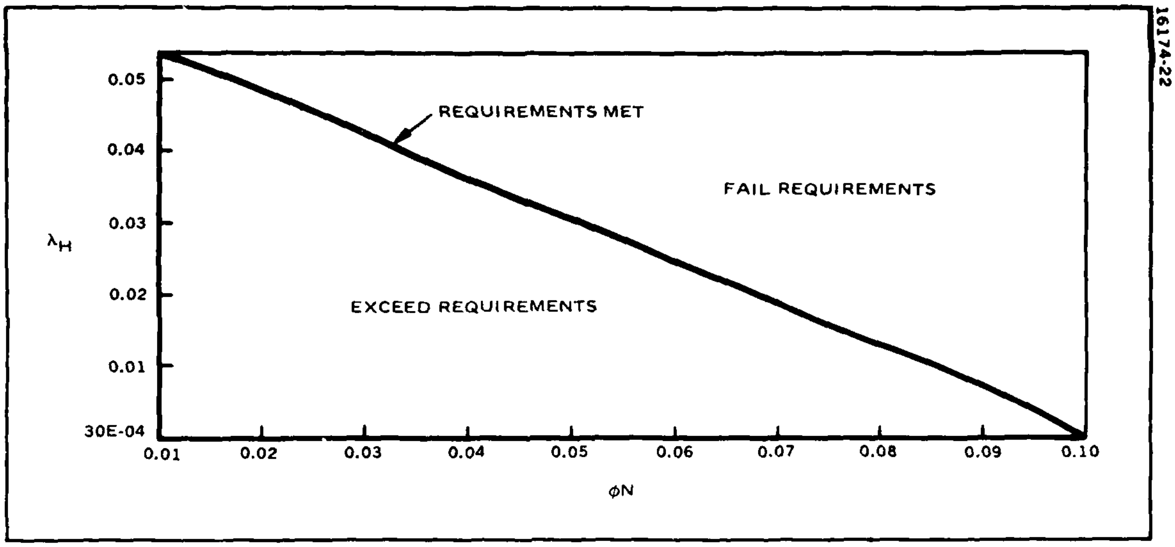


Figure 6.5-1. Isometric Curve of A* = 0.97 for the Series Example

6.5.2 Redundant HW Constructs: Five State

Refer to Section 4.2-2, and the five-state case. In this example, we took $M = 9$, $N_H = 10$, $c = \lambda_p = 0.70$, $\mu_s = 1$, $\mu_H = 2$, and $\phi = 0.001$ and varied λ_H and ϕ_N to achieve $A^* = 0.99$. Figure 6.5-2 shows the resulting isometric curve generated with the help of the computer program of Appendix F.

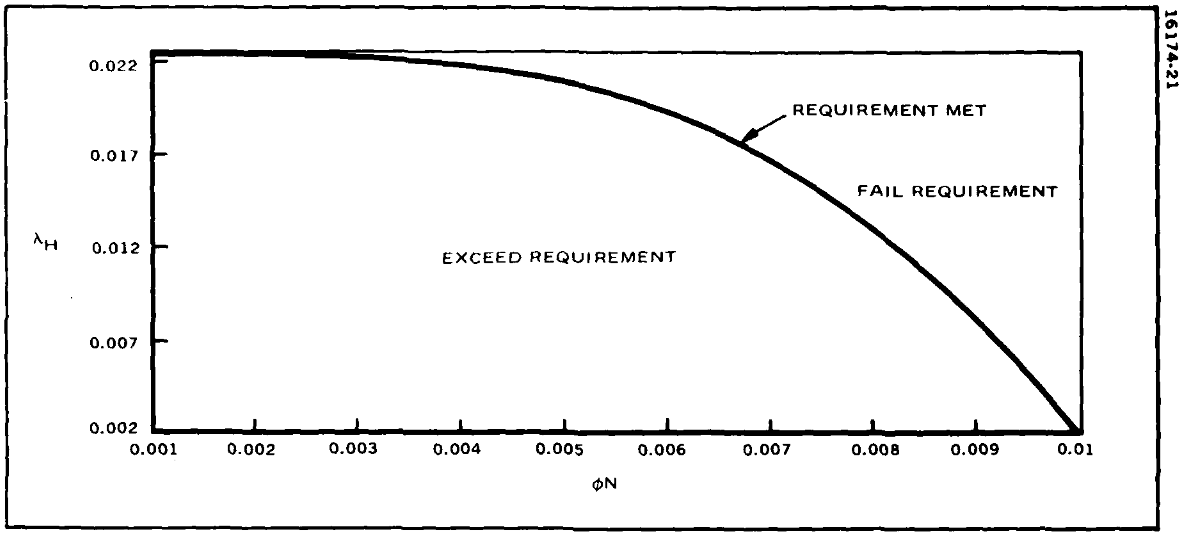


Figure 6.5-2. Isometric Curve of A* = 0.99 for the Redundant HW Example

Section 7.0

CONCLUSIONS AND RECOMMENDATIONS

A theory for combining HW and SW reliability models has been developed and used to derive a general combined HW/SW reliability model. This model provides an accurate description of the reliability/maintainability characteristics of systems possessing both HW and SW.

The general HW/SW model was applied to simple reliability constructs using a Markov-type SW process and extended to more complex reliability constructs including specific application to an example C^3 system. Under the assumptions of this SW process, the HW/SW model is compatible with the maintenance philosophies employed for C^3 systems. In particular, recovery of the SW operating system without correcting a fault, imperfect debugging, and numerous HW/SW modes of interaction are all features of the model which make it flexible enough to handle most C^3 systems.

Analyses of availability concepts using the HW/SW model indicate the inadequacy of the commonly used "steady-state" availability as a valid figure of merit, a fact which has been observed in C^3 systems. The availability of a typical C^3 system, being time dependent, will exhibit the transient effect of imperfect SW correction in the form of an undershoot below the steady-state value. The magnitude and duration of this transient are determined by the SW maintenance policy, the initial number of faults in the SW, the SW failure rate for a single fault, and the relative magnitude of SW failure rate and HW failure rate. These phenomena are reflected by parameters in the model. The SW maintenance policy is affected by the rate at which maintenance teams correct faults (this rate is $c = \lambda p$ where λ is the rate at which attempts are made and p is the probability of a successful correction). The initial number of SW faults is N and the SW failure rate associated with each fault is ϕ. When HW dominates the system in terms of failure rate, the transient undershoot of availability below steady state can be eliminated altogether. When HW and SW failure rates are of approximately the same order of magnitude, the values of c (rate of fault correction) and ϕ determine the transient. With ϕ held fixed, the length of time availability stays below steady state increases as c decreases and decreases as c increases, while with c fixed, the maximum value of the undershoot increases as ϕ increases, and decreases as ϕ decreases (see figure 4.1-3).

The implication of the foregoing has an obvious impact on specifying availability requirements for systems which contain embedded SW. The design and analysis of systems possessing both HW and SW should be carried out from the standpoint of minimum availability instead of the sometimes over-optimistic steady-state availability.

Using the experimental computer program representation of the HW/SW model, two approximations were considered: 1) "lumping" the SW failure and repair rates (e.g., using the "initial" SW failure rate) with the corresponding HW failure and repair rates, and 2) calculating the separate availability of HW and SW and then combining these to obtain a total HW/SW availability

$$A(t)_T = A(t)_{HW} \cdot A(t)_{SW}$$

Reasonable approximations could be obtained in general, however, only for the series case. The accuracy of lumping the HW and SW failure and repair rates is dependent on the amount of SW present in the system, becoming worse as steady-state is reached in a SW-dominant system. Calculating separate HW and SW availabilities provides a more accurate approximation than "lumping", although the difficulty in computing the SW availability term would probably also necessitate the use of a computer. For a redundant HW configuration, the approximation errors in these two cases are dependent on the number of HW units and the number of bugs initially present in the system, and can result in a very poor approximation in the transient period.

In order to make the model proposed in this study more practical it will be necessary to develop a comprehensive computer program for its implementation with a step-by-step user's guide on detailed applications. The computer program documented in this study is but an experimental version of such a program and serves only as a model for a more comprehensive computer program. An effort to develop, from the experimental program or other means, a program which can be used interactively to compute the relevant reliability measures and perform tradeoffs over wide ranges of the parameters is therefore recommended.

In addition to the development of an interactive computer program, a detailed study of the statistical aspects of estimating the model parameters including confidence estimation is needed. Such a study would involve the unification of techniques (which are at this time, scattered about the SW/HW reliability literature)for estimating the model parameters and the study of the statistical properties of the estimators.

Once these statistical properties have been established, it is also recommended that a combined HW/SW test methodology be developed similar to that developed for MIL-STD-781 "Reliability Tests: Exponential Distribution".

section 8.0

BIBLIOGRAPHY AND REFERENCES

Arszman, J. and Campbell, J. (1972). The analysis of computer availability. Technical report T6-77-7, Defence Tech. Info. Agency, Cameron Station, Virginia.

Athreya, K. B. and Karlin, S. (1971). On branching processes with random environments, I: extinction probabilities. Ann. Math. Statist., 42, 1449-1520.

Athreya, K. B. and Karlin, S. (1971). On branching processes with random environments, II: limit theorems. Ann. Math. Statist., 42, 1843-1858.

Barlow, R. E. and Proschan, F. (1965). Mathematical Theory of Reliability. New York: John Wiley and Sons.

Barlow, R. E. and Proschan, F. (1975). Statistical Theory of Reliability and Life Testing: Probability Models. Holt, Rinehart, and Winston, Inc.

Basin, S. L. (1974). Estimation of software error rate via capture-recapture sampling. Science Applications, Inc., Palo Alto, CA.

Boehm, B. W. et al (1973). Characteristics of Software Quality, TRW-Redondo Beach, TRW-SS-73-09.

Brooks, W. D. and Motley, R. W. (1980). Analysis of discrete software reliability models. IBM, final tech. report, RADC-TR-80-84. (A086334)

Brooks, W. D. and Weiler, P. W. (1977). Software reliability analysis. IBM Tech. Report FSD 77-0000, IBM Corp., Federal Systems Division, Caithersburg, MD.

Brown, J. R. and Lipow, M. (1975). Testing for software reliability. Proceedings, International Conference on Reliable Software, Los Angeles, CA.

Buck, R. C. (1956). Advanced Calculus. New York: McGraw-Hill Book Co.

Bulirsch, R. and Stoer, J. (1966). Numerical treatment of ordinary differential equations by extrapolation methods. Num. Math., 8, 1-13.

Burden, R. L., Faires, J. D. and Reynolds, A.C. (1978). Numerical Analysis. Boston, Mass.: Prindle, Weber and Schmidt.

Chung, K. L. (1967). Markov Chains with Stationary Transition Probabilities. New York: Springer-Verlag.

Cogburn, R. and Torrez, W. (1981). Birth and Death Processes With Random Environments in Continuous Time. Journal of Applied Probability, v. 18, pp. 19-30.

Costes, A., Landrault, C. and Laprie, J. C. (1978). Reliability and availability models for maintained systems featuring hardware failures and design faults. IEEE Tran. Comp., c-27.

Curtis, B. et. al. (1979). Measuring the psychological complexity of software maintenance tasks with the Halstead and McCabe metrics, IEEE Trans. on SE, SE-5, 96-104.

Dickson, J. C., Hesse, J. L., Kientz, A. C. and Shooman, M. J. (1972). Quantitative analysis of software reliability. Proc. Ann. Rel. and Maint. Symposium, New York.

Dynkin, E. B. (1961). Theory of Markov Processes. New Jersey: Prentice Hall.

Feller, W. (1957). An Introduction to Probability Theory and its Applications, 2nd ed., vol. I, New York: John Wiley and Sons.

Feller, W. (1966). An Introduction to Probability Theory and its Applications, vol. II, New York: John Wiley and Sons.

Feuer, A. R. and Fowlkes, E. B. (1979). Relating computer maintainability to software measures. Proc. 1979 Nat. Computer Conf., 1003-1012.

Fitzsimmons, A. B. (1978). Relating the presence of software errors to the theory of software science. Proc. Eleventh Hawaii Intrl Conf. on Software Sciences, 40-46.

Gilb, T., Controlling Maintainability: A Quantitative Approach for Software, Kolbotn, Norway (unpublished technical paper, available in U. S. from DACS/RADC Griffis AFB).

Glass, R. L. (1978). Patching is alive and, lamentably, thriving in the real-time world. ACM SIGPLAN Notices, 13.

Gloss-Soler, S. A. (1979). The DACS Glossary, A Bibliography of Software Engineering Terms. Under Contract to RADC, Rome Air Development Center, Griffis AFB, Rome, New York.

Goel, A. and Okumoto, K. (1978). Bayesian software prediction models, Bayesian software correction limit policies. Final Tech, Report, RADC-TR-78-155, Vol V (A067690)

Goel, A. L. (1977). Summary of technical progress: Bayesian software reliability prediction models. RADC-TR-77-112, Syracuse Univ.(A039022)

Goel, A. L. (1979). Reliability and other performance measures of computer software. Proc., First Int. Conf. on Rel. and Exploitation of Computer Sys., Wroclaw, Poland.

Goel, A. L. and Okumoto, K. (1978). An imperfect debugging model for software reliability, vol. I, Final Tech. Report, RADC-TR-78-155 (A057870)

Goel, A. L. (1978). Bayesian software correctio. .imit policies, vol. IV, Final Tech. Report, RADC-TR-78-155, (A057873)

Goel, A. L. (1978). A time dependent error detection rate model for a large scale software system. Proc. Third USA-Japan Computer Conf., New York, 35-40.

Goel, A. L. (1979). A Markovian model for reliability and other performance measures of software systems. Proc. Nat. Computer Conf., New York, vol. 48, 760-774.

Goel, A. L. and Okumoto, K. (1980). A time dependent error correction rate model for software performance assessment with applications. RADC-TR- 80-179, (A088186)

Hoel, P. G., Port, S. C. and Stone, C. J. (1972). Introduction to Stochastic Processes. New York: Houghton Mifflin.

Jelinski, Z. and Moranda, P. (1972). Software reliability research. Statistical Computer Performance Evaluation, New York: Academic Press, 465-484.

Kannan, D. (1979). An Introduction to Stochastic Processes. New York: North Holland.

Kaplan, N. (1973). A continuous time Markov branching model with random environments. Adv. Appl. Prob., 5, 37-54.

Kozlov, B. A. and Ushakov, I. A. (1970). Reliability Handbook. New York: Holt, Rinehart and Winston.

Landrault, C. & Laprie, J. (1977). Reliability and avail. modeling of systems featuring hardware and software faults. 7th Ann. Conf. Fault Tolerant Comp., Los Angeles, CA., IEEE.

Lipow, M. (1974). Software Reliability. 12th Ann. Reliability Engineering and Management Inst., Univ. of Arizona, Tucson, Arizona.

Lipow, M. (1977). Prediction of software failure. Proc. 1977 Ann. Reliability and Maintainability Symp., Phil., PA, 18-20.

Lipow, M. (1973). Maximum likelihood estimation of a software time-to-failure distribution. TRW Systems Group Report, 2260.1.9-73B-15, Redondo Beach, CA.

Littlewood, B. and Verrall, J. L. (1973). A Bayesian reliability growth model for computer software. Appl. Statistics, 24, 172-177.

Littlewood, B. (1975). A reliability model for systems with Markov structure, Appl. Statistics., 24, 172-177.

Littlewood, B. (1976). A semi-Markov model for software reliability with failure costs. Proc. MRI Symp. on Software Engineering, New York, 281-300.

Littlewood, B. (1979). How to measure software reliability and how not to. IEEE Trans. Rel., R-28, 1979.

Littlewood, B. Validation of a software reliability model. Second Software Life Cycle Management Workshop, Atlanta, GA., 146-159.

Loeve, M. (1963). Probability Theory. New York: D. Van Nostrand Co., Inc.

Mann, N. R., Schafer, R. E. and Singpurwalla, N. D. (1974). Methods for Statistical Analysis of Reliability and Life Data. New York: John Wiley and Sons.

McCall, J. A. and Matsumoto, M. T. (1980), Software Quality Metrics Enhancements. GE-Sunnyvale, RADC-TR-80-109, Vol I. (A086985)

McCall, J. A. and Matsumoto, M. T. (1980), Software Quality Measurement Manual. GE-Sunnyvale, RADC-TR-80-109, Vol II. (A086986)

Miyamoto, I. (1975). Software reliability in online real time environment. Proc. 1975 Int. Reliable Software, Los Angeles, CA., 108.

Moranda, P. B. (1976). Quantitative methods for software reliability measurements. McDonnell-Douglas Astronautics Co., AFOSR-TR-77-0046.

Moranda, P. B. (1975). Prediction of software reliability during debugging. Proc. 1975 Ann. Rel. Maint. Symp., 327-332.

Moranda, P. B. (1975). A comparison of software error-rate models. 1975 Texas Conference on Computing.

Moranda, P. B. and Jelinski, J. (1971). Software Reliability Research. Conference on Statistical Methods for the Evaluation of Computer System Performance, Providence, RI.

Moranda, P. B. (1975). Probability-based models for the failures during burn-in phase. Joint National Meeting ORSA/TIMS, Las Vegas, Nevada.

Motley, R. W. and Brooks, W. D. (1977). Statistical prediction of programming errors. Final Tech. Report, RADC-TR-77-175. (A041106)

Musa, J. D. (1975). A theory of software reliability and its application. IEEE Trans. on Software Engineering, SE-1, 312-327.

Nelson, E. C. (1975). Software Reliability.
TRW Software Series,
TRW-SS-75-05, Redondo Beach, CA.

Okumoto, K. and Goel, A. L. (1978). Classical and Bayesian inference for the software imperfect debugging model. vol. 2, RADC-78-155. (A057871)

Okumoto, K. and Goel, A. L. (1978). Availability analysis of software systems under imperfect maintenance. vol. 3, Final Tech. Report, RADC-TR-78-155. (A057872)

Purdue, P. (1974). The M/M/1 queue in a Markovian environment. Ops. Res., 22, 562-569.

Rao, C. R. (1973). Linear Statistical Inference and its Applications, 2nd ed., New York: John Wiley and Sons.

Ross, S. M. (1970). Applied Probability Models with Optimization Applications. Holden-Day.

Rushforth, C. K., Staffanson, F. L. and Crawford, A. E. (1979). Software reliability estimation under conditions of incomplete information. RADC-TR-79-230. (A080189)

Schafer, R. E. et. al. (1979). Validation of software reliability models. Final Tech. Report, RADC-TR-79-147. (A072113)

Schick, G. J. and Wolverton, R. W. (1973). Assessment of software reliability. Proc. Operations Research, Physica-Verlag, Wurzburg-Wien, 395-422.

Schick, G. J. and Wolverton, R. W. (1978). An analysis of competing software reliability models. IEEE Trans. Software Engineering, SE-4, 104-120.

Schneidewind, N. F. (1975). Analysis of error processes in computer software. Proc. Int. Conf. Reliable Software, Los Angeles, CA., 337-346.

Schneidewind, N. F. (1979). An experiment in software error data collection and analysis. IEEE, 5.

Schneider, V. (1980). Some Experimental Estimators for Developmental and Delivered Errors in Software Development Projects. Proc. COMPSAC 80, 495-498.

Shooman, M. L. (1972). Probabilistic models for software reliability prediction. Statistical Computer Performance Evaluation, Academic Press, 485-502.

Shooman, M. L. and Ruston, H. (1978). Software modeling studies. Summary of technical progress, vol I. Final Tech. Report, RADC-TR-78-4. (A052615)

Shooman, M. L. (1973). Operational testing and software reliability estimation during development. IEEE Symp. Computer Software Reliability, N.Y.C.

Shooman, M. L. (1975). Software reliability: measurement and models. 1975 Ann. Reliability and Maintainability Symp. 28-30.

Smith, W. (1968). Necessary conditions for the almost sure extinction of a branching process with random environments. Ann. Math. Statist., 39, 2136-2140.

Smith, W. and Wilkinson, W. (1969). On branching processes with random environments. Ann. Math. Statist., 40, 814-827.

Smith, W. and Wilkinson, W. (1971). Branching processes in Markovian environments. Ann. Math. Statist., 38, 749-763.

