NUREG/CR-6942

Dynamic Reliability Modeling of Digital Instrumentation and Control Systems for Nuclear Reactor Probabilistic Risk Assessments

The Ohio State University

U.S. Nuclear Regulatory Commission
Office of Nuclear Regulatory Research
Washington, DC 20555-0001

NUREG/CR-6942

Dynamic Reliability Modeling of Digital Instrumentation and Control Systems for Nuclear Reactor Probabilistic Risk Assessments

Manuscript Completed: May 2006
Date Published: October 2007

Prepared by
T. Aldemir[1], M.P. Stovsky[1], J. Kirschenbaum[2], D. Mandelli[1], P. Bucci[2], L.A. Mangan[1], D.W. Miller[1], X. Sun[1], E. Ekici[3], S. Guarro[4], M. Yau[4], B. Johnson[5], C. Elks[5], and S.A. Arndt[6]

[1]The Ohio State University
Department of Mechanical Engineering
Nuclear Engineering Program
Columbus, OH 43210

[2]The Ohio State University
Department of Computer Science and Engineering
Columbus, OH 43210

[3]The Ohio State University
Department of Electrical and Computer Engineering
Columbus, OH 43210

[4]ASCA, Inc.
1720 S. Catalina Avenue, Suite 220
Redondo Beach, CA 90277-5501

[5]University of Virginia
Department of Electrical and Computer Engineering
Charlottesville, VA 22904

[6]U. S. Nuclear Regulatory Commission
Washington, DC 20555-0001

S.A. Arndt, NRC Project Manager

Prepared for
Division of Fuel, Engineering and Radiological Research
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555-0001
NRC Job Code K6472

AVAILABILITY OF REFERENCE MATERIALS IN NRC PUBLICATIONS

NRC Reference Material

As of November 1999, you may electronically access NUREG-series publications and other NRC records at NRC's Public Electronic Reading Room at http://www.nrc.gov/reading-rm.html. Publicly released records include, to name a few, NUREG-series publications; *Federal Register* notices; applicant, licensee, and vendor documents and correspondence; NRC correspondence and internal memoranda; bulletins and information notices; inspection and investigative reports; licensee event reports; and Commission papers and their attachments.

NRC publications in the NUREG series, NRC regulations, and *Title 10, Energy*, in the Code of *Federal Regulations* may also be purchased from one of these two sources.

1. The Superintendent of Documents
   U.S. Government Printing Office
   Mail Stop SSOP
   Washington, DC 20402–0001
   Internet: bookstore.gpo.gov
   Telephone: 202-512-1800
   Fax: 202-512-2250
2. The National Technical Information Service
   Springfield, VA 22161–0002
   www.ntis.gov
   1–800–553–6847 or, locally, 703–605–6000

A single copy of each NRC draft report for comment is available free, to the extent of supply, upon written request as follows:

Address: Office of the Chief Information Officer,
Reproduction and Distribution
Services Section
U.S. Nuclear Regulatory Commission
Washington, DC 20555-0001
E-mail: DISTRIBUTION@nrc.gov
Facsimile: 301–415–2289

Some publications in the NUREG series that are posted at NRC's Web site address http://www.nrc.gov/reading-rm/doc-collections/nuregs are updated periodically and may differ from the last printed version. Although references to material found on a Web site bear the date the material was accessed, the material available on the date cited may subsequently be removed from the site.

Non-NRC Reference Material

Documents available from public and special technical libraries include all open literature items, such as books, journal articles, and transactions, *Federal Register* notices, Federal and State legislation, and congressional reports. Such documents as theses, dissertations, foreign reports and translations, and non-NRC conference proceedings may be purchased from their sponsoring organization.

Copies of industry codes and standards used in a substantive manner in the NRC regulatory process are maintained at—

The NRC Technical Library
Two White Flint North
11545 Rockville Pike
Rockville, MD 20852–2738

These standards are available in the library for reference use by the public. Codes and standards are usually copyrighted and may be purchased from the originating organization or, if they are American National Standards, from—

American National Standards Institute
11 West 42nd Street
New York, NY 10036–8002
www.ansi.org
212–642–4900

Legally binding regulatory requirements are stated only in laws; NRC regulations; licenses, including technical specifications; or orders, not in NUREG-series publications. The views expressed in contractor-prepared publications in this series are not necessarily those of the NRC.

The NUREG series comprises (1) technical and administrative reports and books prepared by the staff (NUREG–XXXX) or agency contractors (NUREG/CR–XXXX), (2) proceedings of conferences (NUREG/CP–XXXX), (3) reports resulting from international agreements (NUREG/IA–XXXX), (4) brochures (NUREG/BR–XXXX), and (5) compilations of legal decisions and orders of the Commission and Atomic and Safety Licensing Boards and of Directors' decisions under Section 2.206 of NRC's regulations (NUREG–0750).

ABSTRACT

As part of the U.S. Nuclear Regulatory Commission's (NRC's) effort to advance the state-of-the-art in digital system risk and reliability analysis the NRC Office of Nuclear Regulatory Research is sponsoring research into both traditional and dynamic methods for modeling. The results of a recent study reported in NUREG/CR-6901 indicate that the conventional event-tree (ET)/fault-tree (FT) methodology may not yield satisfactory results in the reliability modeling of digital I&C systems. Using subjective criteria based on reported experience, NUREG/CR-6901 has identified the dynamic flowgraph methodology (DFM) and the Markov methodology as the methodologies that rank as the top two with most positive features and least negative or uncertain features when evaluated against the requirements for the reliability modeling of digital I&C systems. The NUREG/CR-6901 has also concluded that benchmark systems should be defined to allow assessment of the dynamic methodologies proposed for the reliability modeling of digital I&C systems using a common set of hardware/ software/ firmware states and state transition data. This report: a) defines such a benchmark system based on the steam generator feedwater control system of an operating pressurized water reactor (PWR), b) provides procedures to illustrate how dynamic reliability models for the benchmark system can be constructed using DFM and Markov methodologies, and, c) illustrates how the resulting dynamic reliability models can be integrated into the probabilistic risk assessment (PRA) model of an existing PWR using SAPHIRE as an example ET/FT PRA tool. The report also discusses to what extent the DFM and the Markov methodology meet the requirements given in NUREG/CR-6901 for the reliability modeling of digital I&C systems. Some challenges are identified. It is concluded that it may be possible to meet most of these challenges by linking the existing ET/FT based plant PRA tools to dynamic methodologies through user friendly interfaces and using distributed computing. The challenge that is the most difficult to address is the acceptability of the failure data used. While it is also concluded that the proposed methods can be used to obtain qualitative information on the failure characteristics of digital I&C systems as well as quantitative, and, in that respect, can be helpful in the identification of risk important event sequences even if the data issue is not resolved, the report presents only a proof-of-concept study. Additional work is needed to validate the practicality of the proposed methods for other digital systems and resolve the challenges identified.

FOREWORD

In 1995, the U.S. Nuclear Regulatory Commission (NRC) issued its Probabilistic Risk Assessment (PRA) Policy Statement, which encourages increased use of PRA and associated analyses in all regulatory matters, to the extent supported by the state-of-the-art in PRA and the data. Toward that end, the NRC's Office of Nuclear Regulatory Research (RES) is sponsoring research to evaluate and develop traditional PRA modeling methods [e.g., event tree/fault tree (ET/FT) approaches] and dynamic PRA methods for use in modeling digital instrumentation and control (I&C) systems. The research presented in this report is one part of the NRC's overall effort to advance the state-of-the-art in digital system risk and reliability modeling to provide a means to risk-inform the agency's licensing reviews of digital I&C systems.

The NRC has not yet implemented risk-informed decision-making in the review of digital I&C systems because the agency and the nuclear industry do not presently have universally accepted methods for modeling digital system reliability. Additionally, the PRA technical community has not yet agreed on the level of detail that digital I&C systems require in reliability modeling. Nonetheless, it is clear that PRA models must adequately represent the complex system interactions that can contribute to digital system failure modes.

While the traditional ET/FT approach has been used in modeling the reliability of digital I&C systems in nuclear power plants, the technical literature has raised numerous concerns regarding the capability of that approach to properly account for dynamic interactions that can occur in digital systems. Studies indicate that such interactions may lead to coupling between the triggered or stochastic logic events (e.g., valve opening, pump startup, etc.) during an accident, with significant impacts on predicated systems failure probabilities. Dynamic methods, such as dynamic fault trees, Markov models, and the dynamic flowgraph methodology (DFM), can account for the coupling between systems through explicit consideration of time in system evolution and interaction. NUREG/CR-6901, "Current State of Reliability Modeling Methods for Digital Systems and Their Acceptance Criteria for Nuclear Power Plant Assessments," identified Markov models and DFM as the most promising dynamic methods for modeling digital systems.

This report provides a proof-of-concept for the use of Markov models and DFM to model digital I&C systems. It illustrates how these dynamic models can be developed and integrated into PRAs, using a representative benchmark system and an existing PRA model of a pressurized-water reactor. As part of this ongoing research, the NRC will complete and publish a separate study to quantify the reliability and risk of the digital system modeled in this report.

Brian W. Sheron, Director
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission

CONTENTS

Figures

Tables

EXECUTIVE SUMMARY

The results of a recent study published as NUREG/CR-6901(Current State of Reliability Modeling Methodologies for Digital Systems and Their Acceptance Criteria for Nuclear Power Plant Assessments) indicate that the conventional event-tree (ET)/fault-tree (FT) methodology may not yield satisfactory results when a digital I&C system:

- interacts with a process that has multiple Top Events, logic loops and/or substantial time delay (with respect to system time constants) between the initiation of the fault and Top Event occurrence,
- relies on sequential circuits that have memory,
- has tasks that compete for the I&C system resources, and,
- anticipates the future states of controlled/monitored process.

Using subjective criteria based on reported experience, the study NUREG/CR-6901 has identified the dynamic flowgraph methodology (DFM) and the Markov methodology coupled with the cell-to-cell-mapping technique(CCMT) as the methodologies that rank as the top two with the most positive features and least negative or uncertain features when evaluated against the requirements for the reliability modeling of digital I&C systems. The NUREG/CR-6901 also concluded that benchmark systems should be defined to allow assessment of the methodologies proposed for the reliability modeling of digital I&C systems using a common set of hardware/software/firmware states and state transition data.

This report presents a benchmark system that can be used for such a comparison and then illustrates how the DFM and Markov/CCMT methodologies can be implemented for the reliability modeling of the benchmark system. The report also describes how the outputs of these methodologies can be incorporated into an existing probabilistic risk assessment (PRA) for a nuclear power plant, using the SAPHIRE code as an example ET/FT PRA tool. The compliance of the procedures proposed with the requirements for the reliability modeling of digital I&C systems given in NUREG/CR-6901 is discussed.

The benchmark system is based on the steam generator (SG) digital feedwater control system (DFWCS) of an operating 2-loop pressurized water reactor (PWR). Each DFWCS controls a feedwater pump (FP), a main feedwater regulating valve (MFV), and a bypass feedwater regulating valve (BFV). The feedwater control system operates in four different modes, depending on the power generated in the primary system.

Each digital feedwater controller is comprised of a main computer (MC) and backup computer (BC), MFV, BFV, and FP controllers which provide both control and fault tolerant capabilities. A pressure drop indicator (PDI) serves as a backup for the MFV controller by sampling the output of the MFV controller. If the MFV controller fails, the PDI controller serves as a manual controller for the MFV. The signals from all the controllers are cross connected to provide redundancy as well as fault tolerance. The coupling between control hardware, software/firmware and process variables are described through differential/algebraic equations for level, compensated level, level error, flow demand, compensated flow error, compensated power, FP, MFV and BFV demands, FP speed, MFV position and BFV position. The coupling between the feedwater flow into the SG and steam flow out of the SG is represented through another set of differential/algebraic equations and steam tables which comprise the 2-region

SG model. Some challenging properties of the benchmark from a reliability modeling viewpoint include:

- possible dependence of the control action on system history,
- possible dependence of system failure modes (low SG level, high SG level) on exact timing of the failures,
- possible introduction of artifacts during power changes,
- functional as well as intermittent failure possibility,
- error detection capability, and
- possible system recovery from failure modes.

A failure data generation procedure using a 3-state Markov model and field data appended with fault injection is also described.

The implementation of the DFM and Markov/CCMT methodologies is illustrated using the benchmark system. The DFM combines multi-valued logic modeling and analysis capabilities to handle systems consisting of components that have multiple degraded states and exhibit dynamic behavior. In applying DFM, the system of interest is first represented in a digraph (directed graph) model. Once such a model has been produced, automated deductive/ inductive algorithms that are built into the methodology can be applied to: a) identify how system level states (which may represent specific conditions of interest, be they success, anomaly or failure states) can be produced by any combinations and sequences of basic component states (deductive analysis), and/or, b) determine how a particular basic component state can produce various possible sequences and system-level states (inductive analysis). The DFM can provide the multi-state and time-dependent equivalent of both FT analysis and failure mode and effect analysis (FMEA).

In the reliability model construction using the Markov/CCMT methodology, the system evolution is represented through a series of discrete transitions within the system state/controlled variable space. These transitions take into account the natural dynamic behavior of the controlled/monitored process variables, the control laws, and hardware/firmware/software states. The discrete transitions for the process variables are modeled using the cell-to-cell mapping technique which represents the system dynamics in terms of transition probabilities between computational cells that partition the system state/controlled variable space. System state for the Markov model is defined in terms of both the system location in the discretized state/controlled variable space and hardware/software/firmware configurations. Once the model is constructed, dynamic event trees can be obtained for any initiating event which would yield the possible event sequences following the initiating event.

A turbine trip initiating event is used to illustrate how the reliability model for the benchmark system can be incorporated into an existing PRA. For the PRA model, the SAPHIRE model of a NUREG-1150 (Severe Accident Risks: An Assessment for Five U.S. Nuclear Power Plants) plant is used.

The results of the study shows that: a) both the DFM and the Markov/CCMT methodology can account for all the features of the benchmark system with good agreement in their results, and b) their results can be integrated into an existing PRA. Possible challenges with the methodologies include:

1. analyst skill levels needed for the implementation of the methodologies,
2. computational demand for the correct description of the coupling between failure events,
3. acceptability of the data used for quantification by a significant portion of the technical community, and
4. the limitation in the capabilities of the existing ET/FT based plant PRA tools to represent the timing of failure events.

Challenges 1 and 2 originate from the complexity and diverse nature of the phenomena to be accounted for and are not specific to DFM or the Markov/CCMT methodology. It may be possible to address the limitations posed by Challenges 1, 2 and 4 by linking the existing ET/FT based plant PRA tools to dynamic methodologies through user friendly interfaces and using distributed computing. Challenge 4 may also require post-processing of the results obtained from the plant PRAs after the integration of the digital I&C system reliability models to remove timing inconsistencies between minimal cut set events.

Challenge 3 is perhaps the most difficult to address. There is no consensus in the technical community on how software reliability should be quantified and, in fact, whether such a concept is appropriate at all. However, the proposed methodologies can be used to obtain qualitative information on the failure characteristics of digital I&C systems (i.e. prime implicants) as well as quantitative, and, in that respect, can be helpful in the identification of risk important event sequences even if the data issue is not resolved.

Finally, the properties of the benchmark system considered in this study may not apply to all the reactor protection and control systems in nuclear power plants . For digital I&C systems which may have less complex interaction between the failure events, the conventional ET/FT approach may be adequate for the reliability modeling of the system. It is also important to note that the report presents only a proof-of-concept study. Additional work is needed to validate the practicality of the proposed methods for other digital systems and resolve the challenges identified.

ABBREVIATIONS

ACRS	Advisory Committee on Reactor Safeguards
AFW, AFS	Auxilliary feedwater system
AOV	Air operated valve
Arb	DFM state for the BFV controller failed in the arbitrary state
ATVG	Automatic test vector generation
ATWS	Anticipated transient without scram
BC	Backup computer
BFV	Bypass flow valve
BFVC	DFM node for BFV Controller
Cdf	Cumulative distribution function
CL	DFM node for compensated level
CMF	Common mode failure
Comp	DFM node for backup computer
CP	DFM node for compensated power
CST	Cell-to-cell mapping technique
CCMT	Condensate storage tank
COTS	Commercial off the shelf
CPU	Central processing unit
CVSS	Controlled variable state space
DET	Dynamic event tree
DFM	Dynamic flowgraph methodology
DFWCS	Digital feedwater control system
ECR	Error containment regions
EL	DFM node for level error
ELP	DFM node for previous level error
ET	Event tree
ET/FT	Event tree/fault tree methodology
FMEA	Failure modes and effects analysis
FP	Feed pump
Frz	DFM state of the BFV controller failed in the frozen state
F-S	DFM state for the BFV failed stuck
fSN	DFM Node for steam flow
FT	Fault tree
FTA	Fault tree analysis
FWCS	Feedwater control system
HP	High pressure
HPI	High pressure injection
HW	Hardware
ICE	In-circuit emulator
L	DFM node for steam generator level
LOCA	Loss of coolant accident
LOSP	Loss of offsite power
LP	DFM node for previous steam generator level
I&C	Instrumentation and control
MC	Main computer
MDP	Motor driven pump

MEI	Mutually exclusive implicant
MFV	Main flow valve
MFVC	Main flow valve controller
MFW	Main feedwater system
MOV	Motor operated valve
MS	Macro-state (see Fig.2.3.2)
MSIV	Main Steam Isolation Valve
MTBF	Mean time between failures
NAS	National Academy of Sciences
NASA	National Aeronautics and Space Administration
NATO	North Atlantic Treaty Organization
NRC	Nuclear Regulatory Commission
pdf	Probability distribution function
PDI	PDI controller
PI	Prime Implicant (from deductive DFM Analysis), Proportional Integral (controller)
PID	Proportional Integral Derivative (controller)
PWR	Pressurized water reactor
RPS	Reactor protection system
Sbn	DFM node for the BFV position
SbnP	DFM node for the previous BFV position
SE	Sensor
SLD	Source level debugger
SRV	Safety relief valve
SG	Steam generator
SW	Software
TDP	Turbine driven pump
Tfk	Label for transfer box k in the DFM Model
Ttk	Label for transition box k in the DFM Model

1. INTRODUCTION

1.1 Purpose of the Report

Nuclear power plants are in the process of replacing and upgrading aging and obsolete instrumentation and control (I&C) systems. Most of these replacements involve transitions from analog to digital technology. The current path for licensing of digital I&C upgrades and new designs relies on the NRC's deterministic regulations. In 1995, the U.S. Nuclear Regulatory Commission (NRC) issued the Probabilistic Risk Assessment (PRA) Policy Statement, which encourages the increased use of PRA and associated analyses in all regulatory matters to the extent supported by the state-of-the-art in PRA and the data. This policy applies, in part, to the review of digital systems, which offer the potential to improve plant safety and reliability through such features as increased hardware reliability and stability and improved failure detection capability[1]. However, there are presently no universally accepted methods for modeling digital systems in current-generation PRAs. Further, there are ongoing debates among the PRA technical community regarding the level of detail that any digital system reliability model must have to adequately model the complex system interactions that can contribute to digital systems failure modes.