Solomon, F. (1975). Random walks in a random environment. Ann. Prob., 3, 1-31.

Sukert, A. N. (1977). A multi-project comparison of software reliability models. Computers in Aerospace Conference, 413-421.

Sukert, A. N. (1976). A software reliability modelling study, In-house Tech. Report, RADC-TR-76-247. (A030437)

Sukert, A. N. (1977). An investigation of software reliability models. Proc. Ann. Rel. Maint. Symp., Philadelphia, PA., 478-484.

Sukert, A. N. and Goel, A. L. (1980). A guidebook for software reliability assessment. Proc. Ann. Rel. Maint. Symp., San Francisco, CA., 188-190.

Sukert, A. N. (1978). Error modelling applications in software quality assurance. Proc. Software Quality and Assurance Workshop, San Diego, CA., 33-38.

Thayer, T. A., Lipow, M. and Nelson, E. C. (1976). Software reliability study. Final Tech. Report, RADC-TR-76-238. (A030798)

Torrez, W. (1978). The birth and death chain in a random environment: instability and extinction theorems. Ann. Prob., 6, 1020-1043.

Torrez, W. (1979). Calculating extinction probabilities for the birth and *death chain in a random environment*. J. Appl. Prob., 16.

Trivedi, T. A. and Shooman, M. (1975). Computer software reliability: many-state Markov modeling techniques. Interim Report, RADC-TR-75-169. (A014824)

Turn, R. (1978). Hardware-software tradeoffs in reliable software development. IEEE 11th Ann. Asilomar Conference of Circuits and Computers, 282-288.

Wagoner, W. L. (1973). The final report on a software reliability measurement study. Technology Division, The Aerospace Corp., report # TOR-0074(4112)-1, El Segundo, CA.

Walsh, T. J. (1979). A software reliability study using complexity measures. NCC Proc., 761-768.

Yau, S. S. and McGregor, T. E. (1979). On software reliability modeling. Interim Report, RADC-TR-79-127. (A072420)

Yau, S. S. (1980). Performance stability measures for software maintenance. Third Minnowbrook Workshop on Software Performance Evaluation.

Yin, B. and Winchester, J. (1978). The establishment and use of measures to evaluate the quality of software designs. ACM Conference Proc.

APPENDIX A

IMPORTANT NOTATIONS AND DEFINITIONS

$c\ (=\lambda p)$	SW fault correction rate.
F_{TC}	Combined SW maintainability measure.
$p_j(.,.,.,.)$	Solution to the Kolmogorov differential equations corresponding to the system state transition diagram when there are j bugs in the SW.
p	Probability of perfect debug.
λ	Rate at which SW maintenance personnel attempt to find and fix SW faults.
$X(t)$	Number of SW faults at time t.
$Y(t)$	System state at time t.
$0, 1, \ldots, J$	Totality of system states; $0 \equiv$ full-up.
OP	The set of system operational states.
$t_0, t_1, \ldots$	Jump times for $X(t)$.
absorbing state	An absorbing state is a state such that once a process reaches it, the process remains there forever.
N_H	Total number of HW units.
M	Number of HW units required.
$P\ (E_j)$	Probability of module execution.
ϕ	Constant of proportionality related to SW failure rate.
λ_H	HW failure rate.
μ_H	HW repair rate.
μ_S	SW "repair" rate.
ν	HW standby unit duty cycle.
δ_M	SW maintainability figure of merit.

APPENDIX B

PURELY DISCONTINUOUS MARKOV PROCESSES

The material in this section is adapted from Kannan (1979), Feller (1957, 1966), and Hoel, Port and Stone (1972). Throughout this report, all random processes will be defined on the same probability space (E, F, P) where E is the set of elementary events, F is a sigma algebra of subsets of E, and P is a probability measure defined on F.

By stochastic process (sp) is meant a collection of random variables X(t), $t \geq 0$ defined on (E, F, P). The range of X(t) will be assumed to be a finite collection of numbers which can be taken, without loss of generality, to be $\{0, 1, 2, \ldots, J\}$. When X(t) is taken to be the state of the SW, for example, $X(t) = k$, $0 \leq k \leq J$, represents the number of faults remaining in the SW. Similarly, when X(t) is taken to be the state of a system, $X(t) = k$, $1 \leq k \leq J$, could be degraded and failed states while $X(t) = 0$ represents the full-up state. In these examples, the index variable t is assumed to be the time.

The sp $\{X(t), t \geq 0\}$ is called a Markov process if for $0 \leq t_0 < t_1 < \ldots < t_n < s < t$ and integers $x_0, x_1, \ldots, x_n, x, y$ in $\{0, 1, \ldots, J\}$:

$$P\{X(t) = y \,|\, X(t_k) = x_k,\ 0 \leq k \leq n,\ X(s) = x\}$$

$$= P\{X(t) = y \,|\, X(s) = x\}. \tag{B.1-1}$$

If, in addition to (B.1-1), for all $t > s \geq 0$

$$P\{X(t) = y \,|\, X(s) = x\} = P\{X(t-s) = y \,|\, X(0) = x\} \tag{B.1-2}$$

the process is called a time-homogeneous Markov process. The property (B.1-1) can be interpreted by saying that the process' future is independent of its past evolution, given the present value of the process. When both (B.1-1) and (B.1-2) are satisfied, the function $p(h,x,y) = P\{X(h) = y \,|\, X(0) = x\}$ is called the transition function of the process. Both (B.1-1) and (B.1-2) will be assumed hereafter.

The function $q(x)$, $x \in \{0,1,\ldots,J\}$ is called the intensity function of the Markov process if $q(x)\Delta t + o(\Delta t)$ is the probability that X(t) will undergo a random change in the time interval $(t, t+\Delta t)$ when $X(t) = x$. The conditional probability $Q(x,y)$ that $X(t+\Delta t) = y$ given that $X(t) = x$ and a change takes place in $(t, t+\Delta t)$, is called the relative transition function of the Markov process. It is clear that for all $x, y \in \{0, 1, \ldots, J\}$:

$$q(x) \geq 0$$

$$Q(x,x) = 0 \qquad \text{(B.1-3)}$$

$$\sum_{k=0}^{J} Q(x,k) = 1$$

The stationary Markov process $\{X(t), t \geq 0\}$ is called a purely discontinuous Markov process if in an arbitrary time interval $(t, t+\Delta t)$, $X(t)$ undergoes a change with probability $q(x)\, \Delta t + o(\Delta t)$, remains unchanged with probability $1-q(x)\Delta t + o(\Delta t)$ and undergoes more than one change with probability $o(\Delta t)$. Thus, if $\{X(t), t \geq 0\}$ is a purely discontinuous stationary Markov process with transition function p, intensity function q, and relative transition function Q, it follows that for s, $t \geq 0$, $x,y \varepsilon \{0,1,\ldots,J\}$:

$$p(s,x,y) = \begin{cases} 1 - q(x)s + o(s); \text{ if } x = y \\ sq(x)\, Q(x,y) + o(s)\,; \text{ if } x \neq y \end{cases} \qquad \text{(B.1-4)}$$

$$p(s+t, x, y) = \sum_{k=0}^{J} p(t,x,k)\, p(s,k,y) \qquad \text{(B.1-5)}$$

Substituting (B.1-4) into (B.1-5) yields

$$p(s+t,x,y) = p(t,x,y)(1-q(y)s+o(s)) + \sum_{k\neq y} p(t,x,k)\,[sq(k)\, Q(k,y) + o(s)]$$

so that

$$p(s+t,x,y) - p(t,x,y) = -sq(y)\, p(t,x,y) + \sum_{k\neq y} p(t,x,k)\, sq(k)\, Q(k,y) + o(s). \qquad \text{(B.1-6)}$$

Dividing (B.1-6) by s and letting s tend to zero through positive values yields the so-called Kolmogorov forward differential equations.

$$\frac{\partial}{\partial t} p(t,x,y) = \sum_{k=0}^{J} p(t,x,k)\, A(k,y),\ y = 0,1,\ldots,J \qquad \text{(B.1-7)}$$

where

$$A(x,y) = \begin{cases} -q(x) & \text{if } x = y \\ q(x)\, Q(x,y) & \text{if } x \neq y \end{cases} \qquad \text{(B.1-8)}$$

AD-A116 566 HUGHES AIRCRAFT CO FULLERTON CA GROUND SYSTEMS GROUP F/G 9/2
COMBINED HW/SW RELIABILITY MODELS.(U)
APR 82 L E JAMES, J E ANGUS, J B BOWEN F30602-80-C-0085
UNCLASSIFIED HAC-FR-81-16-351A RADC-TR-82-68 NL

2 OF 2
AD A
116566

END
DATE
FILMED
7-82
DTIC

The values $A(x,y)$ are called infinitesimal parameters (or transition rates when $x \neq y$). Throughout this report A (or A_j) will denote a matrix with elements $A(x,y)$ (or $A_j(x,y)$).

A useful fact in studying purely discontinuous stationary Markov processes is that if such a process starts in state x at some time t, then the amount of time spent in that state is exponentially distributed with mean $\mu(x) = [q(x)]^{-1}$ where q is the intensity function of the process. With this fact and having derived the foregoing mathematical properties, it is useful to describe the evolution of such a process (see Figure B.1-1). The process starts at time 0 in state x_0, say. It remains there for a length of time τ_1, and jumps to state x_1 with probability $Q(x_0, x_1)$. The process remains in state x_1 for a length of time τ_2 which is exponentially distributed with parameter $q(x_1)$, and then jumps to state x_2 with probability $Q(x_1, x2)$; etc. The paths of such a process are step functions with probability 1.

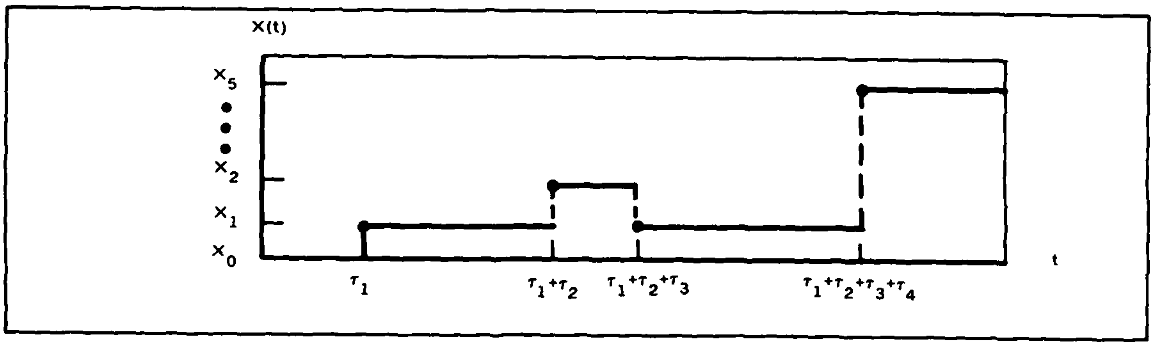


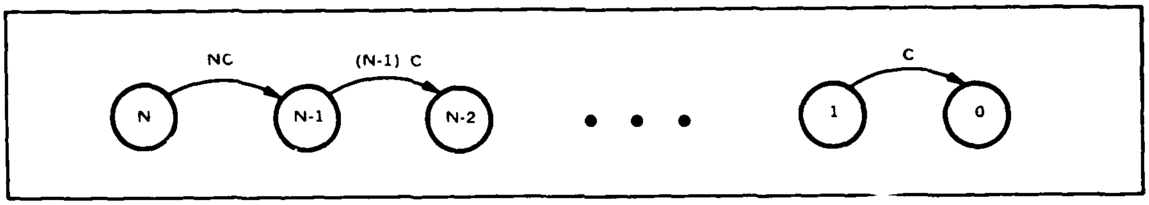
Figure B.1-1. Typical Path of a Purely Discontinuous Stationary Markov Process

APPENDIX C

A MARKOVIAN SW PROCESS

In this section a special case of a purely discontinuous stationary Markov process is worked out and shown to be probabilistically equivalent to the Goel/ Okumoto Imperfect Debugging model (1978). The transition function is derived in closed form and related to the binomial probability function.

Assume that the number of software faults present is a system at time t. X(t), is a purely discontinuous stationary Markov process $\{X(t),\ t \geq 0\}$ with range $\{0,\ 1,\ \ldots,\ N\}$ and transition diagram given by:



16174-12

The interpretation of this diagram is as follows. Supposing that the process starts in state N at time zero (i.e., X(0) = N), the next transition is always to N-1, with transition rate = A(N,N-1) = Nc. When the next transition occurs, it always takes the process to state N-2, with transition rate A(N-1, N-2) = (N-1)c. The process always transitions to the next lowest state until it hits state 0 where it remains. For this process, the parameters described in Appendix B are given by:

$$q(x) = cx,\ x = 0,\ 1,\ \ldots,\ N;\ c > 0,$$

$$Q(x,y) = \begin{cases} 1 & \text{if } y = x - 1 \\ 0 & \text{otherwise} \end{cases}$$

$$A(x,y) = \begin{cases} -cx; & x = y,\ 0 \leq x \leq N \\ cx; & y = x-1,\ 1 \leq x \leq N \\ 0 & \text{otherwise} \end{cases}$$

Such a process is called a linear pure death process. The Kolmogrov forward differential equations are:

$$\frac{\partial}{\partial t}\, p(t,x,x) = -cxp(t,x,x),$$

$$\frac{\partial}{\partial t}\, p(t,x,y) = -cyp(t,x,y) + c(y+1)\, p(t,x,y+1),\ 0 \leq y \leq x - 1 \qquad (C.1\text{-}1)$$

The solution of this system of equations under the initial conditions

$$p(0,x,y) = \begin{cases} 1 & \text{if } x = y = N \\ 0 & \text{otherwise} \end{cases}$$

(i.e., $X(0) = N$ with probability 1) is

$$p(t,N,y) = \binom{N}{y} e^{-cty} (1-e^{-ct})^{N-y}, \quad 0 \le y \le N \qquad (C.1\text{-}2)$$

From the definition of p, (C.1-2) can be rewritten as

$$P\{X(t) = y \mid X(0) = N\} = \binom{N}{y} e^{-cty} (1-e^{-ct})^{N-y}, \quad 0 \le y \le N \qquad (C.1\text{-}3)$$

In the Imperfect Debugging Model, it is assumed that N software faults are present initially, all independent of one another, each of which having constant occurrence rate $\lambda > 0$. The probability of two or more faults occurring simultaneously is assumed negligible and no new faults are introduced. At most, one fault is corrected at correction time and the time to correct a fault is neglected. The fault causing a failure, when detected, is corrected with probability p, $0 < p < 1$ and not removed with probability $q = 1-p$. Let $X^*(t)$ denote the number of faults remaining in the software at time t. It will be shown that $X^*(t)$ has the distribution (C.1-3) with X replaced with X^* and $c = \lambda p$ so that X^* is in fact equivalent to a purely discontinuous stationary Markov process, a special case of the more general semi-Markov process described by Goel and Okumoto.

Equation (3.27) of Goel & Okumoto gives

$$P\{X^*(t) = y \mid X^*(0) = N\} = G_{N,y}(t) - G_{N,y-1}(t), \quad y=0,1,\ldots,N \qquad (C.1\text{-}4)$$

where

$$G_{N,y}(t) \equiv \begin{cases} 1 & ; y = N \\ 0 & ; y \le -1 \\ \displaystyle\sum_{j=1}^{N-y} \frac{N!\,(-1)^{j-1}\,j}{y!\,j!\,(N-y-j)!\,(y+j)} \left(1-e^{-(y+j)p\lambda t}\right), & 0 \le y \le N-1 \end{cases} \qquad (C.1\text{-}5)$$

Expression C.1-4 is simplified as follows. For $0 \le y \le N-1$, (C.1-5) can be used to give

$$G_{N,y}(t) - G_{N,y-1}(t) = \sum_{j=1}^{N-y} \frac{N!(-1)^{j-1}\; j\left(1-e^{-(y+j)p\lambda t}\right)}{y!j!(N-y-j)!(y+j)}$$

$$- \sum_{j=1}^{N-y+1} \frac{N!(-1)^{j-1} j\left(1-e^{-(y-1+j)p\lambda t}\right)}{(y-1)!j!\;(N-y+1-j)!\;(y-1+j)}$$

$$= \sum_{j=1}^{N-y} \frac{N!(-1)^{j-1}\; j\left(1-e^{-(y+j)p\lambda t}\right)}{y!j!(N-y-j)!(y+j)}$$

$$+ \sum_{j=0}^{N-y} \frac{N!(-1)^{j-1}(j+1)\left(1-e^{-(y+j)p\lambda t}\right)}{(y-1)!(j+1)!(N-y-j)!(y+j)}$$

$$= \frac{N!}{(N-y)!y!} \sum_{j=0}^{N-y} \frac{(N-y)!(-1)^{j-1}\left(1-e^{-(y+j)p\lambda t}\right)\;(y+j)}{j!(N-y-j)!(y+j)}$$

$$= \binom{N}{y} \sum_{j=0}^{N-y} \binom{N-y}{j} (-1)^j\, e^{-(y+j)p\lambda t}$$

$$= \binom{N}{y} e^{-yp\lambda t}\, (1-e^{-p\lambda t})^{N-y}.$$

The correspondence of (C.1-3) and (C.1-4) is clear. The interpretation of this correspondence is that the Imperfect Debugging Model is equivalent to a linear pure death process starting at N with infinitesimal parameters defined by:

$$A(x,y) = \begin{cases} -p\lambda x & , \quad x = y \\ p\lambda x & , \quad y = x-1 \\ 0 & , \quad \text{otherwise.} \end{cases}$$

The effect of imperfect debugging is also clear. The "perfect debugging rate" for a single fault, namely λ, is simply reduced by a factor of p, the probability of successfully correcting the fault.

To complete the theory for the baseline combined HW/SW reliability model, it is necessary to derive the distributions for the times at which jumps occur in the process X(t) defined in the beginning of this section. It is clear from the structure of X(t) that the time at which the kth jump occurs (assuming $X(0) = N$) is the sum of independent exponentially distributed random variables $\tau_1, \ldots, \tau_k$ where $E\tau_j = 1/[(N-j+1)c)]$.

Goel and Okumoto compute the distribution of the time required to a specified number of bugs y which is equivalent to the distribution of the time of the kth jump in $X^*(t)$ where $k = N-y$. The expression they derive is given by the cumulative distribution defined by $G_{N,y}(t)$ in (C.1-5) with $0 \le y \le N-1$.

Having derived expression (C.1-3) for the X(t) process the cumulative distribution function of the time t_k of the kth jump in X(t) can be derived by noticing that the event $\{X(s) \le N-k\}$ is equivalent to the event $\{t_k \le s\}$ so that

$$P\left\{t_k \le s\right\} = P\left\{X(s) \le N-k\right\} = \sum_{j=0}^{N-k} \binom{N}{j} e^{-ctj} (1-e^{-ct})^{N-j} \qquad (C.1-6)$$

for $s \ge 0$ and $k = 1, 2, \ldots N$, the distribution of t_0 being degenerate since $t_0 = 0$ will be a convention. If $\tilde{t}_y$ is defined as the time at which X(t) first reaches y, $y = 0, 1, \ldots, N$, then obviously $\tilde{t}_y = t_{N-y}$ and (C.1-6) gives

$$P\left\{\tilde{t}_y \le t \;\middle|\; X(0) = N\right\} = \sum_{j=0}^{y} \binom{N}{j} e^{-ctj} (1-e^{-ct})^{N-j} \qquad (C.1-7)$$

for $y = 0, 1, \ldots, N-1$ and $t \ge 0$. If $c = p\lambda$ and X is replaced by X* of the Imperfect Debugging Model, then $\tilde{t}_y$ is the time at which there are y remaining errors and (C.1-7) is equivalent to $G_{N,y}(t)$ as given in Goel & Okumoto and reproduced in (C.1-5).

APPENDIX D

DEVELOPMENT OF A COMBINED HW/SW RELIABILITY MODEL USING A MARKOVIAN SW PROCESS

In this section some criteria will be presented for selecting a model for the SW process $\{X(t), t \geq 0\}$ and these criteria will be used to show how the calculation of the combined HW/SW reliability model (3.2-2) can be accomplished.