The purpose of this report is to explain the next step in advancing the state-of-the-art in digital system risk and reliability modeling. The overall objective of this research program is to advanced the state-of-the-art to the point where the NRC can develop risk-informed decision making guidance for digital systems. The goal of this research is to develop an approach that is acceptable for modeling digital systems, using both traditional and dynamic modeling methods. This report provides a proof-of concept for the use of dynamic methods (Markov models and Dynamic Flowgraph Methodology (DFM)) for modeling of digital systems. The report illustrates how these dynamic models can be developed and integrated into PRAs, using an example benchmark system and an existing PRA model of a pressurized water reactor (PWR).

An important feature of digital I&C systems that distinguishes them from analog systems is the presence of software/firmware, however, even though many activities such as configuration management, testing, and verification and validation are carried out for digital I&C systems to ensure a high quality product, processes for quantitatively assessing the risk implications of digital upgrades have not yet been widely accepted. The results of a recent study whose findings have been published in NUREG/CR-6901 [2] indicate that the conventional event-tree (ET)/fault-tree (FT) methodology may not yield satisfactory results when a digital I&C system:

- interacts with a process that has multiple Top Events, logic loops and/or substantial time delay between the initiation of the fault and Top Event occurrence,
- relies on sequential circuits which have memory,
- has tasks which compete for the I&C system resources, and
- anticipates the future states of controlled/monitored process.

While NUREG/CR-6901 [2] provides a survey of dynamic methodologies that have potential applicability to the reliability modeling of digital I&C systems relevant to reactor protection and control systems, it also indicates that a lack of benchmark against which the dynamic

methodologies (as well as the ET/FT approach) can be evaluated makes comparison difficult. Using subjective criteria based on reported experience, NUREG/CR-6901[2] has identified the DFM [3, 4] and the Markov methodology [5, 6] as the methodologies that rank as the top two with most positive features and least negative or uncertain features when evaluated against the requirements for the reliability modeling of digital I&C systems (Section 1.2). The NUREG/CR-6901[2] has also concluded that benchmark systems should be defined to allow assessment of the methodologies proposed for the reliability modeling of digital I&C systems using a common set of hardware/software/firmware states and state transition data.

In response to this conclusion of NUREG/CR-6901[2], this report presents a first benchmark digital feedwater control system (DFWCS) that can be used for such a comparison (Chapter 2) and then illustrates how the DFM (Chapter 3) and a combination of the Markov methodology with the cell-to-cell-mapping technique (CCMT) (Chapter 4) can be implemented for the reliability modeling of the benchmark system. A possible characterization of digital systems is also given in Section 1.3. Chapters 5 and 6 describe how the outputs of these methodologies can be incorporated into an existing probabilistic risk assessment (PRA) for a nuclear power plant, using the SAPHIRE code [7] as an example ET/FT PRA tool. Chapter 7 discusses the possible sources of uncertainty in the DFM and Markov/CCMT results and some issues in their quantification. Chapter 8 discusses the compliance of the procedures described in Chapters 3, 4 and 5 with the requirements for the reliability modeling of digital instrumentation and control systems (Section 1.2). Chapter 9 gives the conclusions of the study. While specification, design or operation/maintenance errors can be contributors to the risk importance of digital I&C systems, these types of errors are not within the scope of the study.

At this point it should be reiterated that the main objective of this report is to illustrate the implementation of the DFM and Markov/CCMT methodology on a system representative of the digital I&C systems used in nuclear power plants. It does not aim to demonstrate that dynamic methodologies will identify new failure modes. The fact that they may, has already been shown in the literature as described in detail in NUREG/CR-6901. The report does show, however, that the significance of the timing of faults to the system failure mode (Section 2.5) which would be very difficult to identify through a conventional failure modes and effects analysis. It would basically involve going though the steps to generate the cell-to-cell-transition probabilities of Markov/CCMT methodology (Section 4.2.4) or the decision tables of DFM (Section 3.1.1), except without the systematic organization of the computations and procedures to assess the consequence of the results on the system failure modes and frequencies. It is also not the intention of the report to demonstrate that the use of dynamic methodologies such as DFM and Markov/CCMT methodology will produce significant changes in the predicted Top Event frequencies. Again, it has already been shown in the literature that they may, as described in detail in NUREG/CR-6901. Even if the implementation of the methodologies on some selected upgrades do not produce results significantly different from those that would be obtained from the conventional ET/FT approach, it would not imply that this conclusion would be necessarily true for all future upgrades and/or reactors. Another point to keep in mind regarding the suitability of the conventional ET/FT approach for all digital I&C systems is that the tools to implement the conventional ET/FT approach (e.g. [8-10] are not designed to handle the non-coherence that may arise from the multitasking and recuperation capability of digital I&C systems. Finally, the selection of the DFM and Markov/CCMT methodology for this study is based on the requirements stated in NUREG/CR-6901 and, in particular, the relevancy of applications encountered in the available literature to reactor protection and control systems. It should not imply that they are the only dynamic methodologies available for the reliability

modeling of digital I&C systems. For example, a combination of unified modeling language (UML) notation (for pure qualitative analysis), colored Petri nets (for detailed analysis of behavior and communication patterns) and BBNs for optimization of the Petri nets have been successfully used for complex software-intensive systems [11-14].

1.2 Background

In 1994, the U.S. Nuclear Regulatory Commission (NRC) Advisory Committee on Reactor Safeguards (ACRS) recommended and subsequently NRC commissioned a study by the National Academy of Science (NAS) to study the use of digital systems in nuclear power plants. Simultaneous with this study the NRC Regulatory staff initiated a revision of Chapter 7 (I&C) of the Standard Review Plan (SRP) (NUREG-0800)[15]. The primary objective of the revision was to incorporate review guidance for digital systems used in safety-related and non safety-related systems in I&C in nuclear power plants. The NAS report published in 1997 [16] recommended that "The U.S. NRC should strive to develop methods for estimating failure probabilities of digital systems, including COTS software and hardware for use in probabilistic risk assessment" and indicated that "These methods should include acceptance criteria, guidelines, and limitations for use and any needed rationale and justification." The ACRS issued a Letter Report in 1997 that supported this recommendation [17].

As part of a cooperative agreement between the NRC and The Ohio State University (OSU), a study was initiated in 2004 to develop both policies and methods for inclusion of reliability models for digital systems into current generation nuclear power plant PRAs. The findings of the study regarding the current state of reliability modeling methodologies for digital systems were published as NUREG/CR-6901 (*Current State of Reliability Modeling Methodologies for Digital Systems and Their Acceptance Criteria for Nuclear Power Plant Assessments*) in February 2006 [2]. In agreement with

- the conclusion of the NAS report [16] that digital I&C systems (and digital systems in general) should not be addressed only in terms of hardware or software,
- the findings of the North Atlantic Treaty Organization (NATO) Advanced Research Workshop on the Reliability and Safety Assessment of Dynamic Process Systems [18], and,
- the findings of the Department of Energy sponsored workshop on Instrumentation, Control and Human Machine Interface Technology [19],

NUREG/CR-6901 reviews the state-of-the-art in the reliability modeling of digital I&C systems as integrated hardware/software/firmware systems whose failure modes may be statistically interdependent due to coupling through the monitored/controlled process (Type I interactions) and/or due to communication between different components, multi-tasking and multiplexing (Type II interactions). Some specific issues relevant to Type II interactions are the following [2]:

- Digital I&C systems rely on sequential circuits which have memory. Consequently, digital I&C system outputs may be a function of system history, as well as the rate of progress of the tasks.

- Tasks may compete for a digital controller's resources. This competition requires coordination between the tasks and may lead to problems such as deadlock and starvation[1].
- The choice of internal/external communication mechanisms for the digital I&C system (such as buses and networks) and the communication protocol affect the rate of data transfer and subsequently the digital I&C system reliability and robustness.
- The ability to coordinate multiple digital controllers directly and explicitly may necessitate a finer degree of communication and coordination between the controllers.
- A digital controller can remain active and not only react to data, but can anticipate the state of the system.
- Tight coupling and less tolerance to variations in operation increases the digital I&C system sensitivity to the dynamics of the controlled physical process and hence its representation in the digital I&C system reliability model.

The NUREG/CR-6901 presents a characterization of digital I&C systems that is particularly important to this report. For completeness, this characterization is reiterated and examples from available industry reports are included in Section 1.2.1. Sections 1.2.2 and 1.2.3 present overviews of the available methodologies for modeling Type I and Type II interactions, respectively. Section 1.2.3 presents a subjective assessment of these methodologies with respect to the requirements a methodology must meet for the reliability modeling digital I&C systems.

1.2.1 Characterization of Analog and Digital Systems

Digital systems distinguish themselves from other control and instrumentation systems due to the presence of active hardware and software components, their capabilities and limitations, and the manner in which they are interconnected. This section summarizes the germane analog and digital system features in Section 1.4.1. A brief review of operational experience with digital I&C system failures and their implications on fault modeling are presented in Section 1.4.2.

1.2.1.1 Analog vs. Digital Instrumentation and Control Systems

As presented in NUREG/CR-6901 [2], the available literature focused on nuclear power plant safety assessment generally describes a watershed associated with the migration from 'analog' to 'digital' instrumentation and control systems. This section defines and characterizes what this report considers as analog and digital systems for a meaningful comparative analysis of their control characteristics and reliability. While the following examples may not capture fully all salient characteristics of such systems, it is believed the following discussion provides common definitions and assumptions upon which to base analysis and conclusions throughout the remainder of this report.

1.2.1.1.1 Characteristics of Analog Instrumentation and Control Systems

[1]Deadlock and starvation occur due to granting exclusive access to shared resources in multitasking systems. Deadlock occurs when a set of processes (or tasks) are blocked because each waits for a resource that can only be released by an already blocked process in the same set. Starvation occurs when all members of the set run indefinitely, but with no progress.

Analog I&C systems may be characterized as ‘hard-coded’ or ‘hard-wired’ systems, that is “having a direct physical connection, such as by wire or cable” or “controlled by wiring of the hardware, rather than by software”[20]. Alternatively, hard-coded may be defined as “an aspect of an electronic circuit which is determined by the wiring of the hardware, as opposed to being programmable in software or controlled by a switch” [21].

There are several other germane characteristics of analog control I&C systems. Analog I&C systems contain only combinatorial logic. Combinatorial logic contains no logic loops. In addition, the output from combinatorial logic depends only on the current value of the inputs–there is no history kept within the controller. Analog controllers generally contain “random logic,” that is, there is no regularity to the control logic. Therefore, algorithmic logic, as exemplified by finite state machines, does not apply to random logic controllers [22]. In this manner, analog controllers are reactive in that the controllers act on input measured through sensors. Finally, pure analog I&C systems perform their functions continuously and the data values and their internal representations are continuous waveforms.

It is noted that analog I&C systems may contain elements that exhibit digital characteristics. For example, a control valve may have only two positions—open and closed. In analyzing systems with such characteristics, the analysts are not concerned, in general, whether the valve may be opened partially or not. In fact, such systems are designed to include components that have only two positions.

In addition, analog I&C control systems that include electrical components historically have contained vacuum tubes, relays, transistors, etc. These components are types of electrical switches whose only states are ‘on’ and ‘off.’ These components have been used to build combinatorial circuits for many years.

Finally, ‘ladder logic’ control systems have been used for many years for controlling machinery, pumps, fluid levels, etc. Ladder logic systems are a type of combinatorial circuit built originally from discrete components such as relays, resistors, transformers, etc. [23]. Ladder logic systems are still used extensively. However, the mechanisms for realizing ladder logic have changed dramatically over the years. Currently, ladder logic control systems, traditionally considered to be analog controllers, are realized in programmable logic controllers (PLCs). Such devices are actually digital processors masquerading as analog devices.

1.2.1.1.2 Characteristics of Digital Instrumentation and Control Systems

Perhaps the greatest advantages for migration from analog to digital controllers are cost and flexibility. From their inception, microprocessors demonstrated significant design and fabrication cost advantages over custom-design random logic systems [24]. In addition, the programmability of these systems permits the use of standard hardware components while allowing customization of functionality through programming. Unlike the analog devices examined in the previous section, digital devices are not limited to single functions that are determined by the hard-wired connections to the outside world. Digital stored-program control devices may be specialized to the tasks at hand by loading different programs depending on the responsibilities required of them. Such programs are actually “codification” of processes that may have been performed through random logic, human intervention, or a combination thereof previously.

Microprocessors and the resulting digital I&C systems constructed from them are not combinatorial logic machines. Rather, they rely on sequential circuits—they have memory. Consequently, their outputs may be a function of system history as well as the measured current state of the world, based on sensor inputs. In addition, sequential circuits have a timing mechanism (clock) associated with them. The clock determines the rate of progress for a given task as well as coordinating tasks that may compete for a digital controller's resources.

The same external sensors and actuators may be connected to a digital controller and an analog controller through the same sets of wires. However, in the digital universe one must be careful to insure the sampling rate used for analog to digital conversion is sufficient to overcome the creation of artifacts that may result from too low a sampling rate [25]. Also, the sampling rate, algorithm, and processor speed must be selected and matched carefully to ensure that the response time performance requirements are met.

There exist alternate mechanisms for connecting digital controllers to the outside world, such as buses and networks. Such options are not available to analog controllers. The selection of connection mechanism and the communication protocol chosen affect the rate of data communications as well as its reliability and robustness.

The ability for digital I&C systems to have exclusive access to resources, suspend processing (waiting) while holding exclusive access resources, inability to preempt another digital I&C system from holding a resource, and a possibility of circular waiting for resources also implies the need to analyze the system for potential problems such as deadlock or starvation that may result.

Digital instrumentation and control systems represent data internally as discrete values. In that respect, they are approximations of the "real world" (analog) values that exist outside of the digital elements. Discrete representations of analog values may introduce errors, aliasing, or artifacts. In addition, digital I&C systems perform their computations based on an internal clock—the computation process itself is discrete, unlike the continuous computation performed in analog systems.

Digital instrumentation and control systems comprised of only a one processor are likely to exhibit only Type I interactions. Digital instrumentation and control systems comprised of multiple communicating/cooperating processors using networks, shared memory, or other data communication approaches, may exhibit Type II interactions. Consequently, appropriate analysis techniques must be developed for and applied to such systems.

1.2.1.2 Digital Instrumentation and Control System Experience

Experience to-date with digital instrumentation and control systems in nuclear power plants has identified several challenges as a consequence of their digital nature. Industry experience with digital instrumentation and control systems indicates numerous incidents have been attributed to the use of digital systems located in various plants [26-32].

For example, [27] discusses an oscillation power range monitor (OPRM) slave module that randomly reset (causing the module trip channel to be unavailable for one minute) due to a software watchdog timer trip. The OPRM modules include both a master and slave module

which monitor individual Local Power Range Monitor (LPRM) signals. These signals originate at different points within the core and are used by the OPRM modules to detect power fluctuations within a boiling water reactor (BWR). When the slave module resets, it causes the master module to reconfigure itself due to the now invalid data from the slave module. The period in which both the master module reconfigures itself and the slave module resets is 30 seconds. During this time period, no protection monitoring is performed by either the master or the slave module.

Reference [31] discusses an issue in the James A. FitzPatrick nuclear power plant. In this incident, one of the modules in the torus temperature monitoring system was running an incorrect algorithm for validating resistance temperature detector (RTD) readings. The incorrectly implemented algorithm rejected RTD readings that deviated more than 10% from the average reading. A correctly implemented algorithm would have rejected those readings which deviated more than 100% from the average reading. This issue would have affected bulk temperature readings if localized torus heating had occurred.

Reference [32] presents a situation in which a nuclear power plant was operated at a lower-than-designed feedwater temperature. The lower-than-designed feedwater temperature could have caused the plant to exceed its rated power level. A databank constant in the digital system was set to zero based upon instructions from the vendor after the removal of other components which used the constant. Neither the vendor nor the operators realized that the constant was also in use by the feedwater temperature compensation calculation within the 3D Monicore (core monitoring) system. The change in the constant caused the system to incorrectly calculate the power level.

These incidents summarized above demonstrate that digital systems may introduce Type II interactions among the controllers and processes via explicit communication or inter-dependencies. Additionally, such digital I&C systems may embody types of failures different than analog systems (Section 2.3). Consequently, digital I&C systems may require increased model fidelity to capture their significance fully.

There have been several research efforts to create design techniques and tools to address the unique digital system features within critical systems. Rushby [33] identifies areas of critical digital system design and implementation, and discusses techniques for designing systems for dependability, safety, security, and real-time needs of critical systems. These tools and techniques provide ways to both specify needed properties and to add specific features desired, such as fault tolerance or self-stabilization. For example, if dependability is desired, one must include in the design the types of failures that the systems should be able to tolerate (a fault model). Based on the fault model, different components of the system may be designed to tolerate different types of faults. By composing the different components appropriately, it is possible to create a system with the needed fault tolerance capability. These additional system capabilities may increase confidence in the digital system. However, the additional system capabilities create the need for higher fidelity modeling to accommodate the new capabilities introduced in the system. For a more thorough survey and discussion, see [33].

1.2.2 Methodologies for Modeling Type I Interactions

The NUREG/CR-6901 has identified three main categories of methodologies capable of accounting for Type I interactions in the reliability modeling of digital I&C systems [2]:

- Continuous-time methods
- Discrete-time methods
- Methods with visual interfaces

While the methods with visual interfaces are also either continuous or discrete time methods, the reason they are listed separately is because the availability of a visual interface is usually regarded as rendering them more user-friendly. Continuous-time methods consist of:

- the continuous event tree (CET) method[34]
- the continuous cell-to-cell-mapping (CCCM) method[35]

These methods can use accurate descriptions of system dynamics to yield the probability of finding the system at a specified location in the system state-space at a specified time in a specified configuration. The discrete-time methods include the following:

- DYLAM (Dynamical Logical Methodology)[36, 37]
- DETAM (Dynamic Event Tree Analysis Method)[38]
- DDET (Dynamic Discrete Event Tree) [39]
- ADS (Accident Dynamic Simulator)[40]
- ISA (Integrated Safety Assessment) [41]
- DDET/Monte Carlo (MC) hybrid simulation [42]
- CCMT (Cell-to-Cell Mapping Technique)Aldemir[5]

DYLAM, DETAM, DDET, ADS and ISA are dynamic event tree generation techniques. They use a simulator to model the deterministic dynamic system behavior with a set of branching rules and associated probabilities to generate and quantify the likelihood of possible scenarios of system evolution following an initiating event. DDET/MC generates the branchings with a DDET engine and follows them using Monte Carlo sampling for uncertainty quantification of the likelihood of possible scenarios. The CCMT is based on a discrete time version of CCCM and follows the probabilistic evolution of the system using a Markov chain.

Methods with visual interfaces include:

- Petri nets [43, 44]
- DFM [3, 4]
- the event-sequence diagram (ESD) approach[45]
- the GO-FLOW methodology [46, 47]

Petri nets are similar to finite state machines with transitions, arcs and nodes (places). Arcs connect either transitions to nodes or nodes to transitions. Petri nets also use tokens that can move when the Petri net is executed. A token is consumed by a transition. When a transition fires, it produces tokens in places that it connects to and consumes one token in each of the places that connect to it. In order for a transition to fire it must have at least one token on each of its input places. Petri nets with the addition of a set of transitions that fire at random times are called generalized stochastic Petri nets [48, 49]. The DFM is a digraph-based technique. A process variable is represented by a node discretized into a finite number of states. The system dynamics is represented by a cause-and-effect relationship between these states. The DFM is

described in more detail in Chapter 3. The ESD approach uses a 6-tuple of events, conditions, gates, process parameter set, constraint and dependency rules to represent the probabilistic system evolution. The events represent transitions between system states. The probabilistic approach is an extension of the CET approach. The GO-FLOW methodology uses signal lines and operators. The operators model function or failure of the physical equipment, a logical gate, and a signal generator. Signals represent some physical quantity or information.

All these methodologies are referred to as dynamic methodologies because they explicitly account for the time element in system evolution to model the possible coupling of events through the monitored/controlled process. Subject to given failure data and deterministic system model accuracy, the techniques that allow the most accurate and comprehensive modeling of the probabilistic system dynamics are the ones based on the Chapman-Kolmogorov equation including CET, CCCM, CCMT, and ESD approaches. The main challenge with these techniques is their computational complexity, both in model construction and implementation. Another challenge is compatibility with existing PRA structures. The advantage of the dynamic event-tree generation techniques (such as DYLAM, DETAM, DDET, ADS and ISA) is that they are compatible with the existing PRA structure and are able to generate possible scenarios of the system evolution exhaustively. The main disadvantage is that the number of branches increases according to the power law with the number of branch points. Most of the methods with visual interfaces can be regarded as semi-dynamic, because they represent system dynamics qualitatively (e.g.,Petri nets, GO-FLOW) or in a coarse partitioning of the system state space (i.e., in terms of large, small, medium changes in controlled process variables such as the case with DFM). The others have similar capabilities regarding process dynamics, representing it in a semi-quantitative fashion. All the methods with visual interfaces are capable of scenario and cut set outputs. However, cut sets may change with system evolution in time. Petri nets can be converted to fault trees. Again, fault-tree structures may change in time.

1.2.3 Methodologies for Modeling Type II Interactions

While an important feature of feature of digital I&C systems that distinguishes them from analog systems is the presence of software/firmware, there is no consensus in the reliability community about how the reliability of software systems should be modeled, measured, and predicted, and even whether such a concept makes sense for software. In a similar manner to NUREG-CR/6091[2], this report will also treat digital I&C systems as integrated hardware/software/firmware systems. We will consider reliability modeling methodologies for pure software only when they have been applied to mission critical applications or are being researched in the context of mission critical applications.

The methodologies available for the modeling of Type II interactions systems can be grouped as follows:

- Markov methodology [50-52]
- Dynamic flowgraph methodology (DFM)[3, 4]
- Dynamic fault trees [11, 53-55]
- Petri net methodologies [14, 43, 44, 48, 56-58]
- Bayesian methodologies [59-62]
- Test based methodologies [63-70]

- Software metric-based methodologies [71-73]

The references cited above are not exhaustive but are representative of the methodologies proposed for the reliability modeling of digital I&C systems. It should be also noted that the following comparative discussion of the methodologies is based only on the examples cited in literature that may pertain to nuclear power plants. There is no current benchmark system for a common comparison.

The Markov/CCMT methodology (see Chapter 4) represents system topology in terms of system states (e.g. hardware/software/firmware configurations) and transitions between states. Complex Type II interactions can be represented by explicit modeling of software and hardware through fault injection[2] or as a finite state machine. Markov/CCMT models are widely applicable to many types of digital systems, from high availability e-commerce sites to high reliability systems that are analogous to digital I&C systems in nuclear power plants. Two challenges are: a) model size for systems with a large number of states, and, b) determining the transition rates (or probabilities) between states. Some issues regarding the second difficulty are the following [2]:

- Repair rates are not necessarily appropriate to express the ongoing dynamics of digital systems.
- If fault injection techniques are used to obtain the state transition rates, simulating distributed faults and validating their propagation through the system may be difficult tasks.
- Although software inputs may be classified statistically into equivalence classes, values that are statistically or semantically "close" may not be "close" with respect to what the software computes on those inputs.

References [3] and [4] describe how to apply DFM to validate the safety requirements of digital I&C systems. The approach integrates the digital I&C system and the other physical components with the process aspects of the system. The DFM models the physical and software variables by mapping them into a finite number of states. The effects of process, hardware and software/firmware functional behavior (including failures) on the system performance are represented by decision tables. Fundamental issues that may have an impact on the effectiveness of this approach include the difficulty of choosing a proper set of states for each variable and the accuracy of the constructed decision tables. The trade-off is between the accuracy of the model and the size and complexity of the model.

Dynamic fault-trees use timed house events [54, 55] or functional dependency gates [53] to represent the time varying dependencies between basic events. Quantification of dynamic fault-trees is performed using time dependent Boolean logic [54, 55] or Markov models [53]. Applications of dynamic fault-trees mostly include computer based systems [55], including mission avionic systems [11]. Dynamic fault-trees are able to model the sequencing of events in system evolution and have been used to model fault-tolerant systems[11, 54]. However, it is not clear that they can be used to model the differences in system behavior that depend on the exact timing of failure events [2].

---

[2]Fault injection is a process in which deliberate faults are introduced into a system and the system response is observed

Petri nets and their Type I interaction modeling capabilities were described in Section 1.2.1 of this report. Some applications of Petri nets to model Type II interactions include multiprocessor systems [48], network routing [56] and safety critical real-time control systems [74]. Logic gates, including inhibit gates, delay gates and M-out-of-N gates can be modeled using Petri nets [57]. Also, Markov chains can be generated automatically from Petri nets [56, 57] and Petri nets can be used to detect faults in a manner similar to that of employing fault trees [57, 75]. The limitations of Petri nets include the following [2]:

- Stochastic Petri nets assume an exponential distribution of timed firings, which may not be reasonable for certain types of systems.
- Petri nets are limited to systems that can be represented as states and transitions, which implies that the system has to have a finite number of states. Thus, a Petri net could not represent a continuous variable directly. However, [76] shows that systems with continuous variables can be sometimes modeled as fluid or continuous stochastic Petri nets, which can be represented by integral-differential equations very similar to those in [34].
- Even simple systems may result in many parameters and many states which may lead to computational difficulties in processing the Petri net.

A fair number of approaches are encountered in the literature using Bayesian methodologies for quantifying the reliability of software-based safety critical systems [59-62, 77-83] with some targeting nuclear applications [59-62, 77, 78, 80-83]. A subset of these Bayesian methodologies use Bayesian belief networks (BBNs) for better visualization of system topology as well as simplifying incorporation of data from different sources into the model (e.g.[59, 60]). A Bayesian network is a graphical model that efficiently encodes the joint probability distribution for a large set of variables [61] Bayesian methods can be used while testing the system to increase the reliability accuracy incrementally when new testing data are available. Some other advantages are the following [2]:

- Bayesian analysis can be used for both forward and backward inference [77].
- Bayesian analysis allows inferences to be based upon a combination of objective and subjective evidence [61, 77, 81].
- Conservative/optimal stopping rules can be inferred for the operational testing of safety-critical software [62, 80, 82].
- Bayesian analysis allows combining statistical evidence from disparate operational environments [60, 81].
- Efficient computational methods and tools are available for exploring model consequences [59].

In the use of Bayesian methodologies for representing Type II interactions, the choice of priors is sometimes subjective [78]. Also, it is not clear how Bayesian methods can be used to account for Type I interactions.

Test-based methodologies generally use either black-box testing or error history (or reliability growth) models [66]. Software structure can also be represented [63, 71]. Black-box models (e.g.[63, 65]) consider the software associated with a system or subsystem as one "black box," which is characterized by one overall failure rate (referred to a unit of execution time or

execution cycle), regardless of which subfunction(s) the software may be executing. Error history models (e.g., [68-70]) analyze the software development process to obtain statistical models that can predict future failures. Test-based methodologies are relatively simple to implement and hardware/software/firmware interactions can be accounted for. The main limitation is that testing is a value-added activity with respect to errors in software, i.e., it can only inform the tester of the presence of an error under the tested conditions but not necessarily under untested conditions [2].

Software metric-based methodologies use metrics such as lines of code, function points, defect density, number of unique or distinct operators for a given implementation and number of independent paths in a program to approximate the reliability of software [84]. It has been claimed that software metric-based methodologies can be sufficiently accurate for applications of failure rates of 10^{-4}/year [72, 85]. One advantage of this methodology is that the metrics can be gathered as part of a mature software development process. The main limitation of the approach is that it is only applicable to software and measures the software development process, not the end result of the process. It also assumes that there is a high correlation between the development process and the products resulting from applying the process.

1.2.3 A Subjective Assessment of Available Methodologies

The NUREG/CR-6901 has identified the following requirements a methodology needs to meet for the reliability modeling of digital I&C systems [2]:

1. The model must be able to predict encountered and future failures well.
2. The model must account for the relevant features of the system under consideration.
3. The model must make valid and plausible assumptions.
4. The model must quantitatively be able to represent dependencies between failure events accurately.
5. The model must be designed so it is not hard for an analyst to learn the concepts and it is not be hard to implement.
6. The data used in the quantification process must be credible to a significant portion of the technical community.
7. The model must be able to differentiate between a state that fails one safety check and those that fail multiple ones.
8. The model must be able to differentiate between faults that cause function failures and intermittent failures.
9. The model must have the ability to provide relevant information to users, including cut sets, probabilities of failure and uncertainties associated with the results.
10. The methodology must be able to model the digital I&C system portions of accident scenarios to a level of detail and completeness that non-digital I&C system portions of the scenario can be properly analyzed and practical decisions can be formulated and analyzed.
11. The model should not require highly time-dependent or continuous plant state information.

While no single methodology was found to satisfy all the requirements above, NUREG/CR-6901 has identified the DFM and a combination of the Markov methodology with CCMT as the methodologies with the most potential for near future applications, based on the cases reported

in the literature. In view of the lack of a benchmark system against which methodologies could be compared objectively, NUREG/CR-6901 also recommended that:

- Two benchmark problems should be defined that respectively capture important features of the existing analog I&C systems and their digital counterparts expected to be encountered in risk important nuclear power plant applications.
- The benchmark problems should be used to compare the DFM and the Markov methodologies with regard to the modeling of both Type I and Type II interactions using a common set of hardware/software/firmware states and state transition data.

Chapter 2 describes the first such benchmark problem.

1.3 Review of Current NRC Position on Digital Systems

In response to the Commission PRA policy statement, the NRC staff developed a regulatory structure for using PRA in risk-informed decision making for plant-specific changes [86, 87]. This regulatory guidance is provided in Regulatory Guide 1.174, which provides guidance on the use of PRA findings and risk insights in support of licensee requests for changes to a plant's licensing bases for license amendments and technical specification changes. As part of this guidance a requirement is imposed that the quality of a PRA analysis used to support a license amendment or application will be measured in terms of its appropriateness with respect to scope, level of detail, and technical acceptability. The scope, level of detail, and technical acceptability of the PRA are to be commensurate with the application for which it is intended and the role the PRA results play in the decision process. In other words, the PRA, both in general and for the particular application (digital systems) must adequately model the system being reviewed. It should be made clear that there is a successful path to licensing of digital I&C upgrades and new designs that relies on existing deterministic regulations.

Regardless of how a digital system is modeled, for the modeling to be considered acceptable it must meet certain acceptance criteria. Whatever modeling methods are chosen, they must be able to model the digital I&C systems and the portions of accident scenarios to such a level of detail and completeness that the non-digital I&C system portions of the scenario can be properly analyzed and practical decisions can be made.

1.4 Characterization and Taxonomy of Digital I&C Systems

It has been suggested [33, 88]that by categorizing the various digital systems used in safety critical applications in nuclear power plants, it would be easier to determine which systems should be modeled in the analysis and at what level of detail. The reason for developing a taxonomy or categorization scheme is that digital safety systems often are required to simultaneously satisfy a number of different functions that are characteristic of two or more critical system properties. Additionally, the level of complexity of digital system models that need to be used will depend on the digital system's connections to the rest of the system. As discussed in Rushby [33] it is also useful to be able to identify the various aspects of the systems associated with timing, safety, and fault tolerance requirements.

In Perrrow's analysis [89] of this issue, which is based mostly on safety and system interactions with the larger plant systems, he chose two attributes, 'interaction' and 'coupling', as the basis

of his categorization. Interaction, which in his model can range from linear to complex, refers to the extent to which the behavior of one component in a system can affect the behavior of other components. In a simple, linear system, components affect only other components that are functionally 'downstream;' in a more complex system, a single component may participate in any number of sequences with any number of other components in any order. For example, in a reactor trip system, if a sensor (or group of sensors) indicates the need to trip the reactor, the component (the trip system) will provide an actuation signal to the control rod drive system. However, in a more complex control system, a failure can affect the system it is controlling and the controlled system can provide complex feedback to the controller. Complex interactions can also result from internal communications failures as well. For example, voting logic in a redundant system may need to handshake with redundant channels to avoid timing out a watchdog timer: a complex interaction resulting in an unanticipated plant state may be the result.

Perrow's "coupling," which can range from "loose" to "tight", refers to the extent to which there is slack or flexibility in the system. Coupling is not defined as an independent concept; it is important what the digital system is coupled to. Generally speaking, loosely coupled systems are usually less time constrained than tightly coupled systems, can tolerate operating sequences different than those expected, and may be adaptable to different purposes or to operate under different assumptions than those originally considered. For example, a navigation system on an airliner would be considered a loosely coupled system because the operating state of the navigation system must be adaptable to random events and external operators (the pilot). According to Perrow, the order and exact timing of inputs are not critical to the accurate functioning of loosely coupled systems.

In Perrow's analysis, systems with complex interactions and tight coupling can promote accidents because interactions are hard to understand and predict. Perrow advocates the use of loosely coupled linear systems for safety applications. However, the need for high reliability for safety critical digital systems in the nuclear industry has led to more complicated fault tolerant systems. These systems tend to be more internally complicated, with extensive internal redundancy, error checking, and both internal and external diversity to reduce the potential effects of common cause failures. Additionally, the design of the overall plant frequently requires tighter coupling between a control or protection system and the process it is controlling or monitoring.

Recent research [90] has developed a taxonomy that includes the Perrow coupling attribute (also see Section 1.2, Chapter 8). This taxonomy more completely defines what is meant by tightly coupled and loosely coupled systems, and provides an interaction attribute that is defined based on two sub-attributes, Type I interactions and Type II interactions. As indicated in Section 1.2, Type I interactions are interactions between digital systems such as the reactor protection system and control system and the controlled plant physical processes (e.g., heatup, pressurization) that would produce failure modes that may be statistically interdependent due to coupling through the monitored/controlled process. Type II interactions are hardware/software/firmware interactions within a digital system (e.g., communication between different components, multi-tasking, multiplexing, etc.) which can lead to failure modes that may originate from communication between different components, multi-tasking, and multiplexing.

These two interaction attributes are both important to assessing the level of modeling detail needed to:

1. differentiate between faults that cause function failures and intermittent failures,
2. differentiate between a state that fails one safety feature and those that fail multiple features,
3. demonstrate that there is no important significance to the differences, and
4. insure that no failure modes are missed by the modeling effort (completeness).

However, these attributes cannot assess whether the digital system modeling is adequate in terms of its effect on the accuracy of total plant metrics such as core damage frequency.