A particular model for the SW process, namely the Goel/Okumoto Imperfect Debugging Model described in Appendix C, will be adopted for the combined HW/SW reliability model in order to analyze the expressions for the state occupancy probabilities and derive reliability measured defined in Section 3.3.

D.1 CRITERIA FOR SELECTING THE SW PROCESS

Without some additional assumptions on the structure of the process X(t) (i.e., the number of faults remaining in the SW at time t) there is little use for expression (3.2-2). So, as a first criterion it will be assumed that, in addition to X(t) being a purely discontinuous time-stationary Markov process taking values in $\{0, 1, \ldots, N\}$ and having $X(0) = N$, it will further be assumed that X(t) always moves one unit to the left at each transition. That is, X(t) will be assumed to be a "pure-death" process in the sense that errors are never added (births) to the SW. The partition $0 \equiv t_0 < t_1 < \ldots < t_N < t_{N+1} = t_{N+2} = \ldots = +\infty$ will denote the random times at which X(t) transitions. Mathematically, this implies that the probabilities Q(x,y) defined in (B.1-3) take the form

$$Q(x,y) = \begin{cases} 1 & \text{if } y = x-1 \\ 0 & \text{otherwise} \end{cases} \qquad \text{(D.1-1)}$$

and moreover,

$$P\left\{X(t_k) = N-k \mid X(0) = N\right\} = 1 \qquad \text{(D.1-2)}$$

for k = 0, 1, ..., N.

Physically this means that when the number of faults in the SW changes, it always decreases by 1. That is, bugs are not added by maintenance team interventions and no more than one bug is removed at a time.

Because of this newly imposed structure on X, it is more convenient to compute $P\{Y(t) = n\}$ by conditioning on $\{t_k \leq t < t_{k+1}\}$ so that

$$P\{Y(t) = n\} = \sum_{k=0}^{N} P\left\{Y(t) = n \mid t_k \leq t < t_{k+1}\right\} P\left\{t_k \leq t < t_{k+1}\right\} \ . \qquad \text{(D.1-3)}$$

Assuming $X(0) = N$, then because of the structure of X, the events $\{t_k \leq t < t_{k+1}\}$ and $\{X(t) = N-k\}$ are equivalent so that the last written probability in (D.1-3) is computed by finding the distribution of $X(t)$ via the theory of Appendix B. It remains to express $P\{Y(t) = n \mid t_k \leq t < t_{k+1}\}$ in terms of computable quantities. But, because of the construction of the sample paths of $Y(t)$ it is easy to write down the aforementioned conditional probability in terms of the p_j defined by (3.2-1), i.e.

$$P\left\{Y(t) = n \,\middle|\, t_k \leq t < t_{k+1}\right\} =$$

$$\int \cdots \int \Sigma \cdots \Sigma \; p_N(s_1, 0, i_1) p_{N-1}(s_2, i_1, i_2) \cdots p_{N-k+1}(s_k, i_{k-1}, i_k)$$

$$p_{N-k}\left(t - \sum_{\ell=1}^{k} s_\ell, i_k, n\right) \cdot f_{\tau_1, \ldots \tau_{k+1} \mid \{t_k \leq t < t_{k+1}\}}(s_1, \ldots, s_{k+1}) ds_1 ds_2 \ldots ds_{k+1} \qquad \text{(D.1-4)}$$

for $k=1, \ldots, N$, $t_q = \sum_{\ell=1}^{q} \tau_\ell$, $\tau_i \equiv t_i - t_{i-1}$, and

$$P\left\{Y(t) = n \,\middle|\, 0 \leq t < t_1\right\} = p_N(t, 0, n). \qquad \text{(D.1-5)}$$

Here, it is assumed that $Y(0) = 0$. The multiple integral in (D.1-4) extends over all $s_1 \geq 0$, ..., $s_{k+1} \geq 0$ such that

$$\sum_{\ell=1}^{k} s_\ell \leq t < \sum_{\ell=1}^{k+1} s_\ell,$$

f is the joint density function of $\tau_1, \ldots, \tau_{k+1}$ conditioned on the event $\{t_k \leq t < t_{k+1}\}$, and the multiple sum in (D.1-4) extends over integers $0 \leq i_1 \leq J$, ..., $0 \leq i_k \leq J$.

Equation (D.1-3) can thus be computed by finding p_j (solving, numerically possibly, a system of linear differential equations) and then using (D.1-4) and (D.1-5) to obtain $P\{Y(t) = n\}$. However, an additional assumption will render (D.1-4) and (D.1-5) more tractable.

Since a well designed system will spend the great majority of time in some operational state (this is true in view of typical availability requirements) it is a good approximation to assume that the state of the system at random times $t_1 < t_2 < \ldots < t_N$ is always the same operational state. In fact, although it is not necessary to do so, it can be assumed that the state of the system at times

t_i, i = 1, ..., N is full-up. As an assumption, this is not unreasonable since repair efforts are generally undertaken as items fail so that a system is most likely to occupy the full-up state immediately following a down-state or degraded state. In practical terms, this is equivalent to implementing SW changes only during full-up periods. In mathematical terms, this is equivalent to forcing the points $0<t_1<\ldots<t_N$ to be regeneration-points of the process. Thus, it will now be assumed that $Y(0) = Y(t_1) = \ldots = Y(t_N) = 0$ where 0 represents the full-up state. Under this provision, (D.1-4) can be further simplified, and the calculations performed as follows:

For $1\le k\le N$,

$$P\left\{Y(t) = n \mid t_k \le t < t_{k+1}\right\} = \int_0^t P\left\{Y(t) = n \mid t_k \le t < t_{k+1},\ t_k = y\right\}$$

$$dP\left\{t_k \le y \mid t_k \le t < t_{k+1}\right\} = \int_0^t p_{N-k}(t-y,0,n)\,dP\left\{t_k \le y \mid t_k \le t < t_{k+1}\right\}. \quad \text{(D.1-6)}$$

When k = 0, expression (D.1-5) completes the calculation. The probability distribution $P\{t_k \le y \mid t_k \le t < t_{k+1}\}$ can be computed when the process X(t) is specified.

In equations (D.1-4) through (D.1-6) it has been assumed that Y(0) = 0 with probability 1. If it is desired to have Y(0) = i where i is an arbitrary operational state it is necessary only to change $p_N(s_1,0,i_1)$ to $p_N(s_1,i,i_1)$ in (D.1-4) and $p_N(t,0,n)$ to $p_N(t,i,n)$ in (D.1-5). Also, in (D.1-4) through (D.1-6) it is tacitly assumed that Y can reach full-up from any other state.

D.2 THE COMBINED HW/SW RELIABILITY MODEL

The process X(t) described in Appendix C (and shown to be equivalent to the Goel/Okumoto Imperfect Debugging Model) satisfies (D.1-1) and (D-1-2), and

$$P\{X(t) = j \mid X(0) = N\} = \binom{N}{j} e^{-cjt}\,(1-e^{-ct})^{N-j} \quad \text{(D.2-1)}$$

for j = 0, 1, ..., N. In addition, if $0\equiv t_0<t_1<\ldots<t_N$ denote the jump times of X(t), then the increments $\tau_i = t_i - t_{i-1}$, i = 1, ..., N are independent, exponentially distributed random variables with densities given by

$$f_{\tau_i}(s) = \begin{cases} [c(N-i+1)]\ \exp\left\{-s[c(N-i+1)]\right\}, & s\ge 0 \\ 0, & s<0 \end{cases} \quad \text{(D.2-2)}$$

for i = 1, ..., N. In (D.2-1) and (D-2-2), $c = p\lambda$ where $p\varepsilon(0.1)$ is the probability of a "perfect" debug and $\lambda > 0$ is the rate of maintenance troubleshooting. From now on, this X(t) process will be assumed.

The conditional distribution $P\{t_k \le y \,|\, t_k \le t < t_{k+1}\}$ can now be computed for $y \le t$:

$$P\left\{t_k \le y,\ t_k \le t < t_{k+1}\right\} = P\left\{t_k \le y,\ t_k \le t < t_k + \tau_{k+1}\right\}$$

$$= \int_o^y \int_{t-s}^{\infty} c(N-k)\ e^{-s[c(N-k)]}\ ds\ dP\left\{t_k \le s\right\}$$

$$= \int_o^y e^{-c(N-k)(t-s)}\ f_{t_k}(s)\ ds \qquad \text{(D.2-3)}$$

where $f_{t_k}(s)$ is the derivative of the right-hand side expression of (C. 1-6). Thus,

$$P\left\{t_k \le y \,|\, t_k \le t < t_{k+1}\right\} = \begin{cases} \dfrac{\int_o^y e^{-c(N-k)(t-s)} f_{t_k}(s)\ ds}{P\left\{t_k \le t < t_{k+1}\right\}}; & 0 \le y < t \\ 1; & y \ge t \\ 0; & y < 0 \end{cases} \qquad \text{(D.2-4)}$$

Substituting (D.2-4) into (D.1-6) gives

$$P\left\{Y(t) = n \,|\, t_k \le t < t_{k+1}\right\}$$

$$\frac{\int_o^t p_{N-k}(t-y,0,n) e^{-c(N-k)(t-y)}\ f_{t_k}(y) dy}{P\left\{t_k \le t < t_{k+1}\right\}} \qquad \text{(D.2-5)}$$

When k = 0,

$$P\left\{Y(t) = n \,|\, 0 \le t < t_1\right\} = p_N(t,0,n), \qquad \text{(D.2-6)}$$

since $t_o = 0$. Substituting (D.2-5) and (D.2-6) into (D.1-3) gives

$$P\{Y(t) = n\} = e^{-Nct}\, p_N(t,0,n)$$

$$+ \sum_{k=1}^{N} \int_o^t p_{N-k}(t-y,0,n)\, e^{-c(N-k)(t-y)} f_{t_k}(y)dy \qquad (D.2\text{-}7)$$

or changing summation index,

$$P\{Y(t) = n\} = e^{-Nct}\, p_N\ (t,0,n)$$

$$+ \sum_{j=0}^{N-1} \int_o^t p_j(t-y,0,n) e^{-cj(t-y)} f_{t_{N-j}}(y)dy \qquad (D.2\text{-}8)$$

From equation (C.2-7), it follows that for $0 \le y \le N-1$,

$$f_{t_{N-j}}(y) = \frac{d}{dy}\left\{ \sum_{\ell=0}^{j} \binom{N}{\ell} e^{-c\ell y}\, (1-e^{-cy})^{N-\ell} \right\}$$

$$= \sum_{\ell=0}^{j} \sum_{m=0}^{N-\ell} \binom{N}{\ell}\binom{N-\ell}{m}(-1)^{m+1}\, c(\ell+m)e^{-c(\ell+m)y}, \quad y \ge 0, \qquad (D.2\text{-}9)$$

so that (D.2-8) becomes

$$P\{Y(t) = n\} = e^{-Nct}\, p_N(t,0,n)$$

$$+ \sum_{j=0}^{N-1} \sum_{\ell=0}^{j} \sum_{m=0}^{N-\ell} \left[\binom{N}{\ell} \binom{N-\ell}{m} (-1)^{m+1}\, c(\ell+m) \right.$$

$$\left. \cdot \int_o^t p_j(t-y,0,n) e^{-c[(\ell+m)y+j(t-y)]} dy \right] \qquad (D.2\text{-}10)$$

or changing variables in the integral;

$$P\{Y(t) = n\} = e^{-Nct}\, p_N(t,0,n)$$

$$+ \sum_{j=0}^{N-1} \sum_{\ell=0}^{j} \sum_{m=0}^{N-\ell} \Bigg[\binom{N}{\ell} \binom{N-\ell}{m} (-1)^{m+1}\, c(\ell+m) e^{-c(\ell+m)t}$$

$$\cdot \int_0^t p_j(s,0,n) e^{-c(j-\ell-m)s} ds \Bigg] \cdot \qquad \text{(D.2-11)}$$

Equation (D.2-11) actually gives the probability of the system being in state n at time t conditioned on starting in state 0 (full-up) at time 0, i.e. $P\{Y(t) = n|Y(0) = 0\}$. Often, however, it is necessary to consider $P\{Y(t) = n|Y(0) = i\}$ for some arbitrary $i\epsilon\{0, 1, \ldots, J\}$. But the computations are nearly the same with the result being

$$P\{Y(t) = n|Y(0) = i\} = e^{-Nct}\, p_N(t,i,n)$$

$$+ \sum_{j=0}^{N-1} \sum_{\ell=0}^{j} \sum_{m=0}^{N-\ell} \Bigg[\binom{N}{\ell} \binom{N-\ell}{m} (-1)^{m+1}\, c(\ell+m) e^{-c(\ell+m)t}$$

$$\int_0^t p_j(s,0,n) e^{-c(j-\ell-m)s} ds \Bigg] \qquad \text{(D.2-12)}$$

where $(n,i)\varepsilon\{0, 1, \ldots, J\}$. In deriving (D.2-12) it should be pointed out that $Y(t_1) = \ldots = Y(t_N) = 0$ is still assumed.

Equation (D.2-12) specifies the entire probability structure of the combined HW/SW reliability model (when adapted to the Goel/Okumoto SW model) necessary to derive the availability measure in Section 3.3. A computer program has been developed to compute (D.2-12) when $i = 0$. This program is documented in Appendix F. The availability, assuming $Y(0) = 0$, is given by (3.3-1) with $P\{Y(t) = m\}$ computed from (D.2-12) with $i = 0$.

In order to derive the measures (3.3-2), (3.3-3), and (3.3-4) it will be necessary to derive $P\{Y(t) = n|Y(0) = i\}$ under the provision that all failed states be made absorbing states. The expression (D.2-12) cannot be used directly since its derivation depends on the tacit assumption that the full-up state can be reached from any other state.

For the moment, assume that the transition diagrams and infinitesimal matrices A_j have been modified so that all failed states are combined into one absorbing state i_F, and suppose that $p_j(u,\ell,m)$, the solutions to (B.1-7), have been derived under this provision. Suppose also that n and i are operational states. Given that $Y(0) = i$ and that $t_k \le t < t_{k+1}$ $Y(t) = n$ can only happen if Y is not absorbed in any of the intervals $(0, t_1]$, $(t_1, t_2]$, ... $(t_{k-1}, t_k]$ and Y successfully transitions to n at time t starting from time t_k. Using the notation of Section D.1, it follows that for $k \ge 1$ and $Y(0) = i$

$$P\left\{Y(t) = n \,|\, t_k \le t < t_{k+1}\right\}$$

$$= \int \cdots \int \left[1 - p_N(s_1,i,i_F)\right]\left[1 - p_{N-1}(s_2,0,i_F)\right] \cdots \left[1 - p_{N-k+1}(s_k,0,i_F)\right]$$

$$p_{N-k}\left(t - \sum_{\ell=1}^{k} s_\ell, 0, n\right) f_{\tau_1,\ldots,\tau_{k+1}|\{t_k \le t < t_{k+1}\}}(s_1,\ldots,s_{k+1})\,ds_1 ds_2 \ldots ds_{k+1} \tag{D.2-13}$$

where the integration is performed over the region

$$s_1 \ge 0, \;\ldots,\; s_{k+1} \ge 0, \;\sum_{\ell=1}^{k} s_\ell \le t < \sum_{\ell=1}^{k+1} s_\ell.$$

When k = 1, (D.2-13) reduces to

$$P\left\{Y(t) = n \,|\, 0 \le t < t_1\right\} = p_N(t,i,n). \tag{D.2-13a}$$

Combining these results with (D.2-2) gives

$$P\{Y(t) = n \,|\, Y(0) = i\} = p_N(t,i,n)\; e^{-Nct}$$

$$+ \sum_{k=1}^{N} \int \cdots \int \Bigg\{ \left[1 - p_N(s_1,i,i_F)\right]$$

$$\prod_{j=2}^{k} \left[1 - p_{N-j+1}(s_j,0,i_F)\right] p_{N-k}\left(t - \sum_{\ell=1}^{k} s_\ell, 0, n\right)$$

$$\cdot \prod_{j=1}^{k+1} \left[c(N-j+1)\right] \exp\left\{-s_j \cdot c(N-j+1)\right\} ds_j \Bigg\} \tag{D.2-14}$$

where the integration is performed over

$$s_1 \geq 0, \ \ldots, \ s_{k+1} \geq 0, \ \sum_{\ell=1}^{k} s_\ell \leq t < \sum_{\ell=1}^{k+1} s_\ell .$$

It should be emphasized that (D.2-14) differs from (D.2-12) because (D.2-14) is derived under the condition that all failed states are collapsed into one absorbing state i_F whereas the assumption in (D.2-12) is that no states are absorbing.

Using (D.2-14) in (3.3-2) and (3.3-3) will yield the probability of failure-free operation starting at i and of duration t, and the MTTF_i, respectively.

The non-stationary reliability coefficient defined in (3.3-5) is very difficult to compute but the steady state reliability coefficient $R(\infty, t_0)$ defined in (3.3-6) can be computed if it can be shown that for the baseline model,

$$\lim_{t \to \infty} P\{Y(t) = n | Y(0) = i\} \equiv \pi(n)$$

exists (n = 0, 1, ..., J) and is independent of i. To see that $\pi(n)$ is well defined for the baseline model consider expression (D.1-3) under the provision that Y(0) = i. Because of the baseline model assumptions,

$$P\{t_k \leq t < t_{k+1}\} = \binom{N}{N-k} e^{-c(N-k)t} (1-e^{-ct})^k$$

for k = 0, 1, ..., N (this follows from the comment after expression (D.1-3) and expression (D.2-1)). Unless k = N, this latter expression $\to 0$ as $t \to \infty$ so that the only term in (D.1-3) which can contribute to the limit is the term with k = N, i.e. if the limit exists, then

$$\lim_{t \to \infty} P\{Y(t) = n | Y(0) = i\} = \lim_{t \to \infty} P\{Y(t) = n | t_N \leq t\} P\{t_N < t\}.$$

Using (D.2-5), it follows that

$$P\{Y(t) = n | t_N \leq t\} P\{t_N \leq t\} = \int_0^t p_0(t-y,0,n) f_{t_N}(y) dy.$$

Since the state-space for Y is finite, then if the full-up state can be reached from any other state in a finite number of transitions when X(t) = 0 (i.e., the transition diagram and A_j discussed in Section 3.2 are such that when j = 0, the full-up state can be reached from any other state which is not absorbing in a finite number of transitions) then

$$\lim_{t \to \infty} p_0(t,0,n) = v(n)$$

where v(n) satisfies the algebraic equations

$$\left.\begin{array}{l} \sum_{k=0}^{J} v(k)\ A_0(k,n) = 0;\ n = 0,\ \ldots,\ J \\ \\ \sum_{k=0}^{J} v(k) = 1 \end{array}\right\} \qquad \text{(D.2-15)}$$

(Kannan 1979, p. 136). This condition is usually met in practice (i.e., that full-up is reachable from any other nonabsorbing state), and will be assumed to be true in what follows.

Write

$$\int_o^t p_0(t-y,0,n) f_{t_N}(y)dy$$

as

$$\int_o^\infty g_n(t-y) f_{t_N}(y)dy$$

where

$$g_n(t-y) = \begin{cases} 0;\ \text{if } y>t \\ p_0(t-y,0,n);\ \text{if } 0\le y\le t. \end{cases}$$

Since

$$\left| g_n(t-y) f_{t_N}(y) \right| \le f_{t_N}(y)$$

and f_{t_N} is integrable on $(0,\infty)$, it follows from the Lebesgue Dominated Convergence Theorem that

$$\lim_{t\to\infty} \int_0^t p_0(t-y,0,n) f_{t_N}(y)dy = \int_0^\infty \lim_{t\to\infty} g_n(t-y) f_{t_N}(y)dy$$

$$= \int_0^\infty v(n) f_{t_N}(y)dy = v(n),\ n = 0,\ 1,\ \ldots,\ J.$$

It has thus been shown that

$$\lim_{t\to\infty} P\{Y(t) = n|Y(0) = i\} = \pi(n) = v(n) \tag{D.2-16}$$

where v(n) is defined by (D.2-15).