A three attribute categorization strategy that could be used as the basis of a performance based requirement on level of modeling detail for digital safety systems is discussed below. The first attribute, digital system complexity, would be based on Type II interactions and an overall digital system size and complexity index. The size and complexity index could be a function point or cyclomatic complexity metric [72, 91]. The attribute would measure be how critical Type II interactions and system complexity is to the fault free representation of the digital system.

The second attribute, digital system interactions/inter-conductivity, would be a combination of coupling and Type I interactions. The attribute measure would be with how the digital system under study interacts with other systems and process parameters within the plant and how important accurately assessing these interactions are to the system reliability and plant risk. Digital systems that are loosely coupled and/or have very few Type I interaction would not interact dynamically with the overall system and would have a low interactions/inter-conductivity score.

The third attribute would be digital system importance. This attribute measure would look at both traditional risk important measures, such as component risk achievement worth [72], and how important the system is for maintaining defense-in-depth. This measure could be implemented by use of a plant integrated decision making panel similar to what is currently done to determine if a system is included in the maintenance rule's (a) 4 requirements [92]. It should be noted that because many of the digital systems included in this effort have significant Type I interactions (and will score high on the system interactions/inter-conductivity attribute), they may not be accurately modeled in the current PRA, so the information associated with their risk importance measure may need to be adjusted accordingly.

One way to implement this conceptual idea is to develop a ranking system that would look at a list of attributes that any particular digital system might process, and based on that list, rank the three attributes. Systems like the reactor protection system (RPS) would have a relatively high risk importance score but likely a lower system complexity score. Systems such as a digital feedwater control system (DFWCS) might have a relatively low system importance score but high system complexity and system interactions/inter-conductivity score (see Fig.1.3.1).

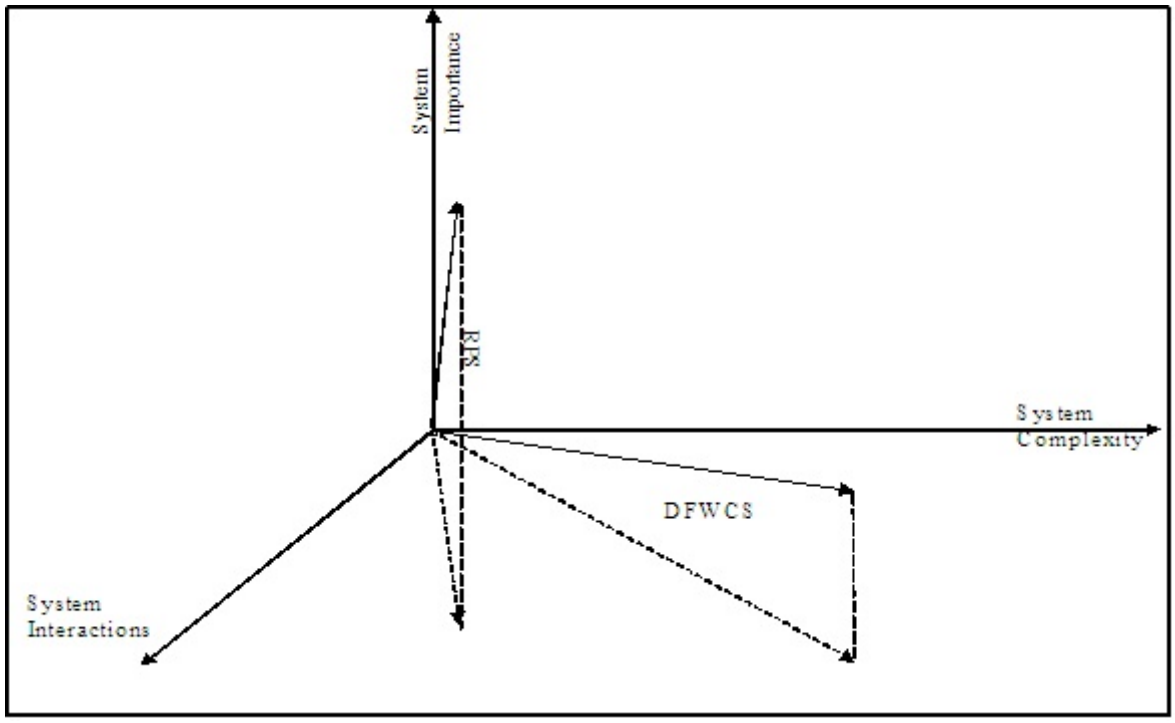

Figure 1.3.1: Conceptual Model of Digital System Modeling Categorizations

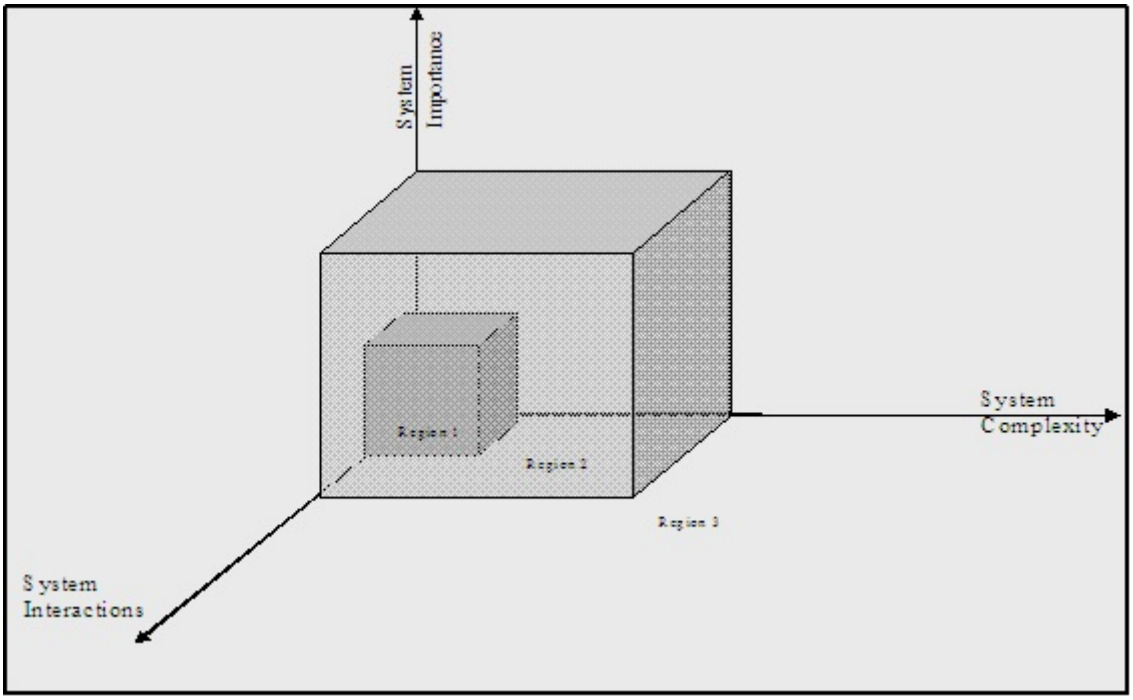

Figure 1.3.2: Possible Implementation of the Conceptual Model

Figure 1.3.2 shows a possible set of regions that would provide guidance as to what level of modeling detail would be needed based on what region a particular system score falls in. For some systems in Region 1 with relatively low complexity, interaction and importance would mean any basic model could be used to support risk informed applications; whereas, systems that fall in Region 3 would need to have complete dynamic models to support their licensing application. The shape of these regions would not need to be as shown and would not even need to be able to be drawn. Figure 1.3.2 simply provides a conceptual model that could be used to explain the concept that, because of their design and function, some digital systems may need to be modeled with at a much higher level of detail than others.

2. DESCRIPTION OF THE BENCHMARK SYSTEM

Notation

C	Fault coverage
t	Time
λ_x	Failure rate in mode *x*
P(t)	Power
x_n	SG*n* level
V_j	Cells that partition the CVSS (*j=1,...,J*)
J	Total number of V_j
n	Component state combination index
N	Number of components state combinations
n_m	Component state index (n_m=*1,...,*N_m)
N_m	Total number of n_m
M	Number of components
E_{Ln}	Level error for SG*n*
C_{ln}	Compensated level for *Sgn*
C_{Fn}	Flow demand for SG*n*
S_{Bn}	BFV position for *SGn*
f_{wn}	Water flow rate into SG*i* (*i* =1,2)
f_{sn}	Steam flow rate out of SG*i* (*i* =1,2)
h_{wn}	Feedwater temperature for SG*i* (*i* =1,2)
$\tilde{S}_{Mn}, \hat{S}_{Mn}$	MFV*n* (*i*n=1,2) position, MFV*n* position set by PDI controller *n* (*n*=1,2)
$\tilde{S}_{Fn}$	Pump speed for FP*n* (n=1,2)
C_{pn}	Compensated power/flux SG*n* (*n* =1,2)
C_{Bn}	BFV*n* (*n* =1,2) demand
r_n	Level setpoint for SGn (*n* =1,2)
β_{Fn}, σ_{Fn}	Controller parameters for FP*n* (*n* =1,2); Arguments indicate table lookup variables
$\lambda_{Mn}, \sigma_{Mn}$	Controller parameters for MFVn (*n* =1,2); Arguments indicate table lookup variables
$\alpha_{Bn}, \beta_{Bn}, \lambda_{Bn}, \mu_{Bn}, \sigma_{Bn}$	Controller parameters for BFVn (in=1,2); arguments indicate table lookup variables
τ_l	Controller parameters (l=1,…,6)
$\hat{N}$	Controller state (failed or operational)
η_{Fn}	FP*n* history (*n* =1,2) maintained as data base
η_{Mn}	MFV*n* history (*n* =1,2) maintained as data base
η_{Bn}	BFVn history (*n* =1,2) maintained as data base

The benchmark system specification is based on the digital feedwater control system for an operating PWR. The architecture, systems, and their interconnections of the system described have evolved from their analog counterparts to digital ones. However, the system described in the following sections is used for illustrative purposes only. It has been generalized to be more representative of this class of systems.

2.1 System Overview

The feedwater system serves two SGs (Fig. 2.1.1). Each SG has its own digital feedwater controller. The purpose of the feedwater controller is to maintain the water level inside each of the SGs optimally within ± 2 inches (with respect to some reference point) of the setpoint level (defined at 0 inches). The controller is regarded failed if water level in a SG rises above +30 and falls below -24 inches. Each digital feedwater controller is connected to a feedwater pump (FP), a main feedwater regulating valve (MFV), and a bypass feedwater regulating valve (BFV). The controller regulates the flow of feedwater to the steam generators to maintain a constant water level in the steam generator.

In addition to the FP, FP seal water system, MFV, and BFV, the feedwater control system contains high pressure (HP) feedwater heaters and associate piping and instrumentation.

In this example, FPs are steam turbine driven, horizontal, double-suction, double volute, single stage, centrifugal pumps. The pumps have a design output of 15,000 gpm at a suction rate of 318.7 psia and a discharge pressure of 118.9 psia. The normal operating discharge pressure is approximately 1100 psig at 100%. The FP is driven by a dual admission, horizontal, 9140 HP, 5350 rpm steam turbine. During plant operation with power greater than 5%, the turbine is aligned to the reheat and main steam system. Steam is supplied from the main steam system during plant startup until reheat steam pressure is sufficient to supply the turbines. If main steam is not available or power is less than 5%, steam can be supplied to the feed pump turbine from the auxiliary steam system. The purpose of the FPs is to pump the feedwater through the high pressure feedwater heaters into the SGs with sufficient pressure to overcome both the SG secondary side pressure and the frictional losses between the feed pump and the SG inlet. The MFV and BFV regulate the amount of feedwater going to the SG in order to maintain a constant water level in the SG.

The MFV is a 10 inch, air operated, angle control valve with 16 inch end connections. This valve is made of steel and has a design rating of 2160 psig at 1000°F. The actuator is a piston type actuator, with separate instrument air supplies to the top and the bottom of the piston. Ball valves control the admission of operating air to the piston for opening and closing operations. The BFV is a 6 inch, air operated, steel control valve.

From an operational point of view, the feedwater control system operates in different modes depending on the power generated in the primary system. These modes are the following:

- Low power automatic mode

- High power automatic mode
- Automatic transfer from low to high power mode
- Automatic transfer from high to low power mode

The low power mode of operation occurs when the reactor operates between 2% and 15% reactor power. In this mode, the BFV is used exclusively to control the feedwater flow. The MFV is closed and the FP is set to a minimal speed. The control laws use the feedwater flow, feedwater temperature, feedwater level in the steam generator, and neutron flux to compute the BFV position. The feedwater level is fed to a proportional-integral (PID) controller[3] using the feedwater temperature to determine the gain. Then this value is summed with the feedwater flow and neutron flux. Essentially, neutron flux and feedwater flow are used to predict required changes in water levels.

High power mode is used when the reactor power is between 15% and 100% reactor power. In this mode, the MFV and the FP are used to control the feedwater flow. The BFV is closed in a manner that is similar to low power mode. The control laws (see Section 2.2) use the feedwater level in the steam generator, steam flow, and feedwater flow to compute the total feedwater demand. This computed value is used to determine both the position of the MFV and the speed of the FP. The FP also uses the other digital feedwater MFV controller's output to compute the speed needed. The feedwater flow and steam flow are summed and fed to a set of PI controller algorithms. The output from these controller algorithms is added to the feedwater level and that result is fed to a PI controller algorithm that uses the steam flow for the controller algorithm's gain.

Each digital feedwater controller is comprised of several components (Fig. 2.1.2) which provide both control and fault tolerant capabilities. The control algorithms are executed on both a main computer (MC) and backup computer (BC). These computers produce output signals for the MFV, BFV, FP and pressure differential indicator (PDI) controllers. The selection of the appropriate signal to be used (from the MC or BC) is determined by the PDI controller. Each of these controllers can forward the MC or BC's outputs to their respective controlled device (i.e. MFV, BFV or FP), or it can maintain the previous output to that device. If the controllers decide to maintain a previous output value to a controlled device, it is necessary for operators to override the controller (Section 2.3).

Transitions between low and high power are controlled by the neutron flux readings. When the system is in low power mode and the neutron flux increases to a point at which high power mode is necessary, the MFV is signaled to open while the BFV closes to maintain needed feedwater flow. The opposite situation occurs when the system is in high power mode and the neutron flux decreases to a point when low power mode is needed.

[3]A PID controller is a proportional-integral-derivative controller.

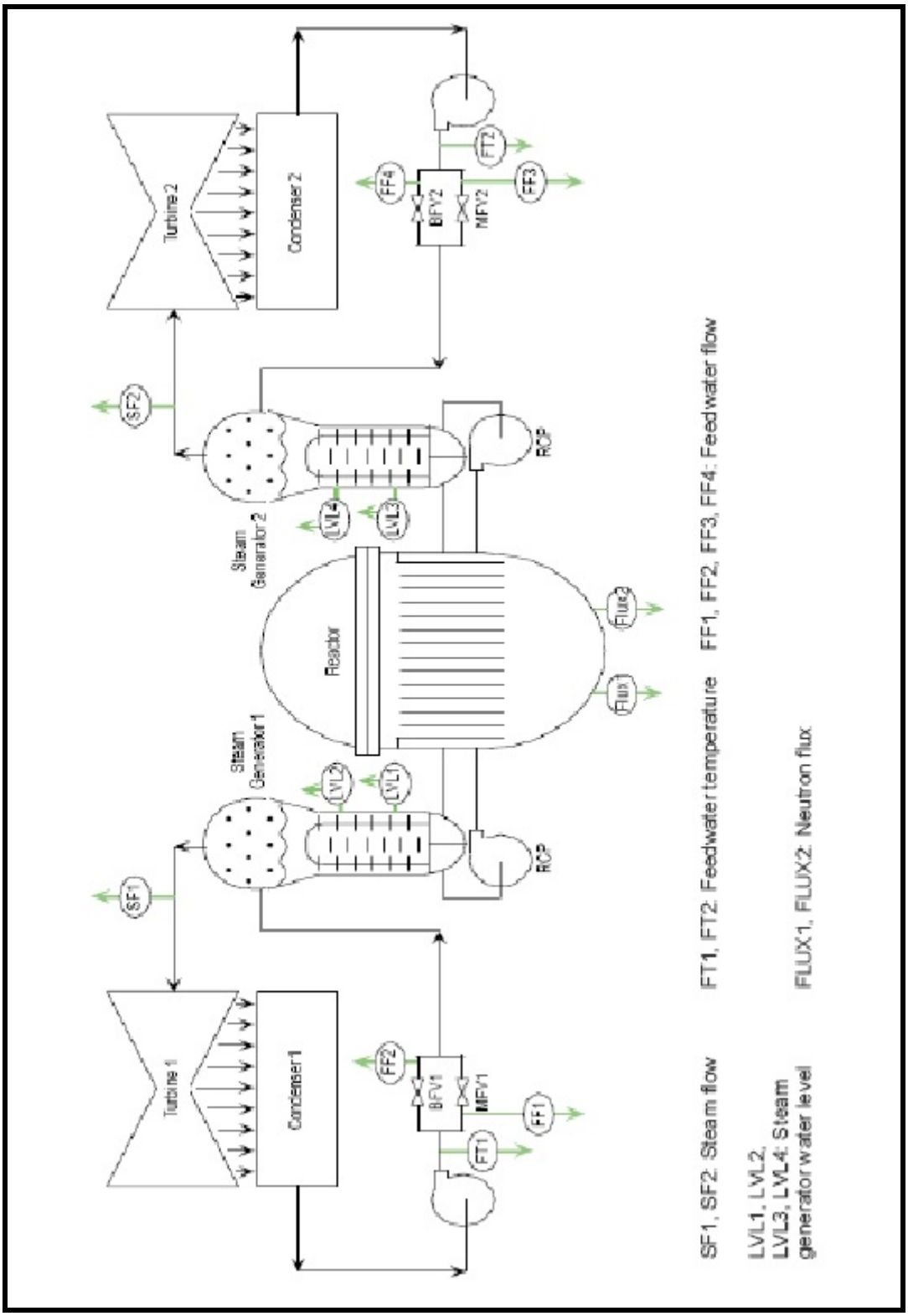

Figure 2.1.1 The Benchmark System Outlay

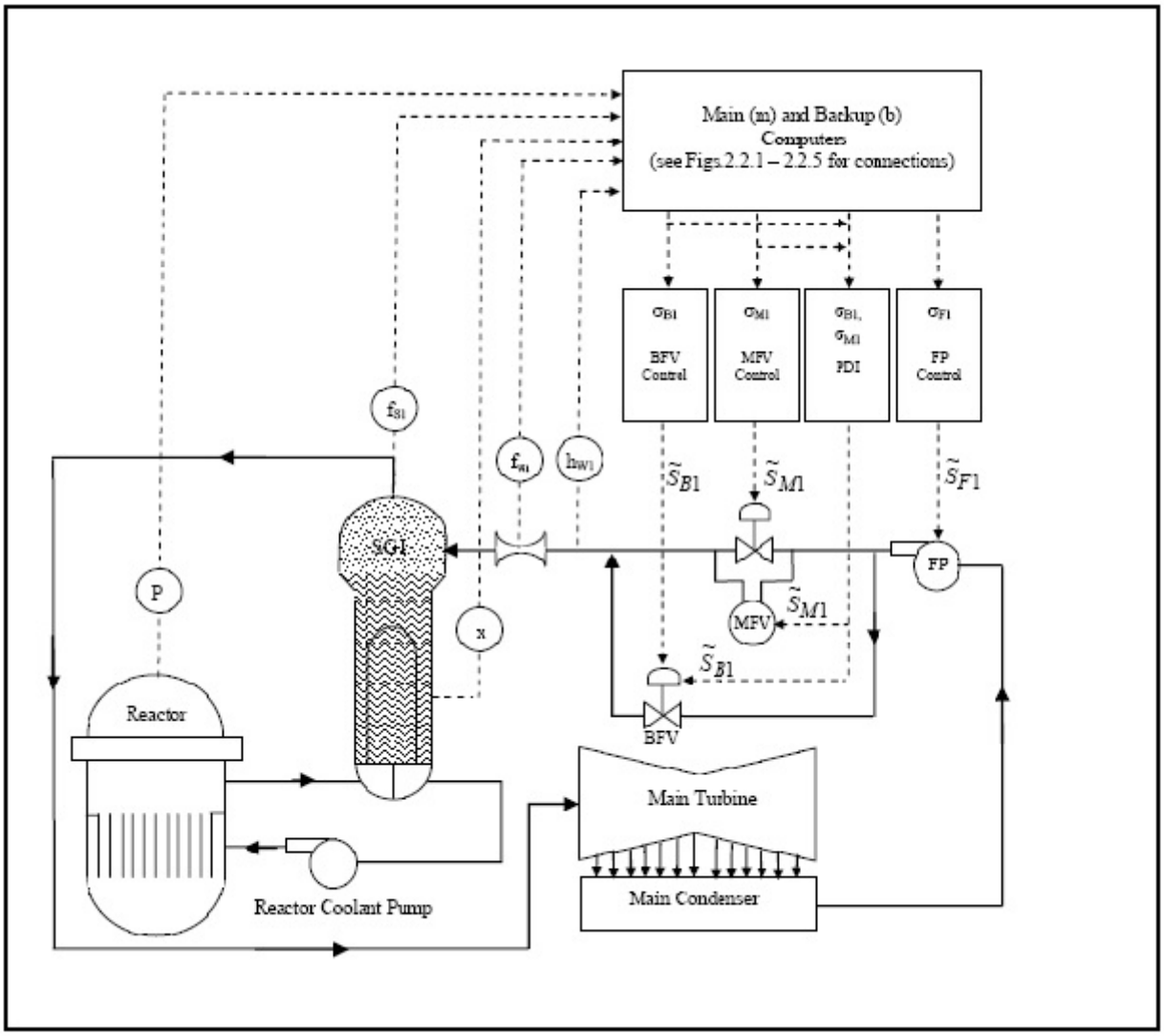

Figure 2.1.2 Detailed View of the DFWCS for SG1

2.2 Detailed View of the Benchmark System

This section describes the DFWCS at a greater level of detail. In particular, the physical connections between the sensors, computers, controllers and actuated devices (i.e., MFV, BFP and FP) are examined. In addition, the control laws are stated and the fault tolerant features of the architecture are described.

2.2.1 Physical Connections for the DFWCS

The DFWCS obtains information about the state of the controlled process through the use of several sensors that measure feedwater level, neutron flux, feedwater flow, steam flow, and feedwater temperature (Fig.2.1.2). In an actual conversion from analog to digital I&C, additional sensors may be employed for redundancy. However, it may not be feasible to increase the number of sensors if modification to the containment building is needed to run the additional electrical lines. This challenge may affect the steam flow, feedwater temperature, and neutron flux sensors. In that respect, the sensors for the companion steam generator may be used to provide redundancy instead of including two sensors for each of the MC and BC. As shown in Figs. 2.2.1-2.2.5, the sensor signals are routed to provide information to both the MC and BC. Setpoint data is delivered from the MFV controller to the MC and BC through an analog signal.

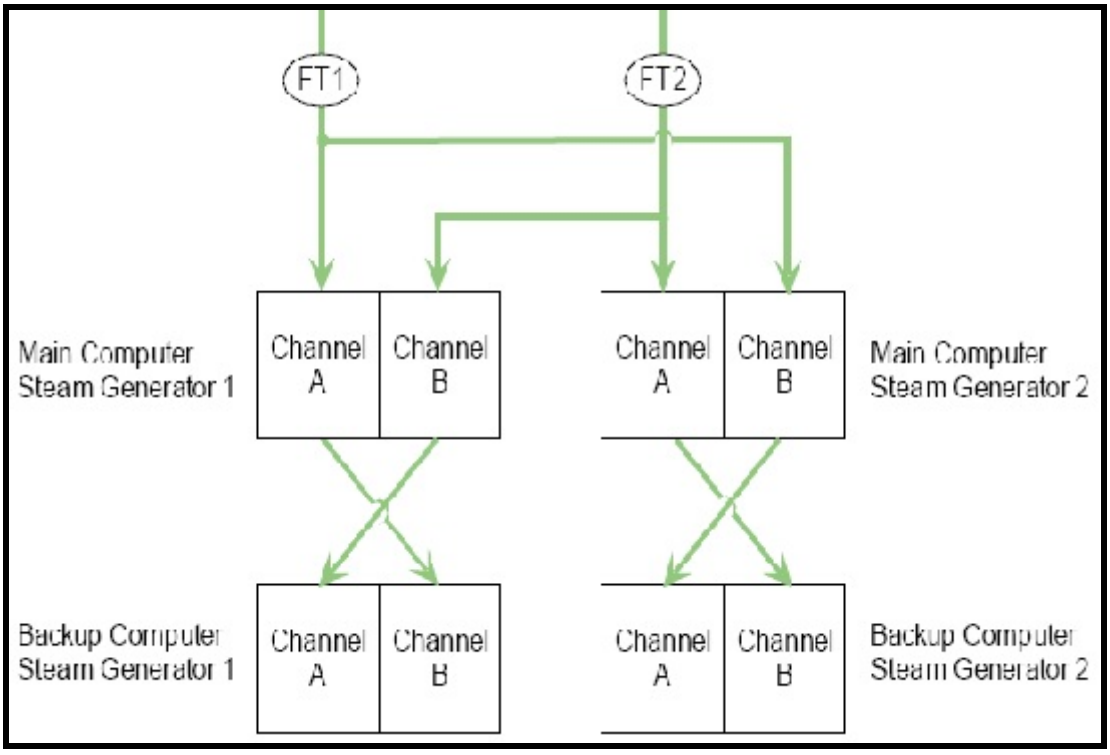

Figure 2.2.1 Feedwater Temperature Sensor Signals

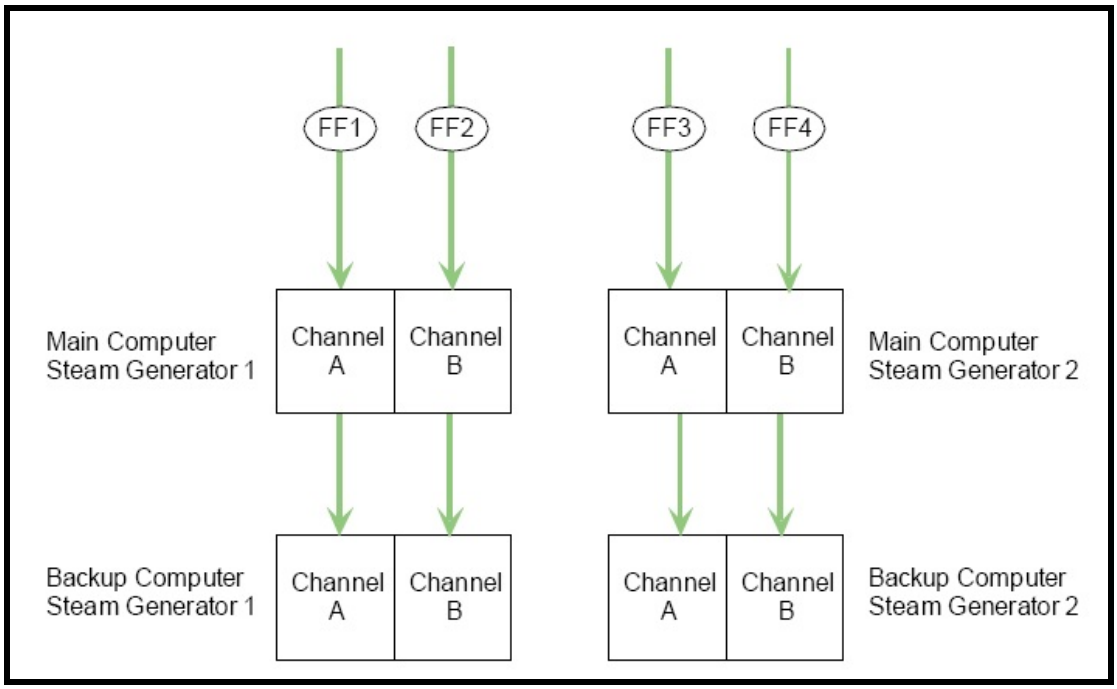

Figure 2.2.2 Feedwater Flow Sensor Signals

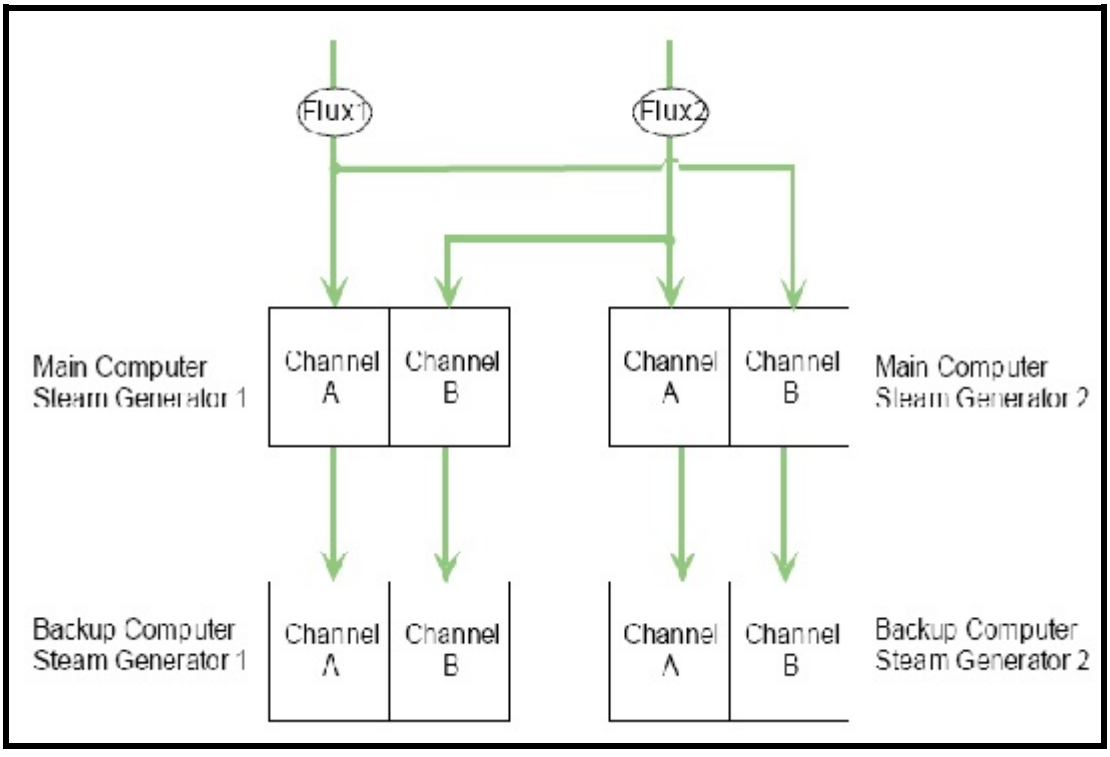

Figure 2.2.3 Neutron Flux Sensor Signals

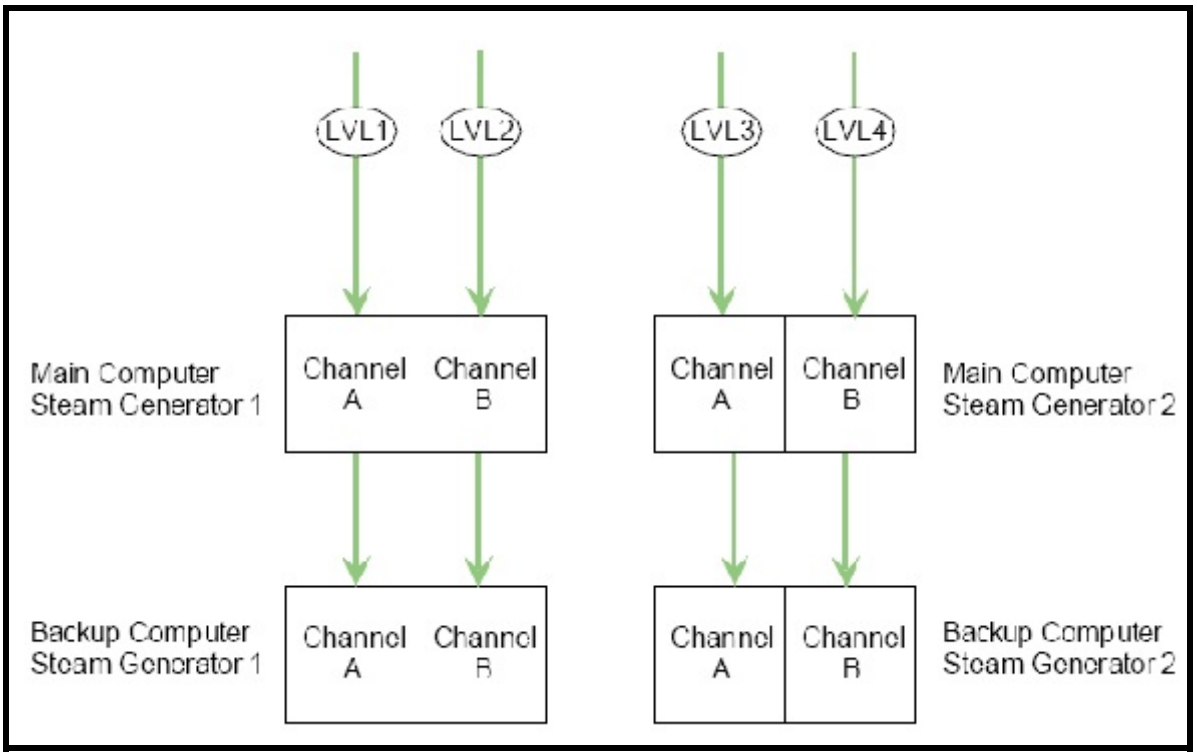

Figure 2.2.4 Feedwater Level Sensor Signals

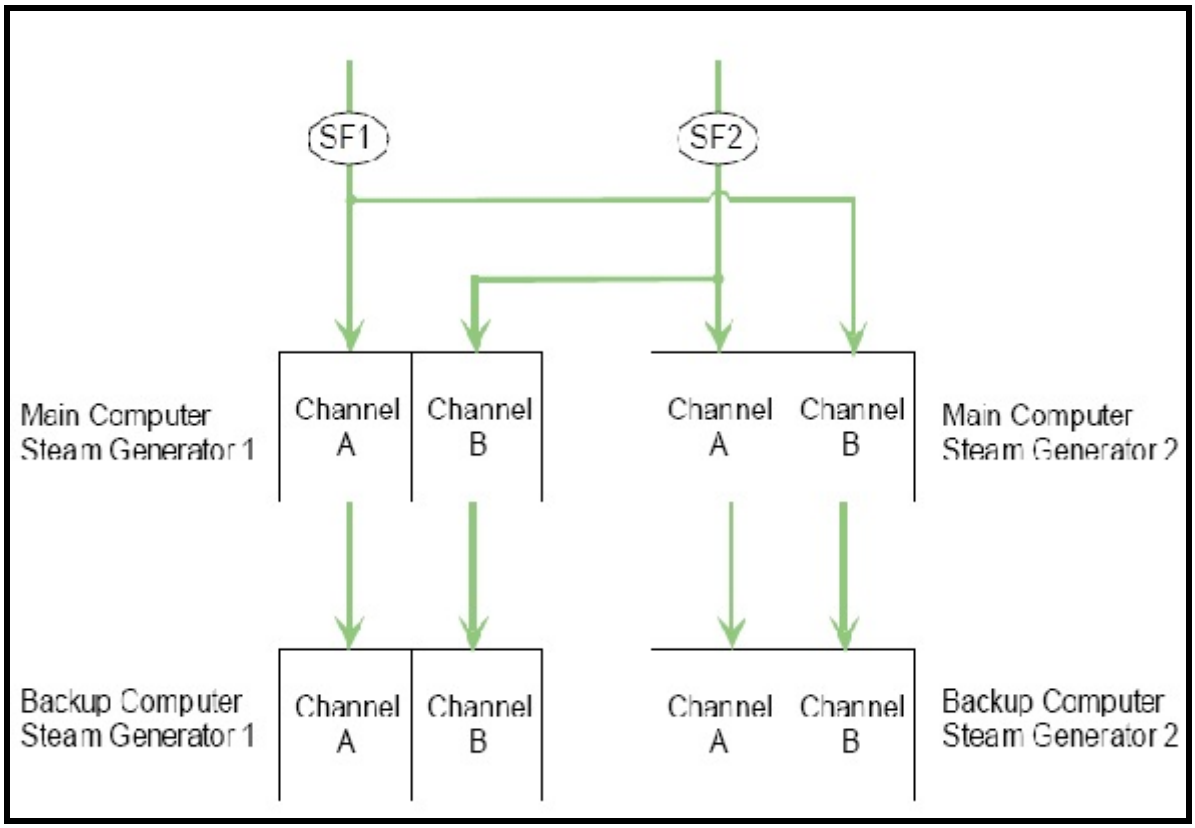

Figure 2.2.5 Steam Flow Sensor Signals

The DFWCS components are connected together in several different ways as shown in Figs. 2.2.6 and 2.2.7. First, both the MC and BC provide input signals to the MFV, BFV and FP controllers through an analog control signal and failure status signals. The MFV, BFV, and FP controllers are configured within the DFWCS to share status information. The PDI controller serves as a backup for the MFV controller by sampling the output of the MFV controller. If the MFV controller output is lost, the PDI will send the last good MFV controller signal to the MFV. The PDI controller also shares status information with the MFV, BFV and FP controllers.