From these considerations it is seen that "steady state" conditions under the baseline model entail the eventual complete debugging of the SW. This does not preclude the possibility of a SW failure due to some other cause instead of a bug, however. This situation can be modeled by including an SW component in the transition diagram which has constant failure/repair rate independent of the number of bugs in the SW.

To compute $R(\infty, t_o)$ (i.e. (3.3-8)) it is necessary to compute $P_m(t_o)$ (see (3.3-4) and (3.3-6)) under "steady state" conditions. The necessary expression is

$$R(\infty,\ t_o) = \Sigma\Sigma\, \pi(m)\ p_0(t_o, m.n) \tag{D.2-17}$$

where the double summation is taken over all (m,n) such that m and n are operational states in the transition diagram corresponding to A_0 (i.e. $X(t) = 0$), and where $p_0(t_o, m, n)$ is computed under the provision that failed states be made absorbing.

APPENDIX E

NUMERICAL/COMPUTATIONAL ASPECTS OF THE COMBINED HW/SW RELIABILITY MODEL

A technique for computing expression (D.2-12) will now be discussed. To implement this technique, it is necessary only to numerically solve systems of linear equations and linear differential equations. No integrations, as indicated in (D.2-12) are necessary.

In computing (D.2-12) the major difficulty is in computing

$$\int_o^t e^{-c(j-\ell-m)s}\ p_j(s,0,n)ds,\ n = 0,\ \dots,\ J.$$

For any admissible (j,ℓ,m) these quantities can be computed by solving a system of linear equations. To see this, define

$$\tilde{x}(t,j,\ell,m) \equiv \begin{bmatrix} \int_o^t e^{-c(j-\ell-m)s}\ p_j(s,0,0)ds \\ \int_o^t e^{-c(j-\ell-m)s}\ p_j(s,0,1)ds \\ \vdots \\ \int_o^t e^{-c(j-\ell-m)s}\ p_j(s,0,J)ds \end{bmatrix} \qquad (E.1\text{-}1)$$

and

$$\tilde{P}_j(t) \equiv \begin{bmatrix} p_j(t,0,0) \\ p_j(t,0,1) \\ \vdots \\ p_j(t,0,J) \end{bmatrix} \qquad (E.1\text{-}2)$$

With this notation, the Kolmogorov forward differential equations (see (B.1-7)) can be written as

$$\frac{d}{ds}\ \tilde{P}_j(s) = A_j^{'}\ \tilde{P}_j(s) \qquad (E.1\text{-}3)$$

where Aj is the infinitesimal matrix (see Appendix B) corresponding to X(t) = j, and ' denotes transpose. The initial conditions for (E.1-3) are pj(0,0,0) = 1, pj(0,0,j) = 0, j ≠ 0. Denote by 1 a column vector of 1's whose dimension will be clear from context. As usual, I will denote an identity matrix, the dimension of which will be clear from context.

Multiplying (E.1-3) by $e^{-c(j-\ell-m)s}$ and integrating from s = 0 to s = t gives

$$\int_o^t e^{-c(j-\ell-m)s} d\,\tilde{P}_j(s) = A_j' \int_o^t e^{-c(j-\ell-m)s}\,\tilde{P}_j(s)ds$$

and integration-by-parts on the left side yields

$$\tilde{P}_j(t)e^{-c(j-\ell-m)t} - \tilde{P}_j(0) = \left(A_j' - c(j-\ell-m)I\right)\int_o^t e^{-c(j-\ell-m)s}\,\tilde{P}_j(s)ds$$

or, writing

$$\tilde{b}(t,j,\ell,m) \equiv \tilde{P}_j(t)e^{-c(j-\ell-m)t} - \tilde{P}_j(0)$$

the linear system of equations (t is assumed fixed)

$$\left(A_j' - c(j-\ell-m)I\right)\tilde{x}(t,j,\ell,m) = \tilde{b}(t,j,\ell,m) \qquad \text{(E.1-4)}$$

is obtained. This system of equations is singular when $j-\ell-m = 0$ and overdetermined otherwise. These problems are caused by the fact that the components of the vector (E.1-1) must sum to 1 identically in t, i.e.,

$$1'\,\tilde{P}_j(t) \equiv 1.$$

Because of this,

$$1'\,\tilde{x}(t,j,\ell,m) = \int_o^t e^{-c(j-\ell-m)s}\,1'\,\tilde{P}_j(s)ds$$

$$= \int_o^t e^{-c(j-\ell-m)s}\,ds \equiv K(t,j,\ell,m)$$

where

$$K(t,j,\ell,m) = \begin{cases} t;\ \text{if } j-\ell-m = 0 \\ \dfrac{(1-e^{-c(j-\ell-m)t}}{c(j-\ell-m)};\ \text{if } j-\ell-m \neq 0. \end{cases} \tag{E.1-5}$$

The problems can be removed by reducing the order of the system (E.1-4) by directly imposing the relation $1'\tilde{x}(t,j,\ell,m) = K(t,j,\ell,m)$ in (E.1-4). To do this, the following notation is needed.

Let $\hat{A}(j,\ell,m)$ be the J x J matrix obtained from $A_j' - c(j-\ell-m)I$ by deleting the last row and last column, and let $H(j,\ell,m)$ be the J x 1 vector obtained from the last column of $A_j' - c(j-\ell-m)I$ by deleting the last element. Similarly, define $\tilde{y}(t,j,\ell,m)$ to be the J x 1 vector obtained from $\tilde{x}(t,j,\ell,m)$ by deleting the last element, and $\tilde{B}(t,j,\ell,m)$ will be the J x 1 vector obtained from $\tilde{b}(t,j,\ell,m)$ by deleting the last element. The new system then becomes

$$(\hat{A}(j,\ell,m) - H(j,\ell,m)1')\tilde{y}(t,j,\ell m) = \tilde{B}(t,j,\ell m) - K(t,j,\ell m)\ H(j,\ell,m). \tag{E.1-6}$$

If $\tilde{y}(t,j,\ell,m)$ solves (E.1-6), then $\tilde{x}(t,j,\ell,m)$ is given by

$$\tilde{x}(t,j,\ell,m) = \begin{bmatrix} \tilde{y}(t,j,\ell,m) \\ K(t,j,\ell,m) - 1'\ \tilde{y}(t,j,\ell,m) \end{bmatrix} \tag{E.1-7}$$

Thus, having obtained $\tilde{P}_j(t)$ for some fixed t's of interest bv solving the system (E.1-3) of linear differential equations, the vector (E.1-1) is computed by solving the system (E.1-6) and using (E.1-7). The state occupancy probabilities, i.e. (D.2-12), are then easily computed.

APPENDIX F

COMPUTER PROGRAM FOR CALCULATING STATE OCCUPANCY PROBABILITIES

A computer program has been developed for computing (D.2-12) (with $i = 0$). This program uses the numerical methods described in Appendix E and was used for computing the examples used in this study. A source listing is included in this appendix along with a detailed flow diagram.

The program is written in FORTRAN IV and designed to run on the IBM 370 or AMDAHL 470 computer. To use the program, it is necessary to change lines 70, 80, 90, and 120 in the main to reflect the values of λ, p, and J ($J \equiv NS$ in the program and is the number of states), respectively. The user supplies the value of NTIME which is the last time point (integer) at which the probabilities are computed. This value is read on unit 5.

In addition, the user must change lines 120 through 300 in subroutine SYS to reflect the infinitesimal matrix desired. The matrix A in this subroutine is related to the infinitesimal matrix A_j used previously in the following fashion (JJJ is the computer name for j in SYS):

$$\underset{\substack{\uparrow \\ \text{In subroutine SYS}}}{A(x+1,y+1)} \qquad = \qquad \underset{\substack{\uparrow \\ \text{Infinitesimal Matrix.}}}{A_j(y,x)}$$

Since the computer array cannot have a subscript value of 0, the state values are shifted one unit (e.g. state 0 is state 1 in the computer program).

A simplified flow chart for the program is included in Figure F-1. Figure F-2 is the detailed flow diagram.

A detailed error analysis was performed to determine the accuracy of the results obtained using this program. The program is not intended for use with N (number of SW bugs initially) large. How large N can be depends on how many time points (NTIME) are calculated. When N and NTIME become large underflows and overflows occur in the exponential function DEXP and in the computation of combinatorials used in D.2-12. Barring these difficulties, the error analysis showed that the program outputs had relative error less than an upper bound on the order of 10^{-10} for the cases considered. These cases included NS = 2, 3, 5, 7; NTIME = 5, 6, ..., 15, N = 5, 10, and values of repair rates on the order of 1 or 2, and SW/HW failure rates on the order of 0.001 to 1.0, and c $(=\lambda p) = 0.95$.

The major sources of error are in DSDIFF (numerical solution of differential equations) and LINEQ (linear equation solver). The algorithm used in DSDIFF is that of Bulirsch and Stoer from the journal Numerische Mathematik, vol 8 (1966) in an article entitled "Numerical Treatment of Ordinary Differential Equations by Extrapolation Methods." The relative error for the solution vectors is controlled to be less than 10^{-13} in this program (see line 160 in subroutine SOLVE). With this error reasonably controlled the next likely source is in LINEQ.

The algorithm used in LINEQ is Gauss-Jordan reduction and the error in using this technique (and any other technique) is dependent on how "close" the system of equations is to being singular. To quantify this, it is necessary to introduce matrix norms. The reference for this material is (Burden, et. al.).

Suppose the system of linear equations under consideration is

$$Ax = b \tag{F.1-1}$$

where

A is nxn, x is nx1, and b is nx1.

For an arbitrary matrix B of dimension pxq, define the norm of B as

$$\|B\| = \max_{1 \le i \le p} \sum_{j=1}^{q} |b_{ij}| \tag{F.1-2}$$

where

b_{ij} is the (i,j) element of B. What is needed is to find a bound for

$$\|x - \tilde{x}\| / \|x\|$$

where x is the exact solution of (F.1-1) and $\tilde{x}$ is the solution obtained from the Gauss-Jordan reduction algorithm.

When A is non-singular (as is the case in the program) it can be shown that

$$\|x - \tilde{x}\| / \|x\| \le \frac{\|A\| \, . \, \|A^{-1}\| \cdot \|b-A\tilde{x}\|}{\|b\|} \tag{F.1-3}$$

The quantity $K(A) \equiv \|A\| \cdot \|A^{-1}\|$ is called the "condition number" of the matrix A. The condition number is always greater than or equal to 1 and its magnitude measures how well the system of linear equations behaves in terms of numerical solution. For practical purposes compution of $\|A^{-1}\|$ is the same as solving the linear system and is subject to as much or more computational inaccuracy. To make computation of the condition number practical, the following technique can be used.

The first step is to compute $\tilde{x}$ using, say, t-digit arithmetic and Gauss-Jordan reduction. The residual vector $b-A\tilde{x}$ is then computed in 2t-digit arithmetic. Gauss-Jordan elimination is then applied to the system $A\tilde{y} = (b-A\tilde{x})$ to yield the solution $\tilde{y}$. The approximate value of K(A) is then

$$K(A) \approx \frac{\|\tilde{y}\|}{\|\tilde{x}\|} \cdot 10^t \qquad \text{(F.1-4)}$$

yielding

$$\|x - \tilde{x}\| / \|x\| \leq 10^t \frac{\|\tilde{y}\|}{\|\tilde{x}\|} \cdot \frac{\|b-A\tilde{x}\|}{\|b\|} . \qquad \text{(F.1-5)}$$

These techniques were applied to the computer program listed in this appendix by adding an extended precision subroutine (i.e. REAL*16) to perform Gauss-Jordan elimination and hence perform the necessary 2t-digit calculations. The maximum relative error (over all calls to LINEQ) was on the order of 10^{-10}. This combined with the controlled error in the subroutine DSDIFF indicated excellent precision for the cases mentioned earlier. The user of this program should be warned, however, that gross computational errors will occur if NTIME and/or N are too large.

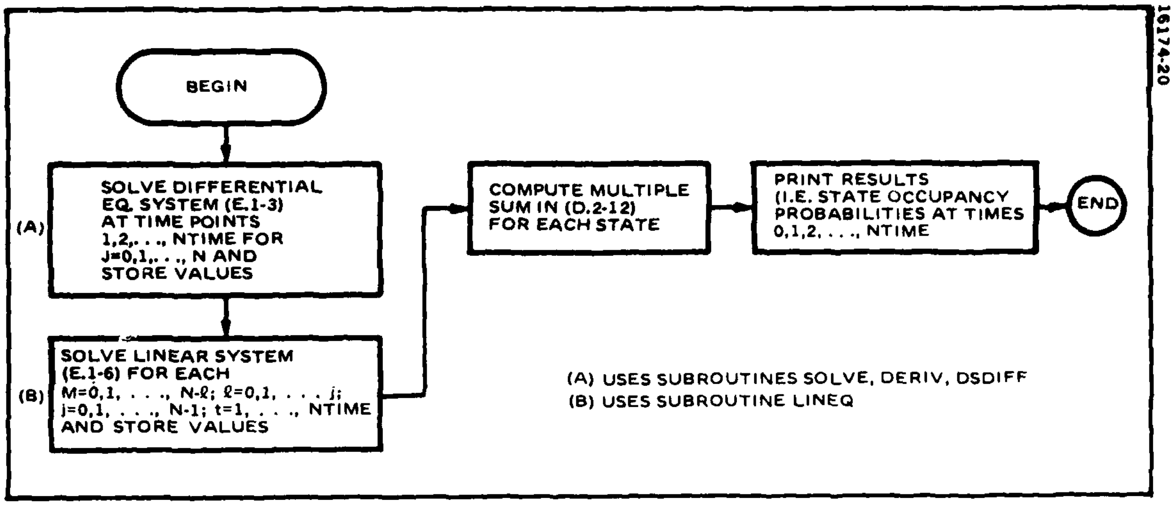

BEGIN
(A)
SOLVE DIFFERENTIAL EQ. SYSTEM (E.1-3) AT TIME POINTS 1,2,. . ., NTIME FOR J=0,1,. . ., N AND STORE VALUES
(B)
SOLVE LINEAR SYSTEM (E.1-6) FOR EACH M=0,1, . . ., N-ℓ; ℓ=0,1, . . . j; j=0,1, . . ., N-1; t=1, . . ., NTIME AND STORE VALUES
COMPUTE MULTIPLE SUM IN (D.2-12) FOR EACH STATE
PRINT RESULTS (I.E. STATE OCCUPANCY PROBABILITIES AT TIMES 0,1,2, . . ., NTIME
END
(A) USES SUBROUTINES SOLVE, DERIV, DSDIFF
(B) USES SUBROUTINE LINEQ
16174-20

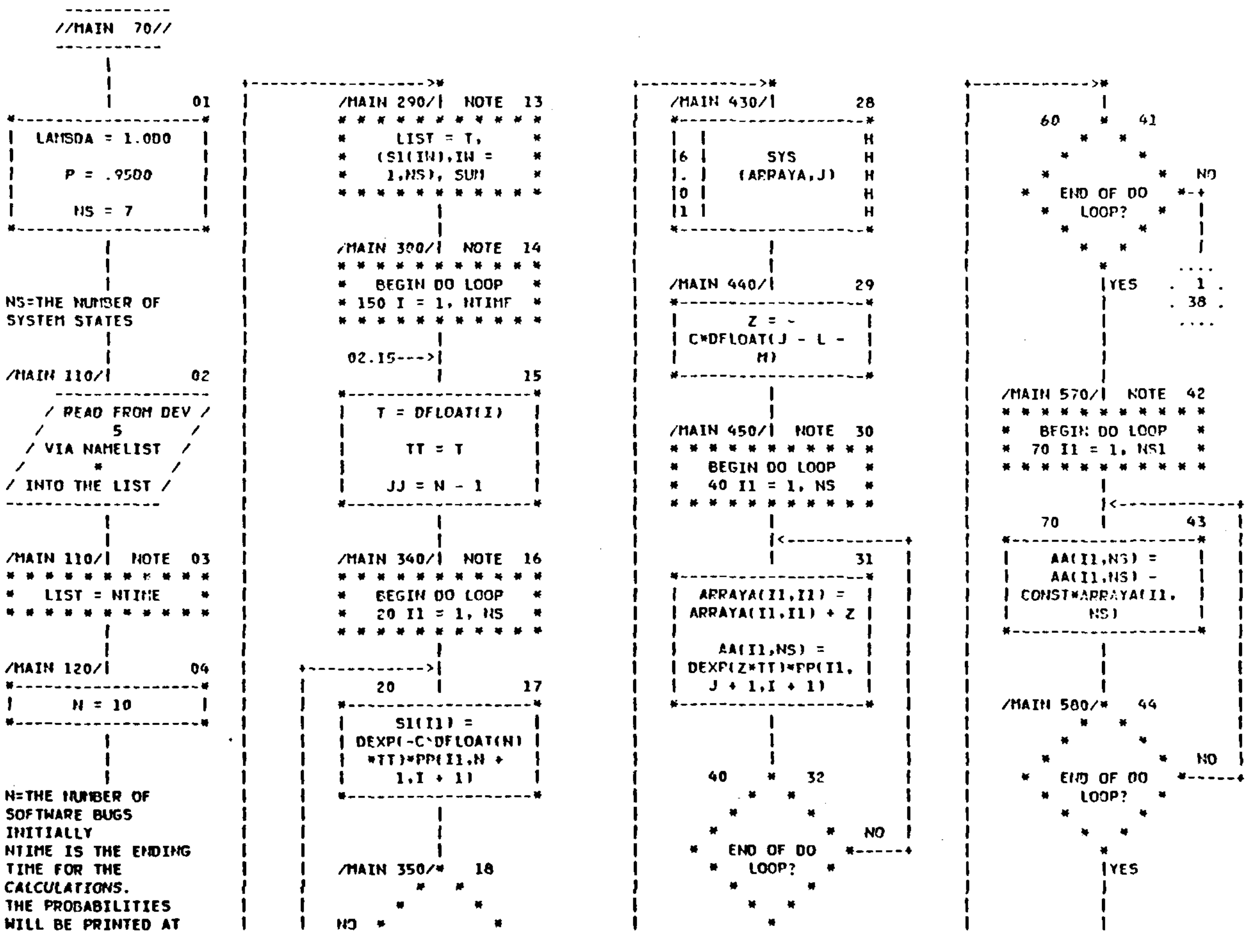
Figure F-1. Simplified Flow Chart for Computer Program