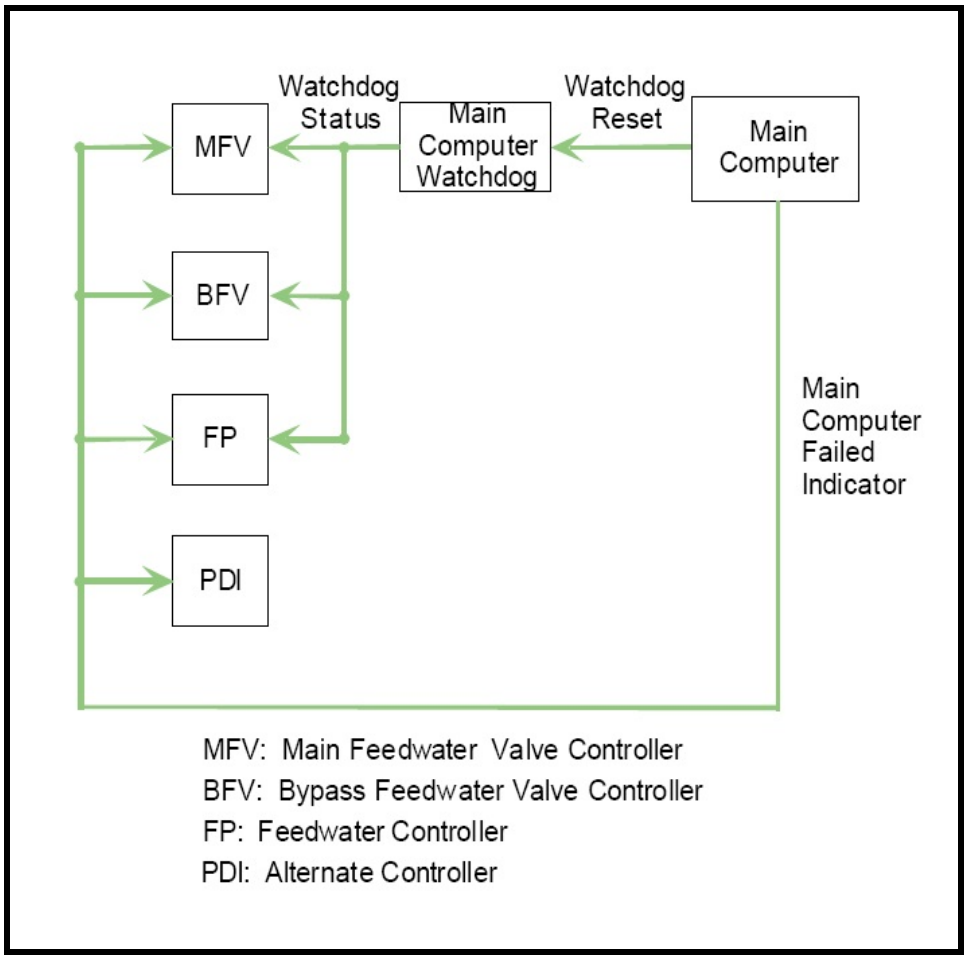

Figure 2.2.6 Digital Feedwater Controller Status Interconnections for MC

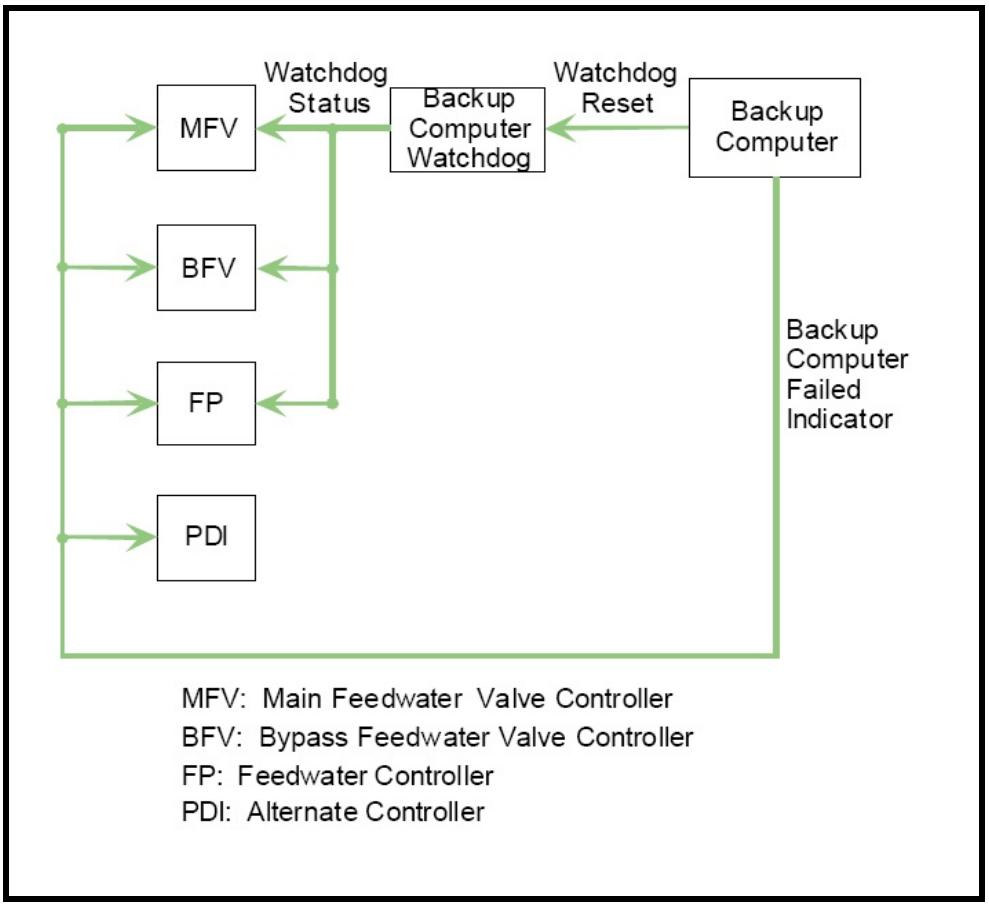

Figure 2.2.7 Digital Feedwater Controller Status Interconnection for BC

2.2.2 Control Laws

The control laws for the feedwater controller for SG*n* (*n*=1,2; see Fig. 2.1.2) under normal system operation are taken from the control algorithm of the DFWCS of an operating PWR and can be expressed as follows:

Rate of level change: $$\frac{dx_n}{dt} = A(f_{wn} - f_{sn}) \quad (2.2.1)$$

Flow Demand: $$C_{Fn}(t) = \beta_{Fn}(f_{sn}) \int dt[r_n - C_{Ln}(t) + E_{Fn}(t)] - \lambda_{Fn}(\sigma_{Bn}) \quad (2.2.2)$$

Compensated Water Level: $$\tau_2 \frac{dC_{Ln}}{dt} = -C_{Ln} + x_n + \tau_1(f_{wn} - f_{sn}) \quad (2.2.3)$$

Compensated Flow Error: $$\tau_6 \frac{dE_{Fn}}{dt} + E_{Fn}(t) = \tau_7[\frac{df_{wn}}{dt} - \frac{df_{sn}}{dt}] \quad (2.2.4)$$

BFV Demand: $$C_{Bn}(t) = \upsilon_{Bn}\alpha_M + \upsilon_{Bn}C_{pn}(t) + \beta_{Bn}(h_{wn})\int dt[r_n - C_{Ln}(t)] - \lambda_{Mn}(\sigma_{Mn}) \quad (2.2.5)$$

Compensated Power: $$\tau_4 \frac{dC_{pn}}{dt} = -C_{pn}(t) + p_n + \tau_5 \frac{dp_n}{dt} \quad (2.2.6)$$

FP Demand: $$\sigma_{Fn}(t) = \begin{cases} \sigma_{Fn} & \text{If Low Power Operation} \\ \sigma_{Fn}(\max(C_{Fn}, \sigma_{Mn}^{-1}(C_{Fn})) & \text{If HighPower Operation} \end{cases} \quad (2.2.7)$$

MFV Demand: $$\sigma_{Mn}(t) = \begin{cases} \sigma_{Mn}(C_{Fn}) & \text{If High Power Operation} \\ 0 & \text{If Low Power Operation} \end{cases} \quad (2.2.8)$$

BFV Demand: $$\sigma_{Bn}(t) = \begin{cases} 0 & \text{If High Power Operation} \\ C_{Bn}(t) & \text{If Low Power Operation} \end{cases} \quad (2.2.9)$$

FP Speed: $$\widetilde{S}_{Fn} = \begin{cases} \sigma_{Fnm} & \text{Main CPU Operational} \\ \sigma_{Fnb} & \text{Main CPU Operational, Backup CPU Failed} \\ \eta_{Fn} & \text{Main CPU Failed, Backup CPU Failed} \end{cases}$$

MFV Position: $$\widetilde{S}_{Mn} = \begin{cases} \sigma_{Mnm} & \text{Main CPU Operational} \\ \sigma_{Mnb} & \text{Main CPU Failed, Backup CPU Operational} \\ \eta_{Mn} & \text{Main CPU Failed, Backup CPU Failed} \end{cases}$$

BFV Position: $$\widetilde{S}_{Bn} = \begin{cases} \sigma_{Bnm} & \text{Main CPU Operational} \\ \sigma_{Bnb} & \text{Main CPU Failed, Backup CPU Operational} \\ \eta_{Bn} & \text{Main CPU Failed, Backup CPU Failed} \end{cases}$$

PDI Decision: $$\hat{S}_{Mn} = \begin{cases} 0 & \widetilde{S}_{Mn} > 0 \\ \eta_{Bn} & \text{Otherwise} \end{cases} \quad (2.2.13)$$

Also, all sensor inputs are averaged before being used by the control laws. For example, the feedwater level for SG1 is the average of the two feedwater level sensors LV1 and LV2 (see Fig. 2.1.1).

Rate of feedwater level change is given by Eq. (2.2.1). In Eq. (2.2.1) the water inflow rate f_{wn} into SG*n* (see Fig. 2.1.2) depends on the MFV and BFV positions and FP speed, respectively, in general. The steam flowrate f_{sn} is determined from the physical process equations modeling the mass and energy transfer in SG*n* (see Appendix A). These flow rates are found from

$$f_{wn} = \left(\sum_i \dot{m}_{Fi}\right)_n \quad (n = 1,2) \tag{2.2.14}$$

and

$$f_{sn} = \left(\sum_i \dot{m}_{Gi}\right)_n \quad (n = 1,2) \tag{2.2.15}$$

where the subscript *i* denotes the number of inlets and exits to *SGn* (*n*=1,2). The right hand sides of Eqs. (2.2.14) and (2.2.15) are obtained from the solutions of the equations in Appendix A for the appropriate upper-lower SG*n* (*n*=1,2) combination (also see Section 2.2.3). For the benchmark system under consideration there is one inlet and one exit for each steam generator and *i*=1.

Equations (2.2.2) - (2.2.4) compute the flow demand for high power mode for the feedwater controller. The dynamic gain $\beta_{Fn}(f_{sn})$ $\lambda_{Fn}(\sigma_{Bn})$ in Eq. (2.2.2) are obtained from a lookup table on the steam flow rate and BFV opening, respectively. Equations (2.2.5) computes the BFV demand for low power mode. The dynamic gain $\beta_{Bn}(h_{wn})$ $\lambda_{Mn}(\sigma_{Mn})$ in Eq. (2.2.5) are obtained from a lookup table on the feedwater temperature and the MFV opening respectively. The subscripts *m* and *b* in Eqs. (2.2.10)-(2.2.15) refer to signals from the main and backup CPUs respectively. The η_{Fn} η_{Mn} η_{Bn} and in Eqs. (2.2.10)-(2.2.15) denote history data for the FP, MFV and BFV positions, respectively. If both of the MC and BC are failed, these data are used to determine the FP, MFV and BFV positions.

2.2.3 Steam Generator Simulation Package

As indicted in Section 2.2.2, f_{wn} and f_{sn} (see Eqs. (2.2.14) and (2.2.15)) are determined from the mass and energy transfer in *SGn*. In that respect, the DFWCS behavior is intimately related to the physical processes in *SGn*. The SG simulation package used in NUREG/CR-6465 [4] was coupled with the control system defined by Eqs. (2.2.1) - (2.2.13) to model the mass and energy transfer in *SGn*. This steam generator simulation package, developed as a Borland C++ project, is based on a vertical U-tube steam generator typical of a two loop PWR. The simulation package consists of a steam generator model, a main feedwater and auxiliary feedwater system models and a steam header model. The steam header model is needed to determine f_{sn} in Eq. (2.2.15). The auxiliary feedwater system model may be needed in determination of the DFWCS response following loss of main feedwater flow or reactor trip at

full power. These models will be discussed in greater detail in the following Sections 2.2.3.1 - 2.2.3.3.

The SG simulator package provides the following process parameters to the control system:

- Heat flux
- Steam flow
- Feedwater flow
- Steam generator level
- Feedwater temperature

The control system in turn processes these inputs through the control laws and outputs the following processor parameters to the SG simulation package:

- Feedpump speed
- Main flow valve position
- Bypass valve position

2.2.3.1 Steam Generator Model

A schematic of the steam generator model is shown in Fig. 2.2.8. Reactor coolant enters the SG hot leg plenum, flows through the SG tubes to the cold leg plenum, and enters the primary system cold leg. While flowing through the tubes, heat is transferred from the primary coolant to the SG secondary (shell) side and boils the secondary coolant.

The shell side of the steam generator consists of an evaporative section and a steam drum. The evaporative section contains the U-tubes, and is located in the lower shell, while the steam drum houses the steam separator and dryer equipment. The steam drum section has a larger overall diameter than the evaporative region. There is a flow restrictor at the top of the steam drum where the steam line connects to the SG.

The shell side of the steam generator is modeled as two non-equilibrium regions (Appendix A) separated by a moving boundary which is the SG level (see Fig. 2.2.8). The simulation model recognizes the different flow areas of the evaporating and steam drum sections. As the level moves between the two sections, the model accounts for the flow area change when computing SG level. The governing equations for the shell side of the SG are given in Appendix A and model the following:

- Conservation of mass in each region
- Conservation of energy in each region
- Equations of state
- Constant volume constraint.

The SG shell side inventory is normally in a saturated state. There are, however, transients that may lead to non-equilibrium conditions. The two regions of the non-equilibrium SG model may be in the following thermodynamic states:

- The lower region (F region) is either subcooled liquid or saturated liquid with bubbles forming and rising to the surface.
- The upper region (G region) is either superheated steam or saturated steam with liquid droplets forming and flowing to the liquid region.

The two regions of the steam generator may have four different combinations of thermodynamic states and there is a different set of governing equations for each combination:

- Upper region (G) superheated steam, lower region (F) subcooled liquid.
- Region G superheated steam, region F saturated liquid with bubbles forming.
- Region G saturated steam with droplets forming, region F subcooled liquid.
- Region G saturated steam with droplets forming, region F saturated liquid with bubbles forming.

The steam generator model accounts for heat and mass transfer between the two regions. Mass transfer is modeled in the bubble rise and condensate drop models. The governing differential equations for each thermodynamic state are derived by first applying the mass and energy equations as well as the equations of state to each region of the steam generator. The governing equations for the four possible combinations of states in the steam generator and the way in which they are solved are given in Appendix A.

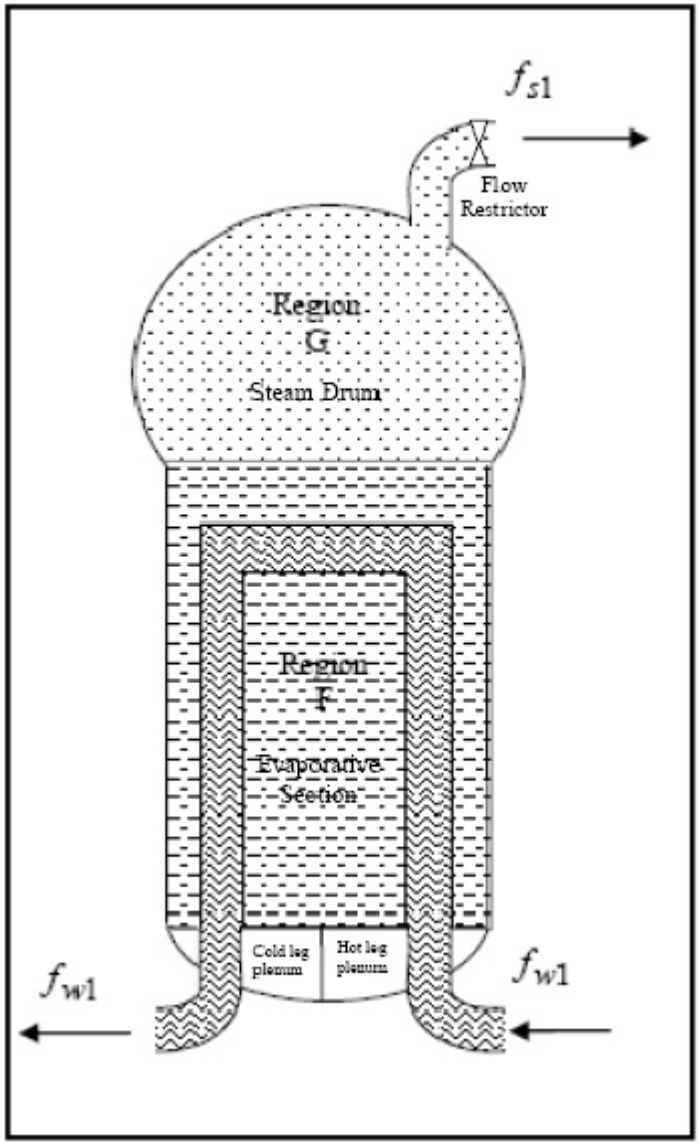

Figure 2.2.8 Schematics of a Steam Generator

2.2.3.2 Main Steam System

The main steam system in this model consists of the system of pipes and valves between the steam generator and the turbine. The system piping includes the main steam header and main steam line. Valves include the main steam isolation valve (MSIV), nine safety valves, the turbine stop valve, and turbine governor valve. A flow restrictor located at the junction between the steam line and steam generator is also included in the model. The operations of the steam dump and steam bypass systems are not included in the model.

The thermodynamic state of the main steam system is governed by conservation of mass and energy. Equations of motion are applied to determine the flow rate between the steam generator and the steam header [4]. The flow model is limited to choked flow conditions. The flow at the flow restrictor, MSIV, the safety valves, and the turbine valve is also governed by the equation of motion and limited to choked flow conditions.

During transients which exceed the capacity of the pressure control system, the steam generator pressure is controlled by a set of nine spring-loaded safety valves. The valve operations are expressed by a set of bistable actions. The model accounts for different lift settings between safety valves to simulate lift and reset sequence.

2.2.3.3 Main Feedwater and Auxiliary Feedwater Systems

The main feedwater system (MFWS) is designed to deliver water to the steam generators during power operations and after reactor trip. For the purpose of this study, feedwater flow delivered to the feedwater regulating and bypass valves are modeled. The feedwater regulating and bypass valves are controlled by the DFWCS.

The auxiliary feedwater system (AFWS) is designed to deliver water to the steam generator upon actuation of the emergency feed signal on low steam generator level. The AFWS flow is controlled by a bistable controller. Its actuation on low steam generator level is independent of the MFWS. The SG level instruments associated with the AFWS operation are redundant and safety related.

2.2.4 Fault Tolerant Features

The benchmark system has a number of fault tolerant features:

- Since the MFV, BFV, FP controllers forward the control signals to the corresponding control points (the MFV, BFV, and FP, respectively, as well as the PDI controller), they provide a level of fault tolerance if both the MC and BC fail by allowing the operators time to intervene by holding the outputs of each to a previously valid value.
- The MC and BC, the MFV, BFV and FP and the PDI controllers are each connected to an independent power source wired to a separate bus. A single power source failure can only affect one computer, all of the MFV/BFV/FP controllers, or the PDI controller at one time.
- Both the MC and BC are set to oversample at 3 times the Nyquist criterion[4] to avoid aliasing.
- The MC and BC are able to process the sensor inputs and perform the control algorithms within one third of the needed response frequency of the physical process. A

[4]The Nyquist criterion states that the highest frequency present in a signal must be less than half of the sample frequency [93].

failure in the MC or BC can be detected and the fail over[5] to a healthy component can occur with enough time to meet the response requirements of the process.

- The water level setpoint is taken from a switch connected to the MFV and is propagated to both the MC and BC. If the setpoint signal goes out of range, then the computers fall back on a preprogrammed setpoint value.
- Each computer (MC or BC) is connected to a watchdog timer. A watchdog timer is a hardware timer and associated connections used to determine if a software error or other computer failure has rendered a processor unusable. A normally functioning computer resets the watchdog timer at regular, defined intervals so the timer does not "go off." However, in the presence of a software error or another computer failure, the timer will not be reset by the computer and the timer can go off. For example, a runaway process, halted (failed) processor, or a sufficiently lengthy computational delay may result in failure to reset the watchdog timer. As a result, the watchdog timer may go off. If the timer goes off, all components in the controller connected to the watchdog timer are notified of the computer failure. In the case of the benchmark system, the MFV, BFV, and FP controllers are notified and transfer control away from the affected computer.
- Each computer (MC or BC) verifies and validates its inputs, checking for out range and excessive rate changes in the inputs that would indicate errors in the sensor readings or problems with the analog to digital conversion of the values. Each computer will ignore input that fails these checks if the other inputs are still valid.
- The values of the inputs are averaged across redundant sensors.
- Deviation between the two sensors is detected and, if the deviation is large enough, the computer can signal a deviation error to the MFV, BFV, and FP controllers so they may switch to the other computer.
- The PDI controller provides one more level of fault tolerance, in that it holds the MFV to a needed position if the MFV does not produce output.
- The MFV, BFV and FP controllers also send their outputs to the MC and BC. When the MC (or BC) is in control, it compares its output to the signals that the MFV, BFV and FP controllers output signal to the actuators. If the output signal differs, then the computer indicates to the MFV, BFV and FP controllers that it has failed.

The digital feedwater controller failover logic consists of the following: the MC has control of the control points initially, with the BC in hot standby. If the MC fails, then the BC takes control. If the BC fails after the MC has failed, then the MFV, BFV, and FP controllers each use one of their recent output value from the computer (essentially the last one that the controller can store) and recycle that value to the control points. Any time a component fails, the operator console is notified to allow operators to take mitigating actions.

[5]Fail over is the process in which a degraded component is removed from control and replaced by a healthy component

2.3 Description of System Operation under Abnormal Conditions

This section presents a failure modes and effects analysis (FMEA) of the digitial feedwater control system. These failure classes include sensor failures, output failures, input failures and internal failures. Each of the failure classes may contain a large number of faults. For example, sensor failure may be the result of a physical sensor failure, cut wires, loose connections, or hardware (such as analog to digital converters) on the receiver failing. While these failure classes may be general, they are expected to capture the necessary information about possible failures of the benchmark feedwater control system. Each component type in the system, MC, BC, MFV controller, BFV controller, FP controller and PDI controller has a separate FMEA chart associated with that component. Full FMEA is given in Appendix B. In addition, the actuated devices (i.e. MFV, BFV and FP) may fail to perform their design due to mechanical failure. The only mechanical failures that will be considered for the benchmark DFWCS are the valves getting stuck in their current position.

2.3.1 Main and Backup Computer FMEA

Table 2.3.1 summarizes MC failure classes. Sensor failures may occur from many possible sources, including decaying or broken wires, failed sensors, intermittent transmission failures, or analog to digital conversion errors. A sensor failure is detected through the use of rate of change checks, range checks, and comparison to previous values used by both the MC and BC. For illustration, if the sensor was reading a 1.5 feet water level signal and then it received a 150 feet signal, then this value is considered to be an invalid sensor reading. Also, an indicator light illuminates on the operators' console. The MC and BC each disregard an invalid sensor reading if there is one sensor of each type that is valid and wait for one computation interval before indicating their failure. However, a common mode sensor failure that causes one sensor of a type from both the MC and BC will not cause the MC and BC to fail. Rather, they will operate using only one sensor. Thus, the controlled process is not affected by a single sensor failure. However, due to the physical wiring of the sensors, if one sensor fails, its failure may affect sensor readings for several computers on both steam generators and may lead to a common mode failure. Even with this type of failure, the digital feedwater system can still maintain control. As in the case of single sensor failure described above, this event may occur from many possible sources including decayed or broken wires, failed sensors, and intermittent transmission failures. Multiple sensor failures are detected in the same manner as the single sensor failure described earlier. The MA and BC uses previous values to perform their computations if multiple sensors fail as in the single sensor failure case. Also, the MC and BC notifies the MFV, BFV, and FP controllers that it considers itself failed if the sensor readings do not return to normal after a brief delay (which depends on the sampling requirements of the physical process). This notification forces the MFV, BFV, and FP controllers to switch their input acquisition device as necessary (i.e. they switch from the MC to BC, BC to MC). In case both the MC and BF are failed, all the controllers maintain the latest valid value. Power level changes are disabled in this mode. An indicator light illuminates on the operators' console. Control could be affected if the MC and BC have invalid data and the MFV, BFV, and FP controllers are forced to control the process.

The MC failure status signal can be activated if the MC takes itself down through sensor validity checks, through application failures or through a communication problem with the MFV, BFV, and FP controllers. The MC failure status signal can also be activated if the MC detects that the output it sent to the MFV, BFV or FP controllers differs from the output actually used by those controllers. Alternatively, the MC's watchdog timer may go off. Finally, the MC may fail and its failure may not be detected due to a communication failure with the MFV, BFV, and FP controllers. Failure is detected through the use of a watchdog timer and the computer's internal validity checks. If the watchdog timer goes off, it signals the MFV, BFV and FP controllers to notify them that the MC has failed. These are the only components that are affected by a MC failure. The MC also checks and indicates that it is unreliable if any detectable errors occur. For instance, the MC indicates that it is unreliable if it can detect that its sensors readings are invalid. A detected failure causes the MC to signal failure and the system then transfers to using the BC. If a failure is not detected, then the system continues to use the MC until either the output goes out of range or the rate of change exceeds what is allowed. Then the BC takes over. The impact of a detected failure is minimal as the BC takes control due to the watchdog timer reset. The system can maintain control. However, an undetected failure may act as a Byzantine failure[6]. For example, if the MC experiences a crash, possibly an arbitrary value may be the output to the BFV, MFV and FP controllers. This is one of the possible additional failure modes of digital I&C systems. Also, the MC's output may drift or simply fail to send an output signal. At some point, the output signal may change such that it goes out of range, the rate of change becomes too high or the output signal is lost. Should this situation occur, the MFV, BFV, and FP controller(s) will detect the error and switch to the BC. However, there still may be a loss of control until the failure is detected. It is assumed that the arbitrary value category includes any Byzantine failures that may occur.

Table 2.3.1 Abbreviated FMEA for the MC

Failure Type	***Detection of Failure***	***Effects of Failure on Controller***	***Effects on Controlled/Monitored Process Variables***
Loss of one sensor inputs of a type of input via computer diagnostics.	Computer detects loss in sensor reading.	Computer ignores failed sensor. If the sensor does not return (to valid input), computer indicates it has failed if the other computer is operating normally.	None. Backup Computer takes control if the sensor does not return.

[6]A Byzantine failure is one in which any failed processes are modeled as actively trying to disrupt the normal goals of the system. This definition comes from the Byzantine agreement problem as defined by Lamport [94]

Loss of both sensor inputs of a type of input via computer diagnostics.	Computer detects loss in sensor reading.	Computer uses the old values of the sensor reading. If the sensor reading do not return, computer indicates that it has failed.	None. At the worst case the Backup Computer takes control.
Intermittent sensor failure.	Computer detects out of range output and physically impossible rates of change.	Computer ignores sensor input and uses old values. It fails itself if sensor does not return.	None. At the worst case the backup computer takes control.
Sensor failure.	Main and backup computers detect this via range output and physically impossible rates of change.	Computer ignores sensor input and uses old values. It fails itself if sensor does not return.	Both the main and backup computers will fail themselves if the sensor does not return.
Both sensors fail.	Main and backup computers detect this via range output and physically impossible rates of change.	Computer ignores sensor input and uses old values. It fails itself if sensor does not return.	Both the main and backup computers will fail themselves if the sensors do not return.
Loss of an output (0.0 vdc).	Component connected to computer detects 0.0 vdc input reading.	Component signals that this computer has failed.	None. The backup computer takes control.
Loss of Power.	Computer failed signal is tripped.	Continue with fail over logic.	None. The backup computer takes control and no effect on water level.
Roundoff/truncation/ sampling rate errors.	Detected by connecting components if output ever is out of range or exceeds the physically possible rate.	Component fails the computer.	If not detected, water level may drift. If detected, the backup computer takes control and no effect on water level.
Unable to meet needed response requirements.	Watchdog timer detects the failed computer.	Failover action occurs.	None. Backup computer takes control with no effect on water level.

Watchdog timer fails to activate.	Detectable when outputs of computer go out of range or exceed the physically possible rate.	Failover action occurs if detected.	If not detected, water level may drift. If detected, the backup computer takes control with no effect on water level.
Watchdog timer activates when computer has not failed.	Not detectable.	Failover action occurs.	None. Backup computer takes control.
Arbitrary value output.	Detectable by connected component if the computer does not reset the watchdog timer.	Component connected to output signals that this computer has failed.	If not detected, water level may increase or decrease. If detected, the backup computer takes control and no effect on water level.
MFV/BFV/FP controllers do not use the output that the MC computed.	Detected by comparing outputs of MFV/BFV/FP controllers with MC output	Component initiates failover operation.	The valves and feedwater pump will remain in the same state if the fail over fails to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration.
Setpoint drift.	Detectable by the computers once the setpoint has drifted out of range.	Computers revert to using a preprogrammed value.	Water level will increase or decrease until it reaches the setpoint that has drifted out of range, then water level will settle to the preprogrammed value.

Table 2.3.2 shows BC failure modes. A simultaneous MC and BC failure is detected through the use of watchdog timers. If the watchdog timers go off, the MFV, BFV and FP controllers are notified of the computer failures. The MFV, BFV, and FP controllers are the only components that are affected directly by this failure. An indicator light illuminates on the operators' console. The failure is mitigated by the action of the MFV, BFV, and FP controllers to hold the outputs at their old values. The impact of these failures is, as expected, quite significant. In this case, the digital control system has failed and the operators must intervene to take manual control of the process as the MFV, BFV, and FP controllers continue to output old values to the MFRV, BFRV, and FP controllers. The problems resulting from a MC and BC failure get worse if the failures are not detected, as the MFV, BFV, and FP controllers will take more time before they take over.

Table 2.3.2 Abbreviated FMEA for the BC

Failure Type	***Detection of Failure***	***Effects of Failure on controller***	***Effects on process variables***

Loss of one sensor inputs of a type of input via computer diagnostics.	Computer detects loss in sensor reading.	Computer ignores failed sensor; if the sensor does not return, computer indicates it has failed.	None.
Loss of both sensor inputs of a type of input.	Computer detects loss in sensor reading.	Computer uses the old values of the sensor reading; if the sensor reading does not return, computer indicates that it has failed.	The valves and feedwater pump will remain in the same state if the failover to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration.
Intermittent sensor failure.	Computer detects out of range output and physically impossible rates of change.	Computer ignores sensor input and uses old values; fails itself if sensor does not return to valid input.	The valves and feedwater pump will remain in the same state if fail over to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration.
Sensor failure.	Main and backup computers detect this via range output and physically impossible rates of change.	Computer ignores sensor input and uses old values. It fails itself if sensor does not return.	Both the main and backup computers will fail themselves if the sensor does not return.
Both sensors fail.	Main and backup computers detect this via range output and physically impossible rates of change.	Computer ignores sensor input and uses old values. It fails itself if sensor does not return.	Both the main and backup computers will fail themselves if the sensors do not return.
Loss of Power.	Computer failed signal is tripped.	Continue with fail over logic (no effect if MC is in control of the process).	The valves and feedwater pump will remain in the same state if fail over fails to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration.

Roundoff/Truncation/ Sampling rate errors.	Detected by connecting components if output ever is out of range or exceeds the physically possible rate.	Component fails the computer.	If not detected, unknown. If detected, the valves and feedwater pump will remain in the same state if fail over fails to the MFV/BFV/FP controllers, thus the effect on the process variables depends on the event in consideration.
Unable to meet needed response requirements.	Watchdog timer detects this and fails the computer.	Fail over action occurs.	The valves and feedwater pump will remain in the same state if fail over fails to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration.
Watchdog timer fails to activate.	Detectable when outputs of computer go out of range or exceed the physically possible rate.	Fail over action occurs if detected.	If not detected, unknown. If detected, the valves and feedwater pump will remain in the same state if fail over fails to the MFV/BFV/FP controllers, thus the effect on the process variables depends on the event in consideration.
Watchdog timer activates when computer has not failed.	Not detectable.	Failover action occurs.	The valves and feedwater pump will remain in the same state if fail over fails to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration.
Arbitrary value output.	Detectable by connected component if the computer does not reset the watchdog timer.	Component connected to output signals that this computer has failed.	If not detected, unknown. If detected, the valves and feedwater pump will remain in the same state if fail over fails to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration.
MFV/BFV/FP controllers do not use the output that the BC computed.	Detected by comparing outputs of MFV/BFV/FP controllers with BC output	Component initiates failover operation.	The valves and feedwater pump will remain in the same state if the fail over fails to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration.
Setpoint drift.	Detectable by the computers once the setpoint has drifted out of range.	Computers revert to using a preprogrammed value.	Water level will increase or decrease until the setpoint has drifted out of range, then water level will settle to the preprogrammed value.