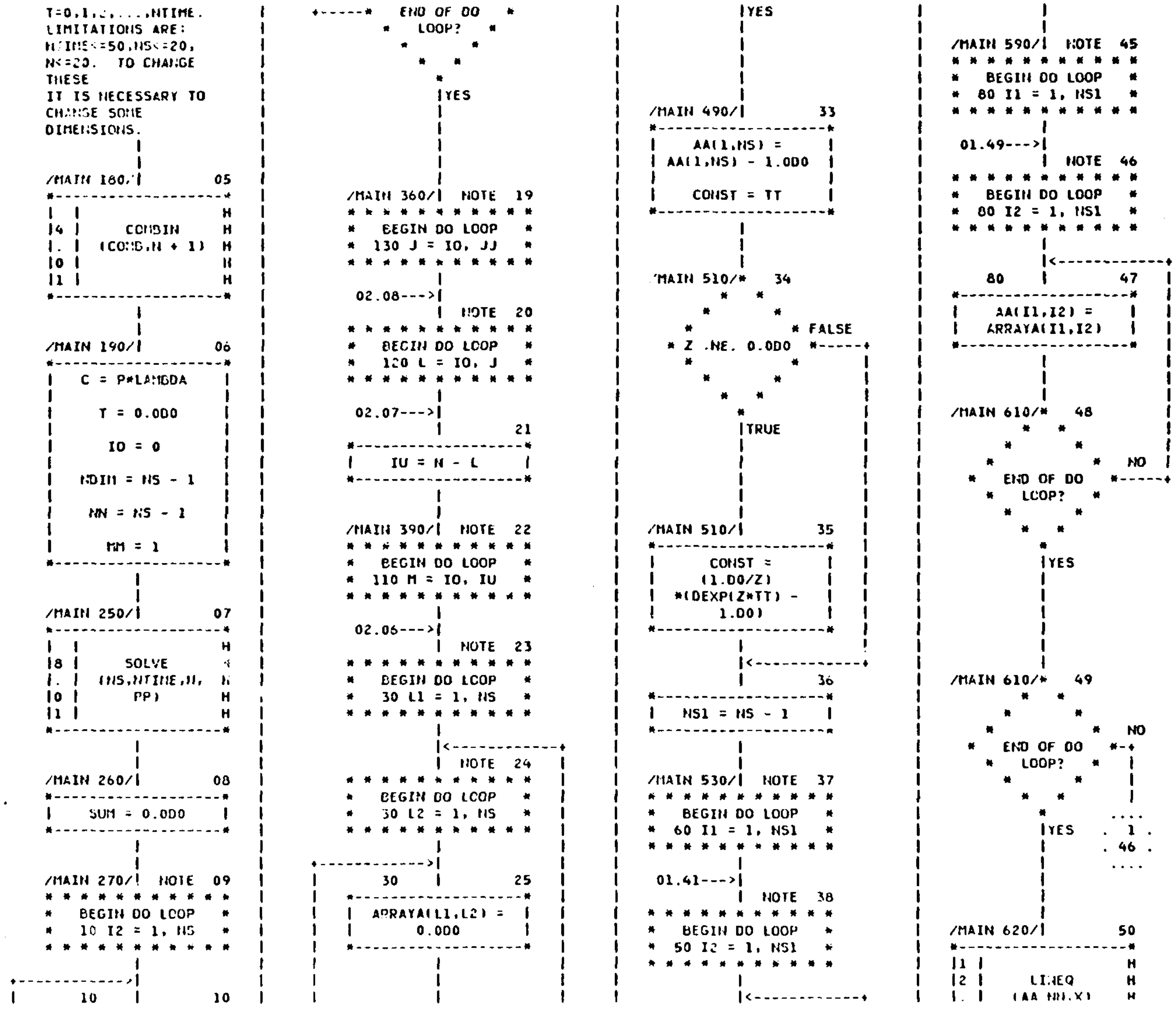
Figure F-2. Detailed Flow Diagram

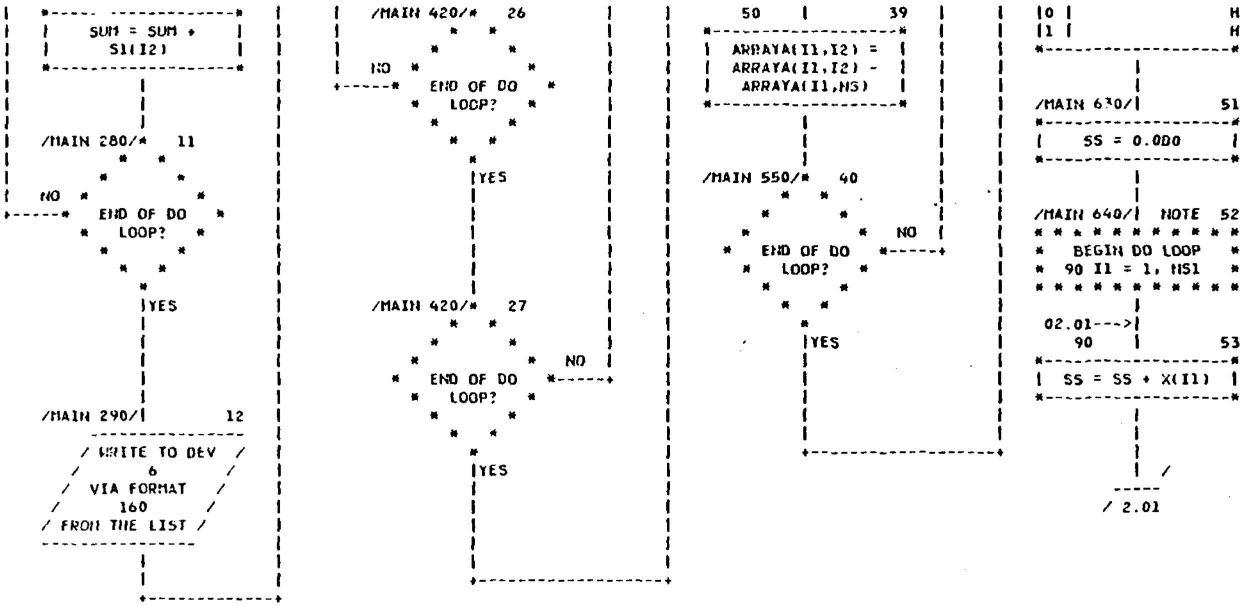
Each consecutive three pages, placed bottom to top, make up one page in the complete flow diagram. The reference numbers at the upper right corner of each box/triangle is a sequence number to identify the box/triangle. The numbers A.B refer to page A, box/triangle number B. For example, 4.01 is page 4, box 1. Page numbers are placed at the upper right corner on the first of each three-page set.

01/30/81 PAGE 01

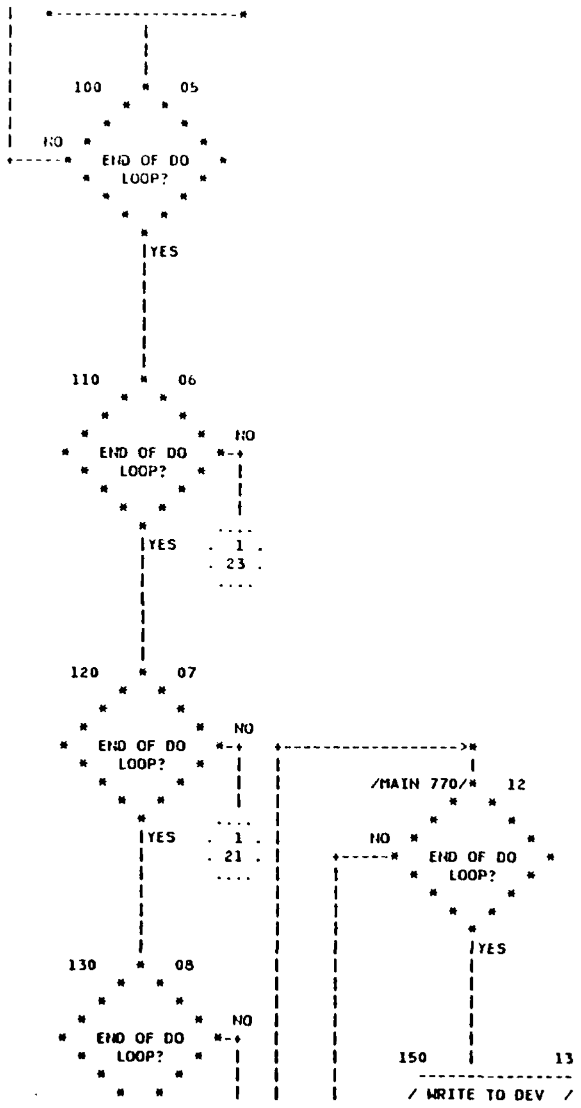
CHART TITLE - PROCEDURES


//MAIN 70//
01
LAMBDA = 1.000
P = .9500
NS = 7
NS=THE NUMBER OF SYSTEM STATES
/MAIN 110/ 02
READ FROM DEV 5 VIA NAMELIST * INTO THE LIST
/MAIN 110/ NOTE 03
LIST = NTIME
/MAIN 120/ 04
N = 10
N=THE NUMBER OF SOFTWARE BUGS INITIALLY
NTIME IS THE ENDING TIME FOR THE CALCULATIONS.
THE PROBABILITIES WILL BE PRINTED AT
/MAIN 290/ NOTE 13
LIST = T, (S1(IN),IN = 1,NS), SUM
/MAIN 300/ NOTE 14
BEGIN DO LOOP 150 I = 1, NTIME
02.15--->
15
T = DFLOAT(I)
TT = T
JJ = N - 1
/MAIN 340/ NOTE 16
BEGIN DO LOOP 20 I1 = 1, NS
20
17
S1(I1) = DEXP(-C*DFLOAT(N)*TT)*PP(I1,N + 1,I + 1)
/MAIN 350/ 18
NO
/MAIN 430/ 28
6.01
SYS (ARRAYA,J)
/MAIN 440/ 29
Z = -C*DFLOAT(J - L - M)
/MAIN 450/ NOTE 30
BEGIN DO LOOP 40 I1 = 1, NS
31
ARRAYA(I1,I1) = ARRAYA(I1,I1) + Z
AA(I1,NS) = DEXP(Z*TT)*PP(I1, J + 1,I + 1)
40 32
END OF DO LOOP?
NO
60 41
END OF DO LOOP?
NO
YES
1
38
/MAIN 570/ NOTE 42
BEGIN DO LOOP 70 I1 = 1, NS1
70 43
AA(I1,NS) = AA(I1,NS) - CONST*ARRAYA(I1, NS)
/MAIN 580/ 44
END OF DO LOOP?
NO
YES


T=0,1,2,....,NTIME.
LIMITATIONS ARE:
NTIMES<=50,NS<=20,
N<=20. TO CHANGE
THESE
IT IS NECESSARY TO
CHANGE SOME
DIMENSIONS.
/MAIN 180/ 05
4.01
CONDIN
(COND,N + 1)
/MAIN 190/ 06
C = P*LAMBDA
T = 0.0D0
IO = 0
NDIM = NS - 1
NN = NS - 1
MM = 1
/MAIN 250/ 07
8.01
SOLVE
(NS,NTIME,N,
PP)
/MAIN 260/ 08
SUM = 0.0D0
/MAIN 270/ NOTE 09
BEGIN DO LOOP
10 I2 = 1, NS
10 10
END OF DO
LOOP?
YES
/MAIN 360/ NOTE 19
BEGIN DO LOOP
130 J = IO, JJ
02.08--->
NOTE 20
BEGIN DO LOOP
120 L = IO, J
02.07--->
21
IU = N - L
/MAIN 390/ NOTE 22
BEGIN DO LOOP
110 M = IO, IU
02.06--->
NOTE 23
BEGIN DO LOOP
30 L1 = 1, NS
NOTE 24
BEGIN DO LOOP
30 L2 = 1, NS
30 25
ARRAYA(L1,L2) =
0.0D0
YES
/MAIN 490/ 33
AA(1,NS) =
AA(1,NS) - 1.0D0
CONST = TT
/MAIN 510/ 34
Z .NE. 0.0D0
FALSE
TRUE
/MAIN 510/ 35
CONST =
(1.D0/Z)
*(DEXP(Z*TT) -
1.D0)
36
NS1 = NS - 1
/MAIN 530/ NOTE 37
BEGIN DO LOOP
60 I1 = 1, NS1
01.41--->
NOTE 38
BEGIN DO LOOP
50 I2 = 1, NS1
/MAIN 590/ NOTE 45
BEGIN DO LOOP
80 I1 = 1, NS1
01.49--->
NOTE 46
BEGIN DO LOOP
80 I2 = 1, NS1
80 47
AA(I1,I2) =
ARRAYA(I1,I2)
/MAIN 610/ 48
END OF DO
LOOP?
NO
YES
/MAIN 610/ 49
END OF DO
LOOP?
NO
YES
1
46
/MAIN 620/ 50
12.
LINEQ
(AA NN,X)


SUM = SUM + S1(I2)
/MAIN 280/ 11
END OF DO LOOP?
NO
YES
/MAIN 290/ 12
WRITE TO DEV 6 VIA FORMAT 160 FROM THE LIST
/MAIN 420/ 26
END OF DO LOOP?
NO
YES
/MAIN 420/ 27
END OF DO LOOP?
NO
YES
50 39
ARRAYA(I1,I2) = ARRAYA(I1,I2) - ARRAYA(I1,N3)
/MAIN 550/ 40
END OF DO LOOP?
NO
YES
10
11
H
H
/MAIN 630/ 51
SS = 0.0D0
/MAIN 640/ NOTE 52
BEGIN DO LOOP 90 I1 = 1, NS1
02.01--->
90 53
SS = SS + X(I1)
2.01

```
  01.53--->*
           |
/MAIN 650/*    01
        *   *
      *       *
    *           *   NO
  *   END OF DO   *-+
    *   LOOP?   *   |
      *       *     |
        *   *       |
          *        ....
          |YES    . 1 .
          |       . 53 .
          |        ....
          |
          |
          |
/MAIN 660/|          02
*-------------------*
|   X(NS) = CONST - |
|        SS         |
*-------------------*
          |
          |
          |
WRITE(6,*)
(X(LL),LL=1,NS),I,J,
L,M
          |
          |
/MAIN 680/|  NOTE   03
* * * * * * * * * * *
*   BEGIN DO LOOP   *
*   100 K = 1, NS   *
* * * * * * * * * * *
          |
+------------>|
|             |          04
|   *-------------------*
|   |  S1(K) = S1(K) +  |
|   |  COMB(N + 1,L +   |
|   |   1)*COMB(IU +    |
|   |       1,M +       |
|   |  1)*((-1)**(M +   |
|   |    1))*C*(L +     |
|   |  M)*DEXP(-C*(L +  |
|   |    M)*TT)*X(K)    |
```


100
05
NO
END OF DO LOOP?
YES
110
06
NO
END OF DO LOOP?
YES
1
23
120
07
NO
END OF DO LOOP?
YES
1
21
/MAIN 770/
12
NO
END OF DO LOOP?
YES
130
08
NO
END OF DO LOOP?
150
13
/ WRITE TO DEV /

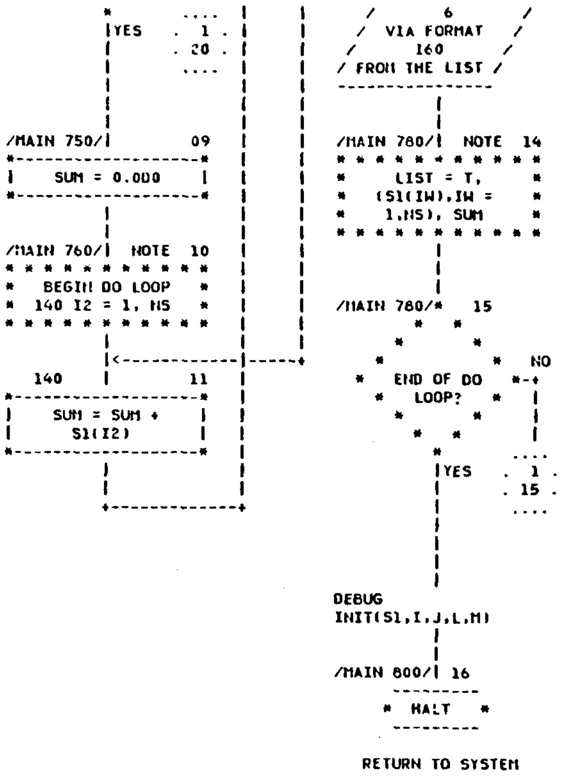
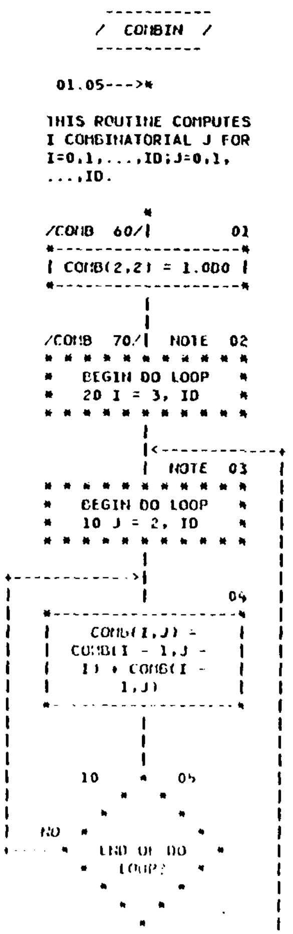

YES
1
20
/MAIN 750/ 09
SUM = 0.000
/MAIN 760/ NOTE 10
BEGIN DO LOOP
140 I2 = 1, NS
140 11
SUM = SUM +
S1(I2)
6
VIA FORMAT
160
FROM THE LIST
/MAIN 780/ NOTE 14
LIST = T,
(S1(IW),IW =
1,NS), SUM
/MAIN 780/ 15
END OF DO
LOOP?
NO
YES
1
15
DEBUG
INIT(S1,I,J,L,M)
/MAIN 800/ 16
HALT
RETURN TO SYSTEM

CHART TITLE - SUBROUTINE COMBIN(COMB,ID)


/ COMBIN /
01.05--->*
THIS ROUTINE COMPUTES I COMBINATORIAL J FOR I=0,1,...,ID;J=0,1,...,ID.
/COMB 60/
01
COMB(2,2) = 1.000
/COMB 70/
NOTE 02
BEGIN DO LOOP 20 I = 3, ID
NOTE 03
BEGIN DO LOOP 10 J = 2, ID
04
COMB(I,J) = COMB(I - 1,J - 1) + COMB(I - 1,J)
10
05
NO
END OF DO LOOP?

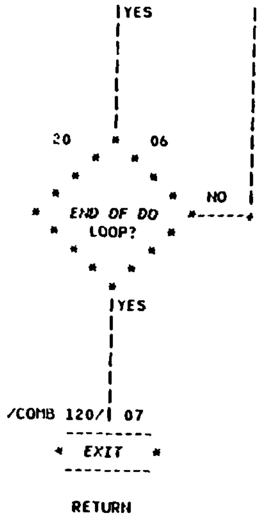

YES
20
06
END OF DO LOOP?
NO
YES
/COMB 120/
07
EXIT
RETURN

CHART TITLE - SUBROUTINE SYS(A,JJJ)

```
                ------------
              /   SYS    /
               -----------

         01.28*-->*

        THIS ROUTINE SUPPLIES
        THE MATRIX AJ
        (I.E. THE
        INFINITESIMAL MATRIX.
        SEE FINAL REPORT) FOR
        VALUES OF
        J=0,1,2,...,N.
        WHEN MCDIFYING THIS
        ROUTINE FOR SPECIFIC
        CASES THIS
        ROUTINE USES A IN
        TRANSPOSED FORM.  SO,
        FOR EXAMPLE,
        THE (I,J) ENTRY IN
        THE INFINITESIMAL
        MATRIX AS DESCRIBED
        IN THE FINAL REPORT
        IS ENTERED IN THIS
        ROUTINE AS THE
        (J,I) ELEMENT OF A.

                  *
        /SYS  120/I         01
        *--------------------*
        I     A(1,1) = -     I
        I   (.05D0*DFLOAT    I
        I   (JJJ) + .02D0)   I
        I                    I
        I      A(2,1) =      I
        I  .05D0*DFLOAT(JJJ) I
        I                    I
        I  A(3,1) = .02D0    I
        *--------------------*
                  I
                  I
        /SYS  150/I         02
        *--------------------*
        I   A(1,2) = 1.D0    I
        I                    I
        I  A(2,2) = - 1.D0   I
        I                    I
        I   A(1,3) = 1.D0    I
        *--------------------*
                  I
                  I
```

```
/SYS  180/|            03
*---------------------*
|     A(3,3) = -      |
|    (1.018D0 +       |
| .045*DFLOAT(JJJ))   |
|                     |
|      A(4,3) =       |
|   .045D0*DFLOAT     |
|       (JJJ)         |
|                     |
|  A(5,3) = .018D0    |
*---------------------*
           |
           |
/SYS  210/|            04
*---------------------*
|   A(3,4) = 1.D0     |
|                     |
|  A(4,4) = - 1.D0    |
|                     |
|   A(3,5) = 1.D0     |
*---------------------*
           |
           |
/SYS  240/|            05
*---------------------*
|     A(5,5) = -      |
|    (1.016D0 +       |
| .04D0*DFLOAT(JJJ)   |
|          )          |
|                     |
|      A(6,5) =       |
| .04D0*DFLOAT(JJJ)   |
|                     |
|  A(7,5) = .016D0    |
*---------------------*
           |
           |
/SYS  270/|            06
*---------------------*
|   A(5,6) = 1.D0     |
|                     |
|  A(6,6) = - 1.D0    |
|                     |
|   A(5,7) = 1.D0     |
|                     |
|  A(7,7) = - 1.D0    |
*---------------------*
           |
           |
           |
A(1,1)
=-(.004*UFLOAT(JJJ))
A(1,2)=2.D0
A(1,3)=2.D0
A(2,1)=DFLOAT(JJJ)
A(2,2)=-2.D0
A(3,1)=.004D0
A(3,3)=-2.D0
           |
             |
/SYS  380/| 07
     ---------
    *  EXIT   *
     ---------

      RETURN
```

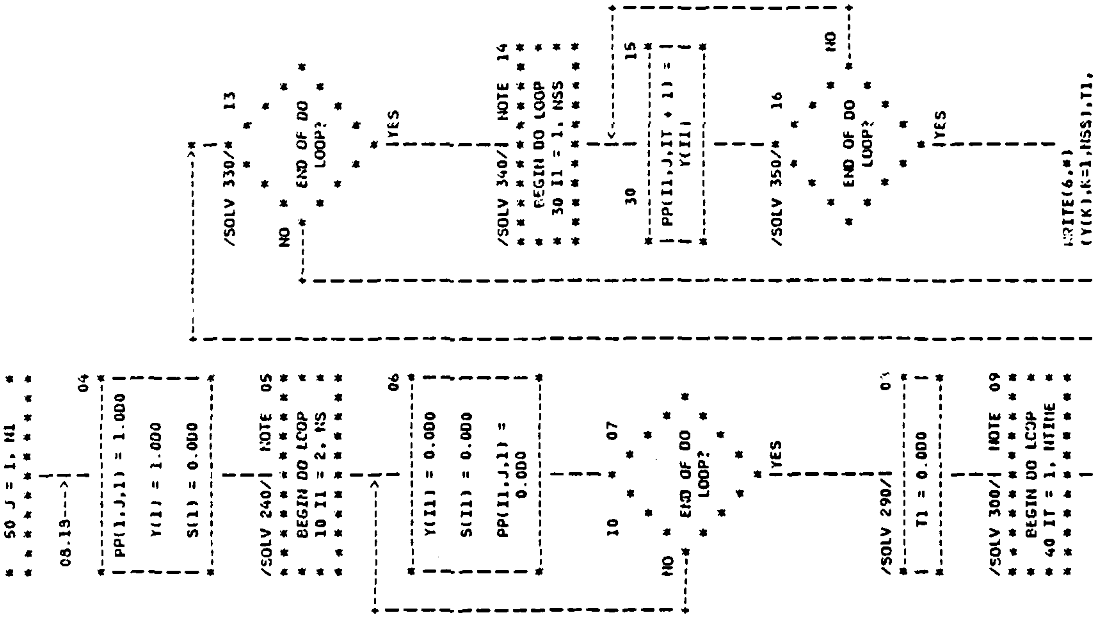
```
CHART TITLE - SUBROUTINE  SOLVE(NS,NTIME,N,PP)


                 -----------
                /  SOLVE   /
                -----------

              01.07--->*

             THIS ROUTINE SOLVES
             THE SYSTEM OF
             DIFFERENTIAL
             EQUATIONS (KOLMOCOROV
             EQUATIONS) FOR EACH J
             AND AT
             EACH POINT IN TIME.
             THE MATRIX PP(I,J,IT)
             UPON
             EXIT WILL CONTAIN THE
             SOLUTION FOR STATE I,
             WHEN
             X(T)=J, AT THE POINT
             T=IT-1), I.E. IN THE
             NOTATION
             OF THE REFORT,
             PJ(IT-1,0,I). IT IS
             ASSUMED THAT
             Y(0)=0 WITH
             PROBABILITY 1.

                       *
             /SOLV 140/|          01
             *-------------------*
             |      H = .100     |
             |                   |
             | II = 1.D0/H + .1  |
             |                   |
             |  EPS = 1.0 - 13   |
             |                   |
             |     NSS = NS      |
             |                   |
             |     N1 = N + 1    |
             *-------------------*
                       |
                       |
             /SOLV 190/|          02
             *-------------------*
             |  N2 = NTIME + 1   |
             *-------------------*
                       |
                       |
             /SOLV 200/|  NOTE  03
             * * * * * * * * * * *
             *   BEGIN DO LOOP   *
```


50 J = 1, N1
08.13--->
04
PP(1,J,1) = 1.000
Y(1) = 1.000
S(1) = 0.000
/SOLV 240/ NOTE 05
BEGIN DO LOOP
10 I1 = 2, NS
06
Y(I1) = 0.000
S(I1) = 0.000
PP(I1,J,1) = 0.000
10
07
END OF DO LOOP?
NO
YES
/SOLV 290/
08
T1 = 0.000
/SOLV 300/ NOTE 09
BEGIN DO LOOP
40 IT = 1, NTIME
/SOLV 330/
13
END OF DO LOOP?
NO
YES
/SOLV 340/ NOTE 14
BEGIN DO LOOP
30 I1 = 1, NSS
30
15
PP(I1,J,IT + 1) = Y(I1)
/SOLV 350/
16
END OF DO LOOP?
NO
YES
WRITE(6,*)
(Y(K),K=1,NSS),T1,

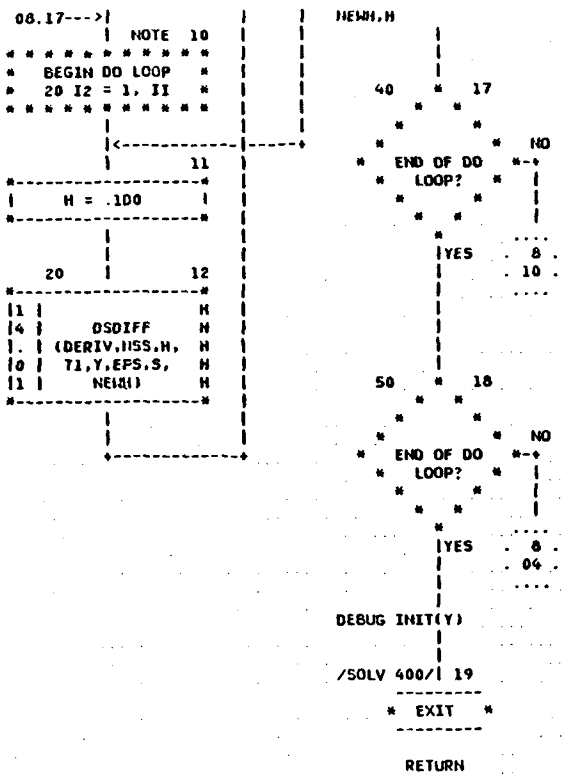

08.17--->
NOTE 10
BEGIN DO LOOP
20 I2 = 1, II
11
H = .100
20
12
1
4
.
0
1
DSDIFF
(DERIV,NSS,H,
T1,Y,EPS,S,
NEW(I)
NEWH,H
40
17
END OF DO
LOOP?
NO
YES
8
10
50
18
END OF DO
LOOP?
NO
YES
8
04
DEBUG INIT(Y)
/SOLV 400/
19
EXIT
RETURN

CHART TITLE - SUBROUTINE DERIV(X,Y,DY)

```
              ----------
             /  DERIV   /
            ----------

                 *

    THIS ROUTINE COMPUTES
    THE DERIVATIVES
    REQUIRED BY
    DSDIFF.

              *
    /DERI  70/|        01
    *--------------------*
    |      JJ = J        |
    *--------------------*
              |
              |
    /DERI  80/|        02
    *--------------------*
    |  |                 H
    |6 |     SYS         H
    |. |  (A,JJ - 1)     H
    |0 |                 H
    |1 |                 H
    *--------------------*
              |
              |
    /DERI  90/| NOTE  03
    * * * * * * * * * * *
    *  BEGIN DO LOOP    *
    *  10 I1 = 1, NSS   *
    * * * * * * * * * * *
              |
              |<-------------+
              |        04    |
    *--------------------*   |
    |  DY(I1) = 0.000    |   |
    *--------------------*   |
              |              |
              |              |
    /DERI 110/| NOTE  05     |
    * * * * * * * * * * *    |
    *  BEGIN DO LOOP    *    |
    *  10 I2 = 1, NSS   *    |
    * * * * * * * * * * *    |
              |              |
+------------>|              |
|      10     |        06    |
|   *--------------------*   |
|   | DY(I1) = DY(I1) +  |   |
```

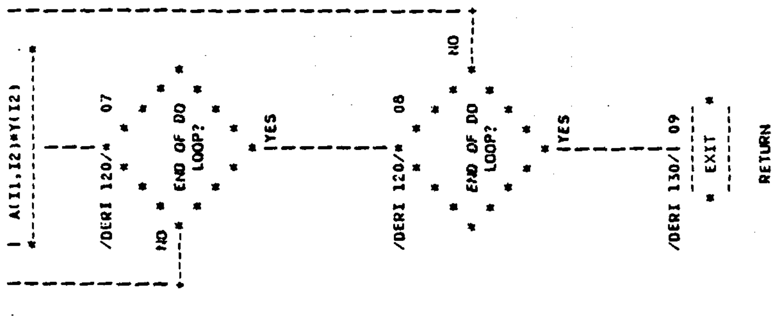
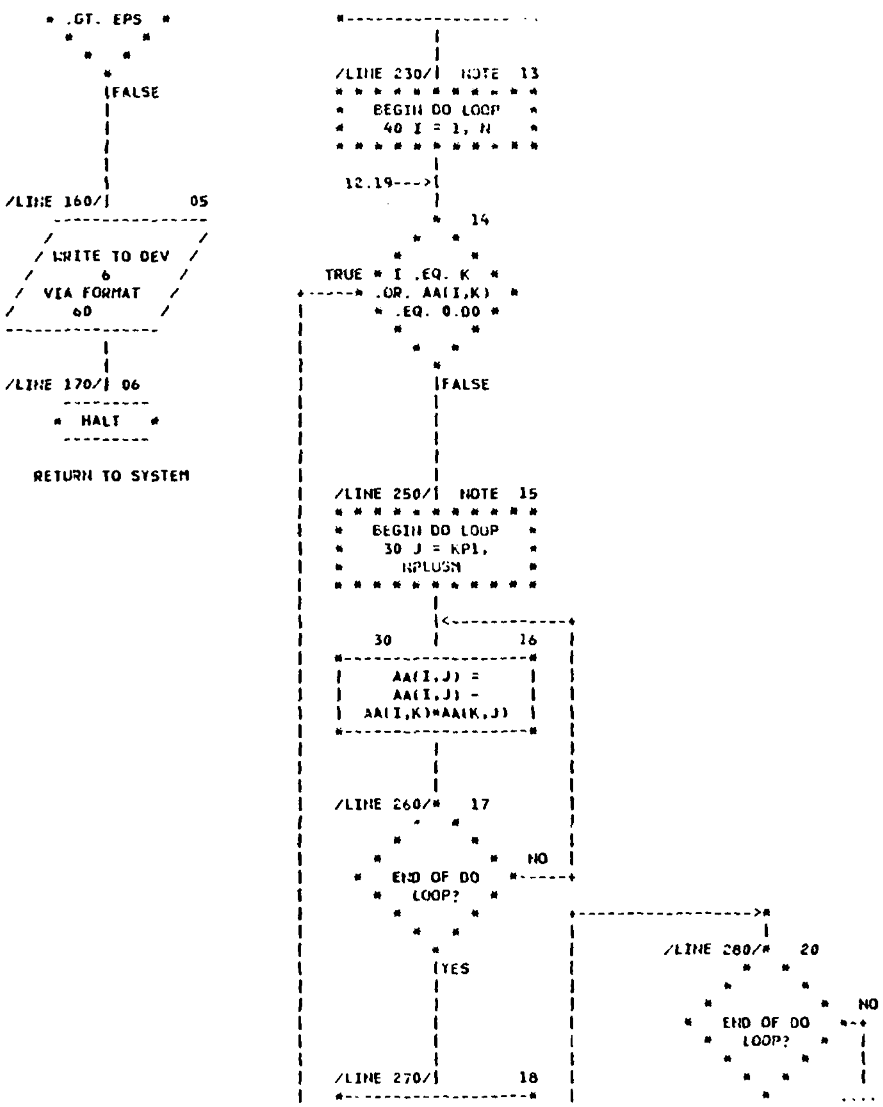

A(I1,I2)*Y(I2)
/DERI 120/* 07
END OF DO LOOP?
NO
YES
/DERI 120/* 08
END OF DO LOOP?
NO
YES
/DERI 130/I 09
EXIT
RETURN

CHART TITLE - SUBROUTINE LINEQ(AA,NN,X)

```
      -----------                       -----------
     / LINEQ    /                      /    10    /
     -----------                       -----------
                                            |
 01.50--->*             +------------------>|
                        |                   | NOTE  07
THIS ROUTINE SOLVES     |         * * * * * * * * * * *
THE SYSTEM OF LINEAR    |         *     CONTINUE      *
EQUATIONS A*X=B WHERE   |         * * * * * * * * * * *
A IS THE ARRAY          |                   |
CONSISTING OF           |                   |
THE FIRST  NN COLUMNS   |         /LINE 190/|         08
OF AA AND B IS THE      |         *-------------------*
LAST COLUMN             |         |    KP1 = K + 1    |
OF AA.                  |         *-------------------*
                        |                   |
          *             |                   |
/LINE  80/|         01  |         /LINE 200/|  NOTE  09
*-------------------*   |         * * * * * * * * * * *
|       M = 1       |   |         *   BEGIN DO LOOP   *
|                   |   |         *    20 J = KP1,    *
|      N = NN       |   |         *      NPLUSM       *
|                   |   |         * * * * * * * * * * *
|    EPS = 1.D - 4  |   |                   |
|                   |   |     +------------>|
|   NPLUSM = N + M  |   |     |      20     |         10
|                   |   |     |   *-------------------*
|    DETER = 1.D0   |   |     |   |     AA(K,J) =     |
*-------------------*   |     |   |  AA(K,J)/AA(K,K)  |
          |             |     |   *-------------------*
          |             |     |             |
/LINE 130/|  NOTE  02   |     |             |
* * * * * * * * * * *   |     |             |
*   BEGIN DO LOOP   *   |     |   /LINE 210/*    11
*    40 K = 1, N    *   |     |           *   *
* * * * * * * * * * *   |     |         *       *
          |             |     |  NO   *           *
 12.20--->|             |     +-----*   END OF DO   *
          |         03  |             *   LOOP?   *
*-------------------*   |               *       *
|      DETER =      |   |                 *   *
|   DETER*AA(K,K)   |   |                   *
*-------------------*   |                   |YES
          |             |                   |
          |             |                   |
          |             |                   |
/LINE 150/*    04       |                   |
        *   *           |                   |
      *       *         |         /LINE 220/|         12
    *           *  TRUE |         *-------------------*
  * DABS(AA(K,K)) * - - +         |  AA(K,K) = 1.D0   |
```


.GT. EPS
FALSE
/LINE 160/ 05
WRITE TO DEV 6 VIA FORMAT 60
/LINE 170/ 06
HALT
RETURN TO SYSTEM
/LINE 230/ NOTE 13
BEGIN DO LOOP 40 I = 1, N
12.19--->
14
TRUE
I .EQ. K .OR. AA(I,K) .EQ. 0.00
FALSE
/LINE 250/ NOTE 15
BEGIN DO LOUP 30 J = KP1, NPLUSM
30
16
AA(I,J) = AA(I,J) - AA(I,K)*AA(K,J)
/LINE 260/ 17
END OF DO LOOP?
NO
YES
/LINE 270/ 18
/LINE 280/ 20
END OF DO LOOP?
NO

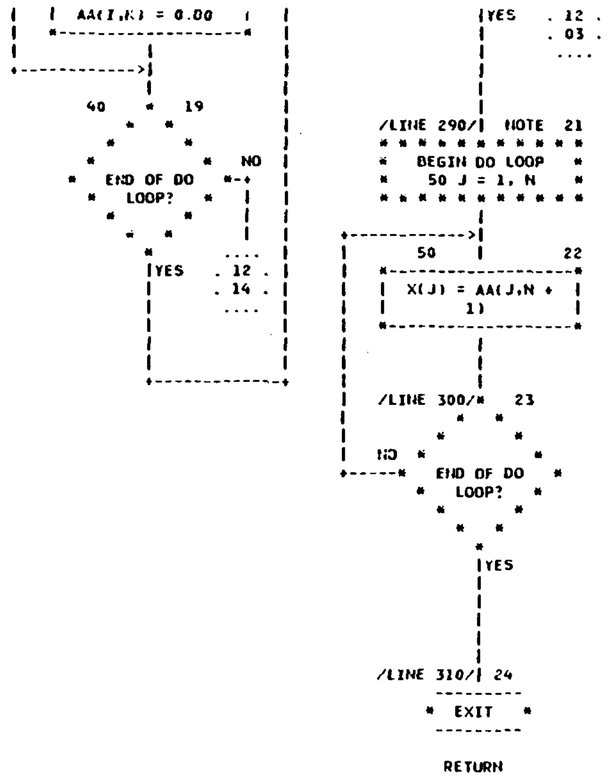
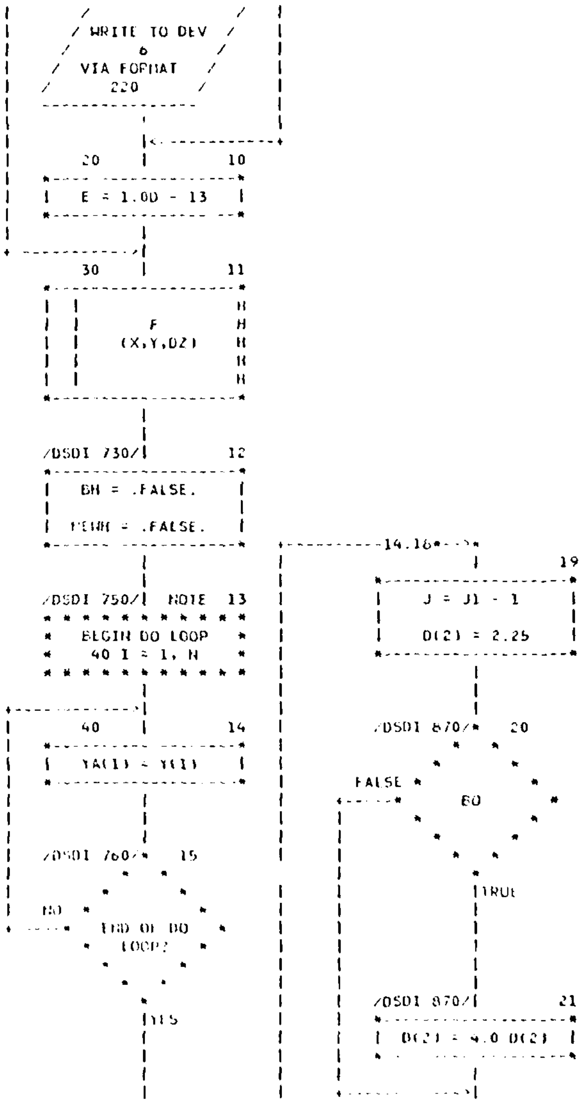

AA(I,K) = 0.00
40
19
END OF DO LOOP?
NO
YES
12
14
YES
12
03
/LINE 290/
NOTE 21
BEGIN DO LOOP
50 J = 1, N
50
22
X(J) = AA(J,N + 1)
/LINE 300/
23
NO
END OF DO LOOP?
YES
/LINE 310/
24
EXIT
RETURN

CHART TITLE - SUBROUTINE DSDIFF(F,N,H,X,Y,EPS,S,NEWH)

```
      -----------                                                           -----------
     /  DSDIFF  /                                                          /    10    /
     -----------                                                           -----------
                                                                                |
 08.12--->*                                                  *----------------->|
                                                             |                  |        05
F IS THE NAME OF A                 CALLING PROGRAM, THE      |        *-------------------*
SUBROUTINE CALLED BY               ARRAY S WILL HAVE HAD     |        |  E = DABS(EPS)    |
'CALL F(X,Z,DZ)'                   ITS CONTENTS MODIFIED     |        *-------------------*
WHICH                              SO THAT                   |                  |
STORES IN THE VECTOR               S(I)=MAX(S(I),            |                  |
DZ THE N COMPONENTS                ABS(Y(I,X)), WHERE        |                  |
OF THE DERIVATIVE                  THE MAXIMUM IS TAKEN      |        /DSDI 660/*    06
DZ/DX ACCORDING TO                 OVER                      |                *   *
THE DIFFERENTIAL                   THE INTEGRATION           |              *       *
EQUATION WHICH IS                  INTERVAL (X,X+H').        |      TRUE  *           *
BEING                                                        |     *-----* E .GE. 1.0D - *
SOLVED, DZ/DX=F(X,Z).                      *                 |     |      *    13     *
X, Z, AND DZ MUST BE                       |                 |     |        *       *
DOUBLE PRECISION.                  /DSDI 620/*    02         |     |          *   *
                                         *   *               |     |            *
                                       *       *             |     |            |FALSE
                                     * N .GT. 0  *  TRUE     |     |            |
N IS THE ORDER OF THE              * .AND. N .LE.  *---------*     |            |
SYSTEM OF                            *  MAXORD   *                 |            |
DIFFERENTIAL                           *       *                   |            |
EQUATIONS.                               *   *                     |            |
N MUST BE NO GREATER                       *                       |            |
THAN MAXORD WHICH IS                       |FALSE                  |  /DSDI 670/*    07
SET BELOW.                                 |                       |          *   *
                                           |                       |        *       *
                                           |                       |      *           *  TRUE
                                           |                       |    *    EPSERR     *-----*
H IS THE BASIC STEP                        |                       |      *           *       |
SIZE.                              /DSDI 630/|        03           |        *       *         |
                                  ----------------                 |          *   *           |
                                  /              /                 |            *             |
                                 / WRITE TO DEV /                  |            |FALSE        |
X AND Y(VECTOR) ARE             /       6      /                   |            |             |
THE INITIAL VALUES.            /  VIA FORMAT  /                    |            |             |
                              /      210     /                     |            |             |
                              ---------  ------                    |            |             |
                                         |                         |            |             |
EPS AND S(VECTOR) ARE                    |                         |  /DSDI 680/|        08   |
THE ERROR BOUNDS.                  /DSDI 640/| 04                  |  *-------------------*   |
DABS(EPS) SHOULD BE                   ---------                    |  | EPSERR = .TRUE.   |   |
NO SMALLER THAN                       *  HALT   *                  |  *-------------------*   |
1.0D-13.                              ---------                    |            |             |
                                                                   |            |             |
                                     RETURN TO SYSTEM              |  /DSDI 690/|        09   |
                                                                   |   ----------------       |
NEWH . . A FLAG WHICH
IS SET EQUAL TO
.TRUE. IF THE STEP
SIZE USED
USED BY OSDIFF IS
DIFFERENT FROM THE
STEP SIZE H GIVEN IN
THE
PARAMETER LIST.  NEWH
IS SET EQUAL TO
.FALSE. OTHERWISE.


EACH CALL OF OSDIFF
PERFORMS ONE
INTEGRATION STEP OF
THE
EQUATION DY/DX=F(X,Y)
ACCORDING TO THE
METHOD OF R. BULIRSCH
AND
J. STOER (NUMERISCHE
MATHEMATIK, IN
PRESS).  THE STEP
SIZE WILL
BE LESS THAN OR EQUAL
TO H.  THE PROGRAM
TAKES THE FIRST OF
THE
NUMBERS H, H/2, H/4,
...., AS STEP SIZE FOR
WHICH NO MORE THAN 9
EXTRAPOLATION STEPS
ARE NEEDED TO OBTAIN
A SUFFICIENTLY
ACCURATE
RESULT.  IF THE STEP
SIZE USED IS
DIFFERENT THAN THE
STEP SIZE
GIVEN IN THE
PARAMETER LIST, THEN
THE LOGICAL FLAG NEWH
WILL BE
SET EQUAL TO .TRUE.,
OTHERWISE IT WILL BE
SET EQUAL TO .FALSE
X AND Y ARE THE
INITIAL VALUES FOR
THE STEP TO BE
COMPUTED.  AFTER
LEAVING THE
SUBROUTINE, THE
ORIGINAL VALUES OF
THE PARAMETERS X
AND Y WILL HAVE BEEN
REPLACED BY X+H' AND
Y(X+H'),
RESPECTIVELY,
```

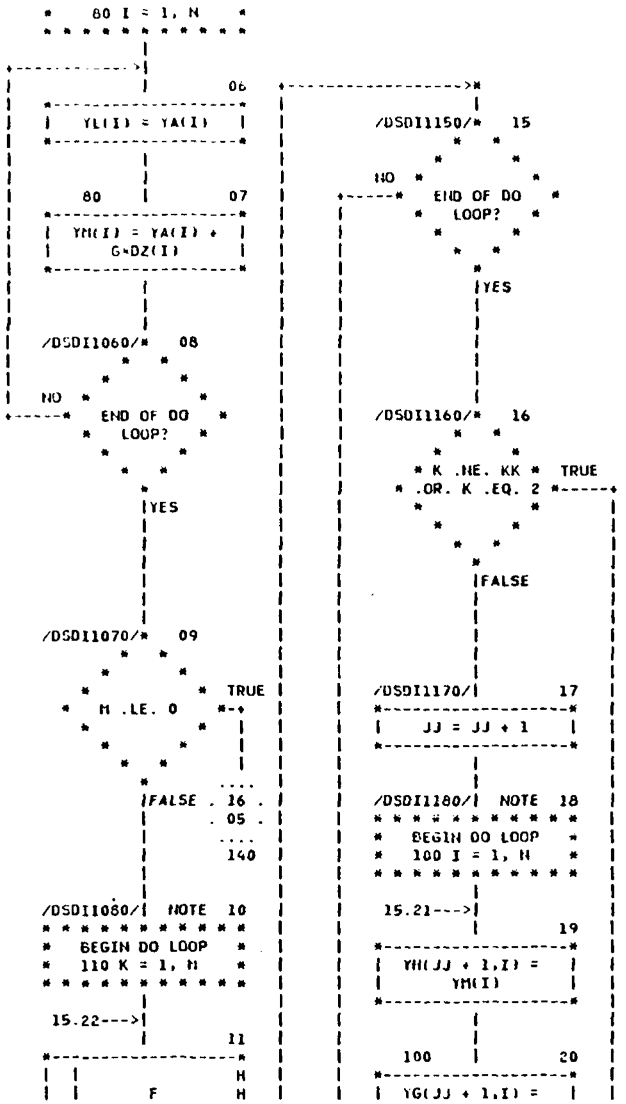


```
WHERE H' IS THE STEP
SIZE ACTUALLY USED.
IN ADDITION THE STEP
SIZE WILL HAVE BEEN
CHANGED AUTOMATICALLY
TO AN ESTIMATED
OPTIMAL
STEP SIZE FOR THE
NEXT INTEGRATION
STEP.  THE ARRAY S
AND THE
CONSTANT EPS ARE USED
TO CONTROL THE
ACCURACY OF THE
COMPUTED
VALUES.  THE
SUBROUTINE IS LEFT,
IF FOR ALL I=1, 2,
...., N TWO
SUCCESSIVE VALUES FOR
Y(I) DIFFER AT MOST
BY AN AMOUNT
EPS*S(I).
EPS SHOULD NOT BE
SMALLER THAN 1.0D-13.
FOR THE FIRST
INTEGRA-
TION STEP IT IS
ADVISABLE TO SET
S(I)=0.0.  BEFORE
RETURN TO THE
```

```
16.24--->|                  |                   |          22
   50    |         16       |        *--------------------*
*--------------------*      |        |  D(4) = 4.0*D(2)   |
|      A = X + H     |      |        |                    |
|                    |      |        |  D(6) = 4.0*D(4)   |
|      FC = 1.5      |      |        |                    |
|                    |      |        |  KONV = J .GT. 2   |
|   6O = .FALSE.     |      |        *--------------------*
|                    |      |                   |
|       M = 1        |      |                   |
|                    |      |                   |
|       R = 2        |      |        /DSDI 910/*    23
|                    |      |                 *   *
|      SR = 3        |      |               *       *
*--------------------*      |             *           *  TRUE
          |                 |           *   J .LE. 6    *--+
          |                 |             *           *    |
/DSDI 830/|        17       |               *       *      |
*--------------------*      |                 *   *        |
|      JJ = - 1      |      |                   *        ....
*--------------------*      |                   |FALSE . 15 .
          |                 |                   |      . 01 .
          |                 |                   |        ....
/DSDI 840/|  NOTE  18       |                   |         60
* * * * * * * * * * *       |                   |
*   BEGIN DO LOOP   *       |                   |
*  190 JI = 1, 10   *       |        /DSDI 920/|          24
* * * * * * * * * * *       |        *--------------------*
          |                 |        |       L = 6        |
          |                 |        |                    |
          +-----------------+        |     D(7) = 64.0    |
                                     |                    |
                                     |   FC = .600*FC     |
                                     *--------------------*
                                                |
                                                |
                                                |
                                               ...
                                             .     .
                                             .15.02.
                                             .     .
                                               ... 70
```

```
CHART TITLE - SUBROUTINE  DSDIFF(F,N,H,X,Y,EPS,S,NEHH)


                  -----------
                 /    60    /
                 -----------
                      |
           14.23--->|
                      |          01
              *-------------------*
              |      L = J        |
              |                   |
              |  D(L + 1) = M*M   |
              *-------------------*
                      |
           14.24--->|
               70     |          02
              *-------------------*
              |     M = 2*M       |
              |                   |
              |       G =         |
              | H/DBLE(FLOAT(M))  |
              |                   |
              |    B = 2.0*G      |
              *-------------------*
                      |
                      |
                      |
             /DSDI1010/*    03
                     *   *
                   *       *
                 *           *   TRUE
               *  BH .AND. J   *-+
                 *   .LT. 8  *   |
                   *       *     |
                     *   *       |
                       *        ....
                       |FALSE . 16 .
                       |      . 01 .
                       |        ....
                       |        120
                       |
                       |
             /DSDI1020/|         04
              *-------------------*
              |  FK = (M - 2)/2   |
              |                   |
              |    M = M - 1      |
              *-------------------*
                       |
                       |
             /DSDI1040/|  NOTE   05
              * * * * * * * * * * *
              *   BEGIN DO LOOP   *
```


80 I = 1, N
06
YL(I) = YA(I)
80
07
YM(I) = YA(I) + G*DZ(I)
/DSDI1060/ 08
NO
END OF DO LOOP?
YES
/DSDI1070/ 09
M .LE. 0
TRUE
FALSE
16
05
140
/DSDI1080/ NOTE 10
BEGIN DO LOOP
110 K = 1, M
15.22--->
11
H
F
H
/DSDI1150/ 15
NO
END OF DO LOOP?
YES
/DSDI1160/ 16
K .NE. KK .OR. K .EQ. 2
TRUE
FALSE
/DSDI1170/ 17
JJ = JJ + 1
/DSDI1180/ NOTE 18
BEGIN DO LOOP
100 I = 1, N
15.21--->
19
YM(JJ + 1,I) = YM(I)
100
20
YG(JJ + 1,I) =

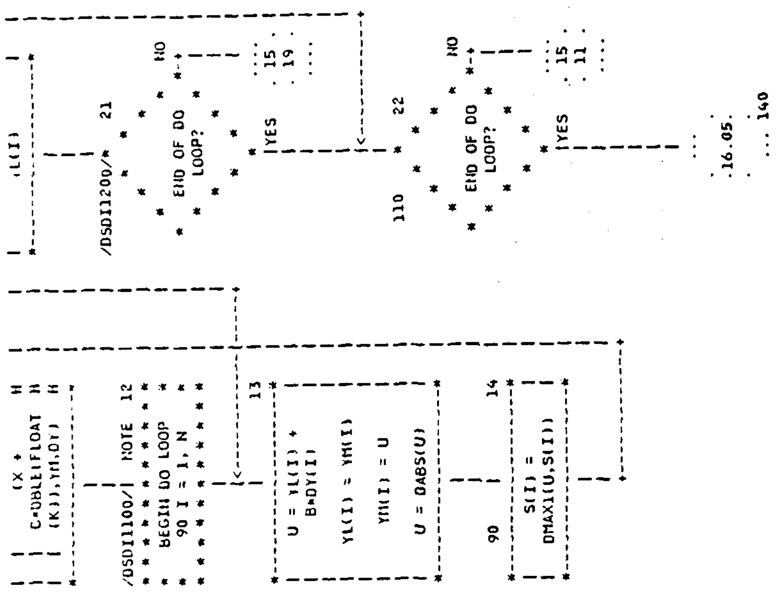
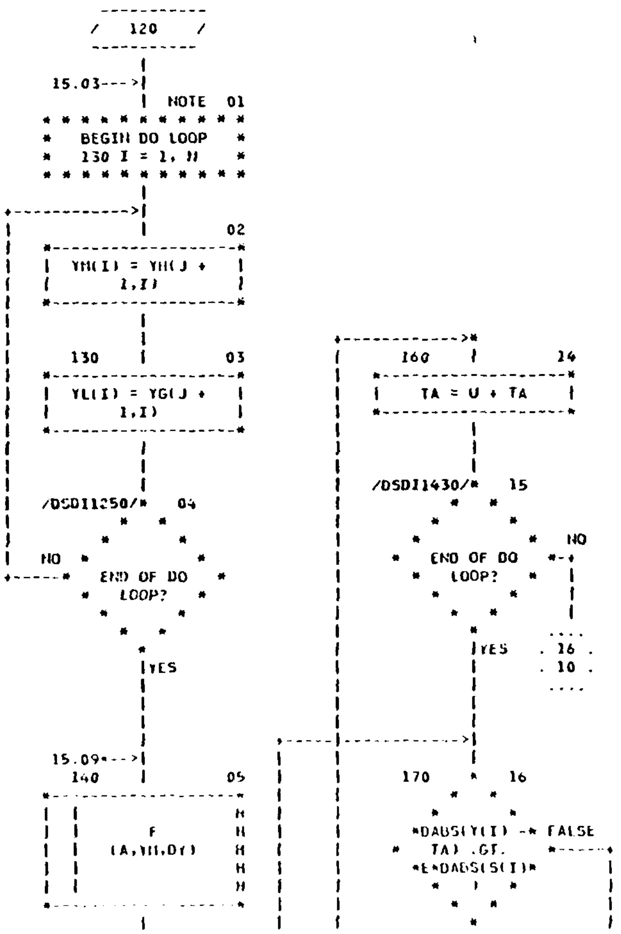
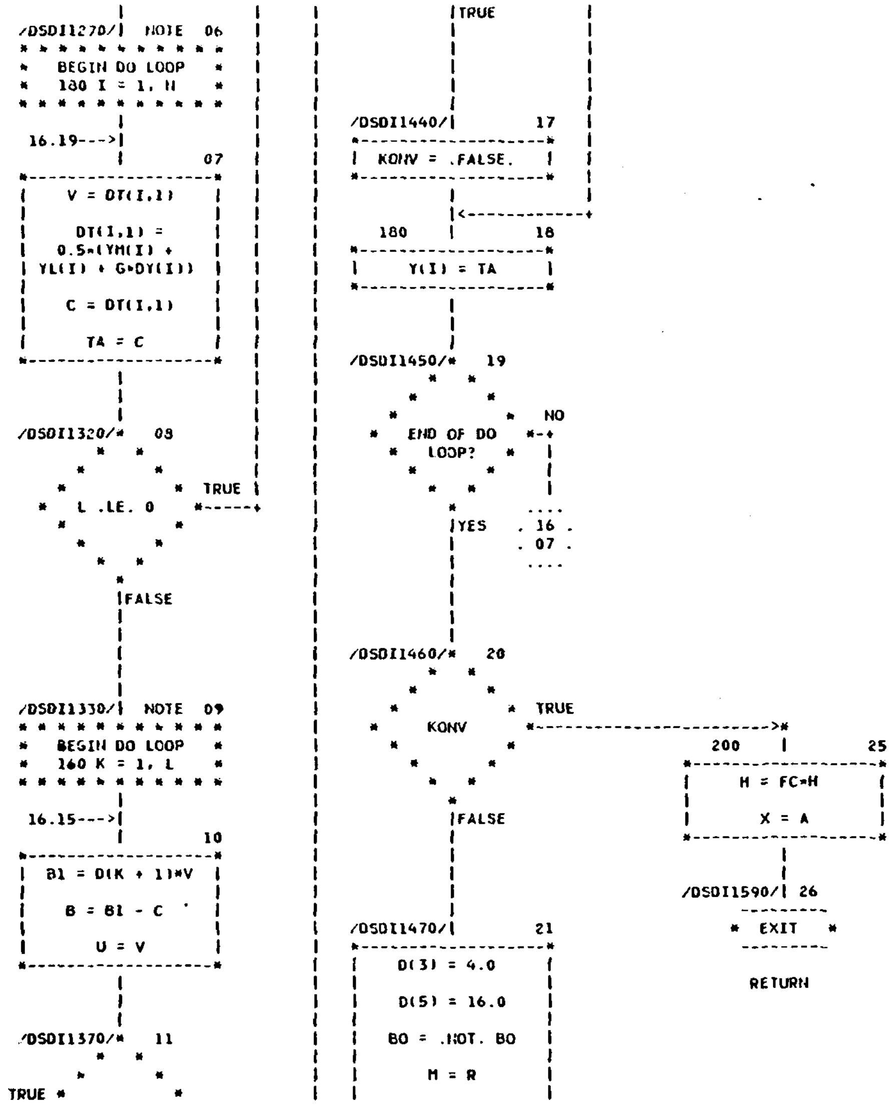
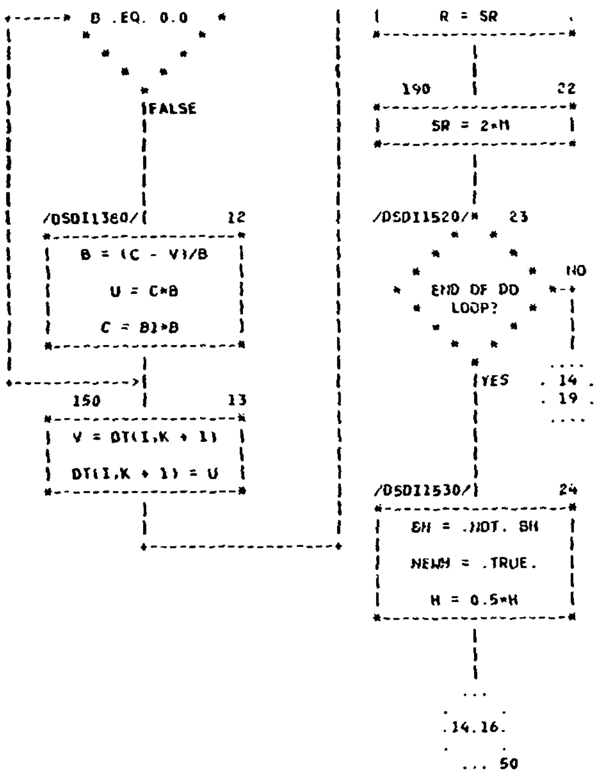

(X +
C*OBLE(FLOAT
(K)),YM,DY)
/DSDI1100/ NOTE 12
BEGIN DO LOOP
90 I = 1, N
13
U = YL(I) +
B*DY(I)
YL(I) = YM(I)
YM(I) = U
U = DABS(U)
90
14
S(I) =
DMAX1(U,S(I))
L(I)
/DSDI1200/ 21
END OF DO
LOOP?
NO
YES
15
19
110
22
END OF DO
LOOP?
NO
YES
15
11
16.05
140

CHART TITLE - SUBROUTINE DSDIFF(F,N,H,X,Y,EPS,S,NEWH)




/DSDI1270/ NOTE 06
BEGIN DO LOOP
180 I = 1, N
16.19--->
07
V = DT(I,1)
DT(I,1) =
0.5*(YM(I) +
YL(I) + G*DY(I))
C = DT(I,1)
TA = C
/DSDI1320/* 08
L .LE. 0
TRUE
FALSE
/DSDI1330/ NOTE 09
BEGIN DO LOOP
160 K = 1, L
16.15--->
10
B1 = D(K + 1)*V
B = B1 - C
U = V
/DSDI1370/* 11
TRUE
TRUE
/DSDI1440/ 17
KONV = .FALSE.
180 18
Y(I) = TA
/DSDI1450/* 19
END OF DO
LOOP?
NO
YES
16
07
/DSDI1460/* 20
KONV
TRUE
FALSE
/DSDI1470/ 21
D(3) = 4.0
D(5) = 16.0
BO = .NOT. BO
M = R
200 25
H = FC*H
X = A
/DSDI1590/ 26
EXIT
RETURN


B .EQ. 0.0
FALSE
/DSDI1360/
12
B = (C - V)/B
U = C*B
C = B1*B
150
13
V = DT(I,K + 1)
DT(I,K + 1) = U
R = SR
190
22
SR = 2*H
/DSDI1520/
23
END OF DO LOOP?
NO
YES
14
19
/DSDI1530/
24
BH = .NOT. BH
NEWH = .TRUE.
H = 0.5*H
14.16.
... 50

```
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 S1(20)/1.D0,19*0.D0/
      REAL*8 COMB(21,21)/21*1.0D0,420*0.0D0/
      REAL*8 PP(20,21,50)
      REAL*8 ARRAYA(20,20)
      REAL*8 X(20),AA(20,21),SS,LAMBDA
      LAMBDA=1.0D0
      P=.95D0
      NS=7
C   NS=THE NUMBER OF SYSTEM STATES
      READ (5,*) NTIME
      N=10
C   N=THE NUMBER OF SOFTWARE BUGS INITIALLY
C    NTIME IS THE ENDING TIME FOR THE CALCULATIONS.
C    THE PROBABILITIES WILL BE PRINTED AT T=0,1,2,...,NTIME.
C   LIMITATIONS ARE: NTIME<=50,NS<=20,N<=20.  TO CHANGE THESE
C   IT IS NECESSARY TO CHANGE SOME DIMENSIONS.
      CALL COMBIN (COMB,N+1)
      C=P*LAMBDA
      T=0.0D0
      IO=0
      NDIM=NS-1
      NN=NS-1
      MM=1
      CALL SOLVE (NS,NTIME,N,PP)
      SUM=0.0D0
      DO 10 I2=1,NS
10    SUM=SUM+S1(I2)
      WRITE (6,160) T,(S1(IW),IW=1,NS),SUM
      DO 150 I=1,NTIME
      T=DFLOAT(I)
      TT=T
      JJ=N-1
      DO 20 I1=1,NS
20    S1(I1)=DEXP(-C*DFLOAT(N)*TT)*PP(I1,N+1,I+1)
      DO 130 J=IO,JJ
      DO 120 L=IO,J
      IU=N-L
      DO 110 M=IO,IU
      DO 30 L1=1,NS
      DO 30 L2=1,NS
30    ARRAYA(L1,L2)=0.0D0
      CALL SYS (ARRAYA,J)
      Z=-C*DFLOAT(J-L-M)
      DO 40 I1=1,NS
      ARRAYA(I1,I1)=ARRAYA(I1,I1)+Z
      AA(I1,NS)=DEXP(Z*TT)*PP(I1,J+1,I+1)
40    CONTINUE
      AA(1,NS)=AA(1,NS)-1.0D0
      CONST=TT
      IF (Z.NE.0.0D0) CONST=(1.D0/Z)*(DEXP(Z*TT)-1.D0)
      NS1=NS-1
      DO 60 I1=1,NS1
      DO 50 I2=1,NS1
 50   ARRAYA(I1,I2)=ARRAYA(I1,I2)-ARRAYA(I1,NS)
 60   CONTINUE
      DO 70 I1=1,NS1
 70   AA(I1,NS)=AA(I1,NS)-CONST*ARRAYA(I1,NS)
      DO 80 I1=1,NS1
      DO 80 I2=1,NS1
 80   AA(I1,I2)=ARRAYA(I1,I2)
      CALL LINEQ (AA,NN,X)
      SS=0.0D0
      DO 90 I1=1,NS1
 90   SS=SS+X(I1)
      X(NS)=CONST-SS
C      WRITE(6,*) (X(LL),LL=1,NS),I,J,L,M
      DO 100 K=1,NS
      S1(K)=S1(K)+COMB(N+1,L+1)*COMB(IU+1,M+1)*((-1)**(M+1))*C*(L+M)*DEX
     1P(-C*(L+M)*T1)*X(K)
100   CONTINUE
110   CONTINUE
120   CONTINUE
130   CONTINUE
      SUM=0.0D0
      DO 140 I2=1,NS
140   SUM=SUM+S1(I2)
150   WRITE (6,160) T,(S1(IW),IW=1,NS),SUM
C     DEBUG INIT(S1,I,J,L,M)
      STOP
C
C
C
160   FORMAT (1X,F3.0,2X,8E14.6)
      END
      SUBROUTINE COMBIN (COMB,ID)
C      THIS ROUTINE COMPUTES  I COMBINATORIAL J FOR
C      I=0,1,...,ID;J=0,1,...,ID.
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 COMB(21,21)
      COMB(2,2)=1.0D0
      DO 20 I=3,ID
      DO 10 J=2,ID
      COMB(I,J)=COMB(I-1,J-1)+COMB(I-1,J)
10    CONTINUE
20    CONTINUE
      RETURN
      END
      SUBROUTINE SYS (A,JJJ)
C      THIS ROUTINE SUPPLIES THE MATRIX AJ
C      (I.E. THE INFINITESIMAL MATRIX. SEE FINAL REPORT) FOR
C      VALUES OF J=0,1,2,...,N.
C      WHEN MODIFYING THIS ROUTINE FOR SPECIFIC CASES THIS
C      ROUTINE USES A IN TRANSPOSED FORM.  SO, FOR EXAMPLE,
C      THE (I,J) ENTRY IN THE INFINITESIMAL MATRIX AS DESCRIBED
C      IN THE FINAL REPORT IS ENTERED IN THIS ROUTINE AS THE
C      (J,I) ELEMENT OF A.
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 A(20,20)
      A(1,1)=-(.05D0*DFLOAT(JJJ)+.02D0)
      A(2,1)=.05D0*DFLOAT(JJJ)
      A(3,1)=.02D0
      A(1,2)=1.D0
      A(2,2)=-1.D0
      A(1,3)=1.D0
      A(3,3)=-(1.018D0+.045*DFLOAT(JJJ))
      A(4,3)=.045D0*DFLOAT(JJJ)
      A(5,3)=.018D0
      A(3,4)=1.D0
      A(4,4)=-1.D0
      A(3,5)=1.D0
      A(5,5)=-(1.016D0+.04D0*DFLOAT(JJJ))
      A(6,5)=.04D0*DFLOAT(JJJ)
      A(7,5)=.016D0
      A(5,6)=1.D0
      A(6,6)=-1.D0
      A(5,7)=1.D0
      A(7,7)=-1.D0
C       A(1,1)=-(.004+DFLOAT(JJJ))
C       A(1,2)=2.D0
C       A(1,3)=2.D0
C       A(2,1)=DFLOAT(JJJ)
C       A(2,2)=-2.D0
C       A(3,1)=.004D0
C       A(3,3)=-2.D0
      RETURN
      END
      SUBROUTINE SOLVE (NS,NTIME,N,PP)
C      THIS ROUTINE SOLVES THE SYSTEM OF DIFFERENTIAL
C      EQUATIONS (KOLMOGOROV EQUATIONS) FOR EACH J AND AT
C      EACH POINT IN TIME.  THE MATRIX PP(I,J,IT) UPON
C      EXIT WILL CONTAIN THE SOLUTION FOR STATE I, WHEN
C      X(T)=J, AT THE POINT T=IT-1), I.E. IN THE NOTATION
C      OF THE REPORT, PJ(IT-1,0,I). IT IS ASSUMED THAT
C      Y(0)=0 WITH PROBABILITY 1.
      IMPLICIT REAL*8(A-H,O-Z)
      EXTERNAL DERIV
      REAL*8 PP(20,21,50),Y(20),DY(20),S(20)
      LOGICAL NEWH
      COMMON /DD/ J,NSS
      H=.1D0
      II=1.D0/H+.1
      EPS=1.D-13
      NSS=NS
      N1=N+1
      N2=NTIME+1
      DO 50 J=1,N1
      PP(1,J,1)=1.0D0
      Y(1)=1.0D0
      S(1)=0.0D0
      DO 10 I1=2,NS
      Y(I1)=0.0D0
      S(I1)=0.0D0
      PP(I1,J,1)=0.0D0
10    CONTINUE
      T1=0.0D0
      DO 40 IT=1,NTIME
      DO 20 I2=1,II
      H=.1D0
20    CALL DSDIFF (DERIV,NSS,H,T1,Y,EPS,S,NEWH)
      DO 30 I1=1,NSS
30    PP(I1,J,IT+1)=Y(I1)
C      WRITE(6,*) (Y(K),K=1,NSS),T1,NEWH,H
40    CONTINUE
50    CONTINUE
C      DEBUG INIT(Y)
      RETURN
      END
      SUBROUTINE DERIV (X,Y,DY)
C      THIS ROUTINE COMPUTES THE DERIVATIVES REQUIRED BY
C      DSDIFF.
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 Y(20),DY(20),A(20,20)/400*0.0D0/
      COMMON /DD/ J,NSS
      JJ=J
      CALL SYS (A,JJ-1)
      DO 10 I1=1,NSS
      DY(I1)=0.0D0
      DO 10 I2=1,NSS
10    DY(I1)=DY(I1)+A(I1,I2)*Y(I2)
      RETURN
      END
      SUBROUTINE LINEQ (AA,NN,X)
C        THIS ROUTINE SOLVES THE SYSTEM OF LINEAR
C       EQUATIONS A*X=B WHERE A IS THE ARRAY CONSISTING OF
C       THE FIRST  NN COLUMNS OF AA AND B IS THE LAST COLUMN
C       OF AA.
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 AA(20,21),X(20)
      M=1
      N=NN
      EPS=1.D-4
      NPLUSM=N+M
      DETER=1.D0
      DO 40 K=1,N
      DETER=DETER*AA(K,K)
      IF (DABS(AA(K,K)).GT.EPS) GO TO 10
      WRITE (6,60)
      STOP
 10   CONTINUE
      KP1=K+1
      DO 20 J=KP1,NPLUSM
 20   AA(K,J)=AA(K,J)/AA(K,K)
      AA(K,K)=1.D0
      DO 40 I=1,N
      IF (I.EQ.K.OR.AA(I,K).EQ.0.D0) GO TO 40
      DO 30 J=KP1,NPLUSM
 30   AA(I,J)=AA(I,J)-AA(I,K)*AA(K,J)
      AA(I,K)=0.D0
 40   CONTINUE
      DO 50 J=1,N
 50   X(J)=AA(J,N+1)
      RETURN
C
C
 60   FORMAT (1X,'ALMOST SINGULAR MATRIX ENCOUNTERED IN LINEQ')
      END
      SUBROUTINE DSDIFF (F,N,H,X,Y,EPS,S,NEWH)
C
C     F IS THE NAME OF A SUBROUTINE CALLED BY 'CALL F(X,Z,DZ)' WHICH
C          STORES IN THE VECTOR DZ THE N COMPONENTS OF THE DERIVATIVE
C          DZ/DX ACCORDING TO THE DIFFERENTIAL EQUATION WHICH IS BEING
C          SOLVED, DZ/DX=F(X,Z).
C     X, Z, AND DZ MUST BE DOUBLE PRECISION.
C
C     N IS THE ORDER OF THE SYSTEM OF DIFFERENTIAL EQUATIONS.
C     N MUST BE NO GREATER THAN MAXORD WHICH IS SET BELOW.
C
C     H IS THE BASIC STEP SIZE.
C
C     X AND Y(VECTOR) ARE THE INITIAL VALUES.
C
C     EPS AND S(VECTOR) ARE THE ERROR BOUNDS.
C     DABS(EPS) SHOULD BE NO SMALLER THAN 1.0D-13.
C
C     NEWH IS A FLAG WHICH IS SET EQUAL TO .TRUE. IF THE STEP SIZE USED
C          USED BY DSDIFF IS DIFFERENT FROM THE STEP SIZE H GIVEN IN THE
C          PARAMETER LIST.  NEWH IS SET EQUAL TO .FALSE. OTHERWISE.
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 Y(N),S(N)
      REAL*8 YL(25)
      REAL*8 DZ(25)
      REAL*8 YA(25),YM(25),DY(25),DT(25,7),YG(8,25),YH(8,25),D(7)
      INTEGER R,SR
      LOGICAL NEWH
      LOGICAL KONV,BO,BH
      LOGICAL EPSERR
      DATA MAXORD/25/
      DATA EPSERR/.FALSE./
C*********************************************************************
C
C          EACH CALL OF DSDIFF PERFORMS ONE INTEGRATION STEP OF THE
C     EQUATION DY/DX=F(X,Y) ACCORDING TO THE METHOD OF R. BULIRSCH AND
C     J. STOER (NUMERISCHE MATHEMATIK, IN PRESS).  THE STEP SIZE WILL
C     BE LESS THAN OR EQUAL TO H.  THE PROGRAM TAKES THE FIRST OF THE
C     NUMBERS H, H/2, H/4, ..., AS STEP SIZE FOR WHICH NO MORE THAN 9
C     EXTRAPOLATION STEPS ARE NEEDED TO OBTAIN A SUFFICIENTLY ACCURATE
C     RESULT.  IF THE STEP SIZE USED IS DIFFERENT THAN THE STEP SIZE
C     GIVEN IN THE PARAMETER LIST, THEN THE LOGICAL FLAG NEWH WILL BE
C     SET EQUAL TO .TRUE., OTHERWISE IT WILL BE SET EQUAL TO .FALSE..
C     X AND Y ARE THE INITIAL VALUES FOR THE STEP TO BE COMPUTED.  AFTER
C     LEAVING THE SUBROUTINE, THE ORIGINAL VALUES OF THE PARAMETERS X
C     AND Y WILL HAVE BEEN REPLACED BY X+H' AND Y(X+H'), RESPECTIVELY,
C     WHERE H' IS THE STEP SIZE ACTUALLY USED.  IN ADDITION THE STEP
C     SIZE WILL HAVE BEEN CHANGED AUTOMATICALLY TO AN ESTIMATED OPTIMAL
C     STEP SIZE FOR THE NEXT INTEGRATION STEP.  THE ARRAY S AND THE
```

```
C     CONSTANT EPS ARE USED TO CONTROL THE ACCURACY OF THE COMPUTED
C     VALUES.  THE SUBROUTINE IS LEFT, IF FOR ALL I=1, 2, ..., N TWO
C     SUCCESSIVE VALUES FOR Y(I) DIFFER AT MOST BY AN AMOUNT EPS*S(I).
C     EPS SHOULD NOT BE SMALLER THAN 1.0D-13.  FOR THE FIRST INTEGRA-
C     TION STEP IT IS ADVISABLE TO SET S(I)=0.0.  BEFORE RETURN TO THE
C     CALLING PROGRAM, THE ARRAY S WILL HAVE HAD ITS CONTENTS MODIFIED
C     SO THAT S(I)=MAX(S(I),ABS(Y(I,X)), WHERE THE MAXIMUM IS TAKEN OVER
C     THE INTEGRATION INTERVAL (X,X+H').
C
C*****************************************************************
C
      IF (N.GT.0.AND.N.LE.MAXORD) GO TO 10
      WRITE (6,210)
      STOP
10    E=DABS(EPS)
      IF (E.GE.1.0D-13) GO TO 30
      IF (EPSERR) GO TO 20
      EPSERR=.TRUE.
      WRITE (6,220)
20    E=1.0D-13
C
30    CALL F (X,Y,DZ)
      BH=.FALSE.
      NEWH=.FALSE.
      DO 40 I=1,N
40    YA(I)=Y(I)
50    A=X+H
      FC=1.5
      BO=.FALSE.
      M=1
      R=2
      SR=3
      J=-1
      DO 190 J1=1,10
      J=J1-1
      D(2)=2.25
      IF (BO) D(2)=4.0/D(2)
      D(4)=4.0*D(2)
      D(6)=4.0*D(4)
      KONV=J.GT.2
      IF (J.LE.6) GO TO 60
      L=6
      D(7)=64.0
      FC=.6D0*FC
      GO TO 70
60    L=J
      D(L+1)=M*M
70    M=2*M
      G=H/DBLE(FLOAT(M))
      B=2.0*G
      IF (BH.AND.J.LT.8) GO TO 120                                      DSDI1010
      KK=(M-2)/2                                                        DSDI1020
      M=M-1                                                             DSDI1030
      DO 80 I=1,N                                                       DSDI1040
      YL(I)=YA(I)                                                       DSDI1050
80    YM(I)=YA(I)+G*DZ(I)                                               DSDI1060
      IF (M.LE.0) GO TO 140                                             DSDI1070
      DO 110 K=1,M                                                      DSDI1080
      CALL F (X+G*DBLE(FLOAT(K)),YM,DY)                                 DSDI1090
      DO 90 I=1,N                                                       DSDI1100
      U=YL(I)+B*DY(I)                                                   DSDI1110
      YL(I)=YM(I)                                                       DSDI1120
      YM(I)=U                                                           DSDI1130
      U=DABS(U)                                                         DSDI1140
90    S(I)=DMAX1(U,S(I))                                                DSDI1150
      IF (K.NE.KK.OR.K.EQ.2) GO TO 110                                  DSDI1160
      JJ=JJ+1                                                           DSDI1170
      DO 100 I=1,N                                                      DSDI1180
      YH(JJ+1,I)=YM(I)                                                  DSDI1190
100   YG(JJ+1,I)=YL(I)                                                  DSDI1200
110   CONTINUE                                                          DSDI1210
      GO TO 140                                                         DSDI1220
120   DO 130 I=1,N                                                      DSDI1230
      YM(I)=YH(J+1,I)                                                   DSDI1240
130   YL(I)=YG(J+1,I)                                                   DSDI1250
140   CALL F (A,YM,DY)                                                  DSDI1260
      DO 180 I=1,N                                                      DSDI1270
      V=DT(I,1)                                                         DSDI1280
      DT(I,1)=0.5*(YM(I)+YL(I)+G*DY(I))                                 DSDI1290
      C=DT(I,1)                                                         DSDI1300
      TA=C                                                              DSDI1310
      IF (L.LE.0) GO TO 170                                             DSDI1320
      DO 160 K=1,L                                                      DSDI1330
      B1=D(K+1)*V                                                       DSDI1340
      B=B1-C                                                            DSDI1350
      U=V                                                               DSDI1360
      IF (B.EQ.0.0) GO TO 150                                           DSDI1370
      B=(C-V)/B                                                         DSDI1380
      U=C*B                                                             DSDI1390
      C=B1*B                                                            DSDI1400
150   V=DT(I,K+1)                                                       DSDI1410
      DT(I,K+1)=U                                                       DSDI1420
160   TA=U+TA                                                           DSDI1430
170   IF (DABS(Y(I)-TA).GT.E*DABS(S(I))) KONV=.FALSE.                   DSDI1440
180   Y(I)=TA                                                           DSDI1450
      IF (KONV) GO TO 200                                               DSDI1460
      D(3)=4.0                                                          DSDI1470
      D(5)=16.0                                                         DSDI1480
      BO=.NOT.BO                                                        DSDI1490
      M=R                                                               DSDI1500
      R=SR                                                              DSDI1510
 190  SR=2*M                                                            DSDI1520
      BH=.NOT.BH                                                        DSDI1530
      NEWH=.TRUE.                                                       DSDI1540
      H=0.5*H                                                           DSDI1550
      GO TO 50                                                          DSDI1560
 200  H=FC*H                                                            DSDI1570
      X=A                                                               DSDI1580
      RETURN                                                            DSDI1590
C                                                                       DSDI1600
C                                                                       DSDI1610
C                                                                       DSDI1620
 210  FORMAT (1X,'ERROR IN DSDIFF; ORDER LESS THAN 1 OR GREATER THAN 25'DSDI1630
     1)                                                                 DSDI1640
 220  FORMAT (51H0ERROR LIMIT TOO SMALL FOR DSDIFF.  WE USE 1.0D-13.)   DSDI1650
      END                                                               DSDI1660
```

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.