2.3.2 The FMEAs for MFV, FP, BFV and PDI decision Controllers

The abbreviated FMEA for MFV, BFV and FP controllers are given in Tables 2.3.3, 2.3.4, and 2.3.5, respectively. The MFV may fail due to a power loss or an internal program crash (possibly from hanging, hard crash, or output failure). This event can only be detected if the MFV output drops to 0.0 volts. In this case, the PDI controller senses the drop in output and asserts the old value of the MFV controller's output to the MFV. The PDI controller acts as the MFV would, outputting the old output to the MFV. At best, the digital feedwater control system reverts to a manual mode (and fails). At worst, the MFV controller does not output the correct signals to the MFV and can cause loss of control of the process. The BFV controllers may fail because of a power loss or a program crash. This event cannot be detected by the digital feedwater control system. The BFV controller does not output the correct signals to the BFV and may cause loss of control of the process. The MFV/BFV/FP controllers may disagree as to the failure states of each computer. It is then possible for the MFV controller to be accepting different signals than the FP and BFV controllers. This event cannot be detected. The impact of this failure is that the SG water level may increase or decrease depending on the failure of the MC/BC and the MFV/BFV/FP controller's indication of those failures. Communication between the MC and BC and the MFV controller is accomplished simply by the analog outputs between them. Also, communication between the MFV, BFV, FP and PDI controller occurs only to coordinate failure detection. This communication type of failure is captured by the computer erroneously reported failed or not failed. Finally, it is noted that the MFV, BFV, PDI and FP controllers are all digital also, and as such may experience crashes that may cause them to be unable to detect failures or output arbitrary values.

Table 2.3.3 Abbreviated FMEA for MFV Controller

Failure Type	***Detection of Failure***	***Failure Effects***	***Failure Effects on Process Variables***
Loss of power.	Detected by PDI.	PDI uses old signal for the MFV.	The MFV will remain in the same state. Thus the effect on the process variables depends on the event in consideration.
Loss of output signal.	Detected by PDI.	PDI uses old signal for the MFV.	The MFV will remain in the same state. Thus the effect on the process variables depends on the event in consideration.
Computer erroneously reported failed.	Not detected.	Component initiates failover operation.	The MFV will remain in the same state if the fail over fails to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration.

Computer erroneously reported not failed.	Can be detected if computer fails and output of computer is out of range, the rate of change is too great, or the watchdog timer goes off.	Component initiates failover if detected.	If detected the MFV will remain in the same state if the fail over fails to the MFV/BFV/FP controllers, thus the effect on the process variables depends on the event in consideration. Otherwise, unknown.
MFV, BFV, FP controllers do not agree from which computer to accept input.	Cannot be detected.	MFV, BFV, and FP may receive inconsistent input.	Unknown effect, may increase or decrease water level.
High output.	Cannot be detected.	MFV will be driven open.	Feedwater flow will increase, causing the water level to increase.
Low output.	Cannot be detected.	MFV will be driven shut.	Feedwater flow will decrease unless the computer is operating in low power mode. In low power mode, there will be no effect.
Arbitrary value output.	Cannot be detected.	MFV will open/close according to the output.	Unknown effect; may increase or decrease water level.

Table 2.3.4 Abbreviated FMEA for BFV Controller

Failure Type	***Detection of Failure***	***Failure Effects***	***Failure Effects on Process Variables***
Loss of power.	Not detected.	Nothing.	Unknown effect on water level.
Loss of output signal.	Not detected.	Nothing.	Unknown effect on water level.
Computer erroneously reported failed.	Not detected.	Component initiates failover operation.	The valves and feedwater pump will remain in the same state if the fail over fails to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration.

Computer erroneously reported not failed.	Can be detected if computer fails and output of computer is out of range, the rate of change is too great, or the watchdog timer goes off.	Component initiates failover if detected.	If detected the valves and feedwater pump will remain in the same state if the fail over fails to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration. Otherwise, unknown.
MFV, BFV, FP controllers do not agree from which computer to accept input.	Cannot be detected.	MFV, BFV, and FP may receive inconsistent input.	Unknown effect, may increase or decrease water level.
High output.	Cannot be detected.	BFV will be driven open.	Feedwater flow will increase, causing the water level to increase.
Low output.	Cannot be detected.	BFV will be driven shut.	Feedwater flow will decrease unless the computer is operating in low power mode. In high power mode, there will be no effect.
Arbitrary value output.	Cannot be detected.	BFV will open/close according to the output.	Unknown effect; may increase or decrease water level

The FP controller may fail because of a power loss or a program crash as the MFV or BFV controller could. This event cannot be detected by the DFWCS. The FP controller would not output the correct signal to the feedwater pump and thus would be unable to control the feedwater pump.

Table 2.3.5 Abbreviated FMEA for FP Controller

Failure Type	**Detection of Failure**	**Failure Effects**	**Failure Effects on Process Variables**
Loss of power.	Not detected.	Nothing.	Unknown effect on water level.
Loss of output signal.	Not detected.	Nothing.	Feedwater flow will decrease to the minimal value, possibly decreasing the water level.

Computer erroneously reported failed.	Not detected.	Component intitiates failover operation.	The valves and feedwater pump will remain in the same state if the fail over fails to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration.
Computer erroneously reported not failed.	Can be detected if computer fails and output of computer is out of range, the rate of change is too great, or the watchdog timer goes off.	Component initiates failover if detected.	If detected the valves and feedwater pump will remain in the same state if the fail over fails to the MFV/BFV/FP controllers, thus the effect on the process variables depends on the event in consideration. Otherwise, unknown.
High output.	Cannot be detected.	FP will be driven to increased pressure.	Feedwater flow will increase, causing the water level to increase.
Low output.	Cannot be detected.	FP will be driven to decreased pressure.	Feedwater flow will decrease, causing the water level to decrease.
Arbitrary value output.	Cannot be detected.	FP will speed up/slow down according to the output.	Unknown effect. May increase or decrease water level.
MFV, BFV, FP controllers do not agree from which computer to accept input.	Cannot be detected.	MFV, BFV, and FP may receive inconsistent input.	Unknown effect. May increase or decrease water level.

The PDI controller may also fail because of a power loss or a program crash as the MFV, BFV and FP controllers and event will not be detected by the digital feedwater control system. If the PDI controller fails by loss of output, then a failed MFV controller will not be detected. If the PDI fails by sending a spurious signal to the MFV when the MFV controller has not failed, then the MFV and PDI controller outputs will sum and give an increased signal to the MFV, causing the MFV to open more than it should.

Table 2.3.6 Abbreviated FMEA for PDI Controller

Failure type	***Detection of Failure***	***Failure Effects***	***Failure Effects on Process Variables***
Loss of inputs (0.0 vdc).	PDI detects a 0.0 vdc input.	PDI outputs an old value of the MFV controller output to the MFV.	The MFV valve will receive a signal that is the sum of the PDI and MFV controller signals, thus increasing feedwater flow.

Loss of power.	Not detected.	None.	None directly, unless the MFV controller has failed, then resulting effect is unknown.
Loss of Outputs.	Not detected.	None.	None directly, unless the MFV controller has failed, then resulting effect is unknown.
Arbitrary failure.	Not detected.	PDI can output any value to MFV.	The MFV valve will receive a signal that is the sum of the PDI and MFV controller signals, thus increasing feedwater flow.

2.3.3 Communication in I&C Systems and Related Problems

Communication failures are being implicitly rather than explicitly considered in the FMEA charts presented in Sections 2.3.1 and 2.3.2 due to the difficulty of obtaining communication specific failure data. This section presents an overview of the state-of-the-art of communications in I&C systems and its potential impact on the design of future digital I&C systems.

Network-based control of factories and automation systems has been in place for decades. The enabling technologies developed for such systems ensure guaranteed transmission of messages within bounded time [95, 96]. A simple way of guaranteeing delay in communication is to connect individual devices to each other directly, leading to fully connected device topologies. While fully connected topologies provide minimum delays and high robustness against individual link failures, the setup and maintenance cost of such systems are high and they scale very poorly even to modest number of nodes. The local area network standards used for control systems aim to provide flexibility in connecting different devices to each other without relying fully-connected topologies. These protocols achieve bounded delay goal by regulating the channel access. One of the most widely known examples of controlled channel access is the use of tokens, which has led to IEEE 802.4 (token bus - currently inactive) and IEEE 802.5 (token ring) [97] standards. Token bus is an extension of IEEE 802.3 (Ethernet) [98] standard to limit the channel access time. A token passed between nodes in a predetermined order determines the right to channel access. In the token ring standard, the same idea is applied to a number of nodes connected in a ring topology.

Hardware-related problems affect the reliability of control networks. Ideally, operation of the network needs to be minimally affected when a network element (a node, an interface, or a link) becomes unavailable. The physical disconnection of network elements due to physical link failures is a relatively common problem, especially in environments where the connections are in close proximity of heavy machinery. In general, protocols are designed to recover from such failures as long as the remaining resources allow this. The resilience to hardware failures is directly connected with the redundancy of resources in the network. As an example, a severed communication link may have a minimal effect of a fully connected network. The same may disrupt all communication on a bus system. Obviously, the partial or full operation disruption

needs to be incorporated into reliability models. Another important aspect to consider in such models is the response time of the communication protocol to such changes. Depending on the protocol details and the severity of the hardware problem, the transient behavior during recovery may result in sub-par performance of the system, potentially leading undesired behavior of the entire system. Therefore, reliability models should include recovery models and their properties, as well.

Another important set of events is the errors in communication. Signals sent over communication links can be corrupted due to EM interference and other noise sources. While shielding techniques are developed to combat interference and noise and there is regulatory guidance to reduce the likelihood of EM interference [99], completely error-free communication is virtually impossible. The use of fiber-optical cables solve problems related to interference and noise, but suffer from high setup and maintenance costs as well as from fragility and inflexibility in laying the cables. Software/protocol-based error detection and correction mechanisms are required to minimize the effect of such errors. Depending on the protocol details, the behavior of the network in terms of availability and delay may change in the face of communication errors. As these bit-level errors are the most frequently occurring problems, they need to be modeled together with the behavior of the protocols, as well.

Due to its wide market penetration, IEEE 802.3 products are also considered for such systems with greater caution. Although IEEE 802.3 standard does not provide delivery or delay guarantees, they can be used safely only at very low loads, where the system behaves almost like a point-to-point link. The unreliable nature of loosely-controlled and shared communication media is more pronounced in the wireless local area networks (WLAN). IEEE 802.11 [100] standard is the most widely accepted standard for WLAN implementations. Recently, the use of WLANs has also been considered for communication between components of control systems [101].

Distributed control technology has also been used to control processes in many power generation facilities. The information exchange in such systems has been performed over wired networks that require long cables to be run between devices. Especially in coal-based power generation plants, the long distances covered and the exposure of communication hardware, especially cables, cause significant risks of communication disruption. As such disruptions take sometimes days to fix, wireless communication between locations with line-of-sight is considered as a plausible alternative to cables.

In nuclear power plants, the traditional method of communication is direct and wired connections between devices. Direct and wired connections have distinct advantages in terms of security and reliability. However, the rigidness of the wiring generally does not allow reconfigurations. Furthermore, system setup and expansion are costly operations when wiring is involved. In terms of flexibility of deployment and reconfiguration, wireless networks have many advantages. While recognizing these advantages, one should also keep in mind the following aspects:

- *Availability*: Wireless communication systems share wireless resources among multiple contenders. In addition to traffic-based queuing delays, environmental factors such as shadowing, fading, and interference from nearby devices, etc., may hinder the availability of the communication service.
- *Reliability*: Sharing of wireless resources can occur in various domains including time, frequency, code, space, and any combination thereof. The coordination generally increases the system complexity, and potentially reduces the utilization of scarce wireless resources. COTS hardware and associated standards, such as IEEE 802.11 [100], rely on loosely-controlled sharing of resources. Consequently, the information delivery between two points in the network is not always guaranteed.
- *Delay*: Due to the reasons listed for reliability considerations, the delay in such networks is not upper-bounded, either. However, medium access control mechanisms such as the ones used in cellular networks overcome delay uncertainty through hard allocation of resources at the expense of low utilization.

While it is technically possible to guarantee all four measures listed above in a wireless network for control, it is very unlikely to achieve these goals with COTS products. We also acknowledge the importance of security related reliability issues in network-based control systems. However, these issues are beyond the scope of this work and the above discussion focuses solely on reliability problems in a secure networking environment.

2.3.4 Discrete-State Representation of the Benchmark System

A discrete-state representation of the benchmark DFWCS is more convenient for both the Markov and DFM methodologies under consideration in this report. For such a representation, the DFWCS topology can be regarded as consisting of three layers of interactions based on Sections 2.3.1-2.3.6:

- intra-computer interactions
- inter-computer interactions
- computer-controller-actuated device interactions

The intra-computer interactions layer consists of 5 states (see Fig.2.3.1). These interactions can be regarded as transitions between the possible states of a single computer. In State A, the computer is operating correctly and nominally. In State B, the computer detects loss/invalid output for 1 sensor of any type (e.g. water level). State C represents loss/invalid output for 2 sensors of any one type. In state D the computer has detected an internal problem and is signaling that it has to be ignored. In State E, either the sensor output is invalid or there is an internal processing error in the computer; however, the computer does not detect the fault and is transmitting the wrong information to the controllers. These states capture the possible failures in the FMEA's in Section 2.3.1 and 2.3.2 that occur within both the MC and BC.

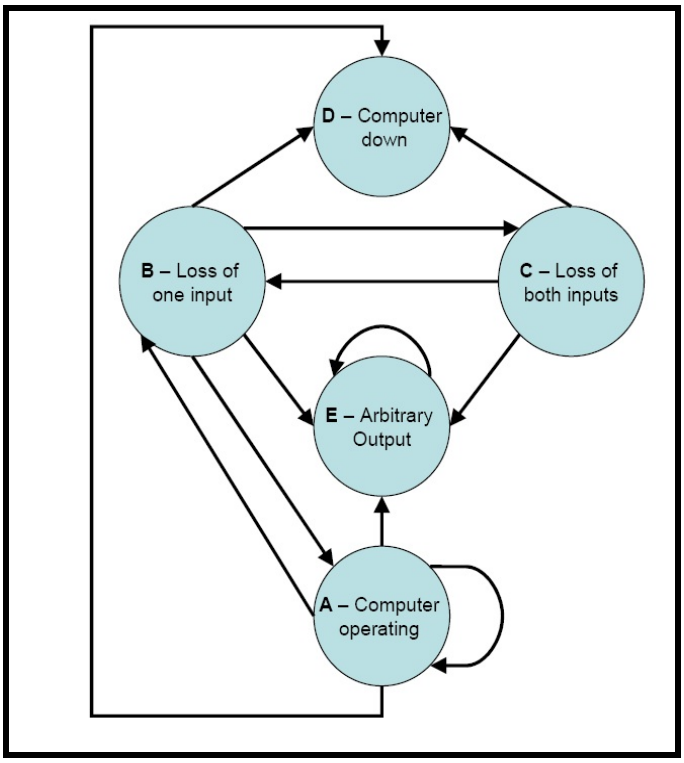

Figure 2.3.1 Intra-computer interactions of the DFWCS

The inter-computer interaction layer in Fig. 2.3.2 can be thought of as including the possible transfers of control of the actuated devices among the MC, BC and controller as identified in the FMEA's in Section 2.3.1 and 2.3.2. For example, the transfer of control from the MC to the BC would be represented here. There are 3 such computer-computer macro-states (MSs) shown in Fig.2.3.2. In State 1 both MC and BC are operating normally. In State 2, one computer is operating correctly and the other is down but can be recovered. In State 3, again one computer is operating correctly and the other is down but it is not recoverable. Transitions between the MSs depend upon the state of the controlling computer. Primary and secondary computers correspond, respectively, to the computer that is sending data to the controller and to the computer that is waiting in hot standby. Either the MC and BC can be the primary or the secondary computer. Recoverable and non-recoverable failures are defined as the following:

- Recoverable failure corresponds to the inability for the computer (which is still operating correctly) to send valid data to the controller (e.g. due to a loss of input from one couple of sensors)
- Non-recoverable failure corresponds to an internal failure of the computer (e.g. the trip of the watchdog timer) or to a loss of output of the computer itself

If the secondary computer (i.e. the computer that is not in control) fails and it is still recoverable, a transition from MS 1 to MS 2 occurs. These transitions simply take each state in MS 1 to the corresponding state in MS 2. For example, State A (or operational) in MS 1 would have a transition to State A in MS 2. When the secondary computer recovers, the opposite transitions occur (from MS 2 to MS 1). Again, for example, if the operating computer is in State A, MS 1, then the transition would start there and end in State A in MS 1.
From the State D in MS 1 the possible transitions represent the takeover of control of the process by the secondary computer (which from now on will be regarded as the primary computer). The transitions from this state go to all states except for State D in MS 2. The rationale behind these transitions is that the secondary computer was operating and may have transitioned to states other than State A in MS 2. The reason that State D in MS 2 is not a possible destination is that if State D is reached by the secondary computer, then another transition from MS 1 to MS 2 must have already taken that into account.

The fail-over action from MS 1 to MS 3 is a result of controller action via the watchdog timer or detecting the output failure from the computer. This action takes down the failed computer permanently and can occur to both the primary and secondary computer. If it occurs to the secondary, the transitions mimic the action of the secondary failure transitions from MS 1 to MS 3 by simply transitioning from a state in MS 1 to the respective state in MS 3. For example, State A in MS 1 would have a transition to State A in MS 3. If the primary computer fails in a non-recoverable manner when both MC and BC are operating (i.e. when the DFWCS is in MS1), then the DFWCS can go to any state in MS 3 except State D by the same rationale for transitions between MS1 and MS2 . The transitions must take into account that the secondary computer may have already entered different states and these must be represented in the transitions to MS 3.

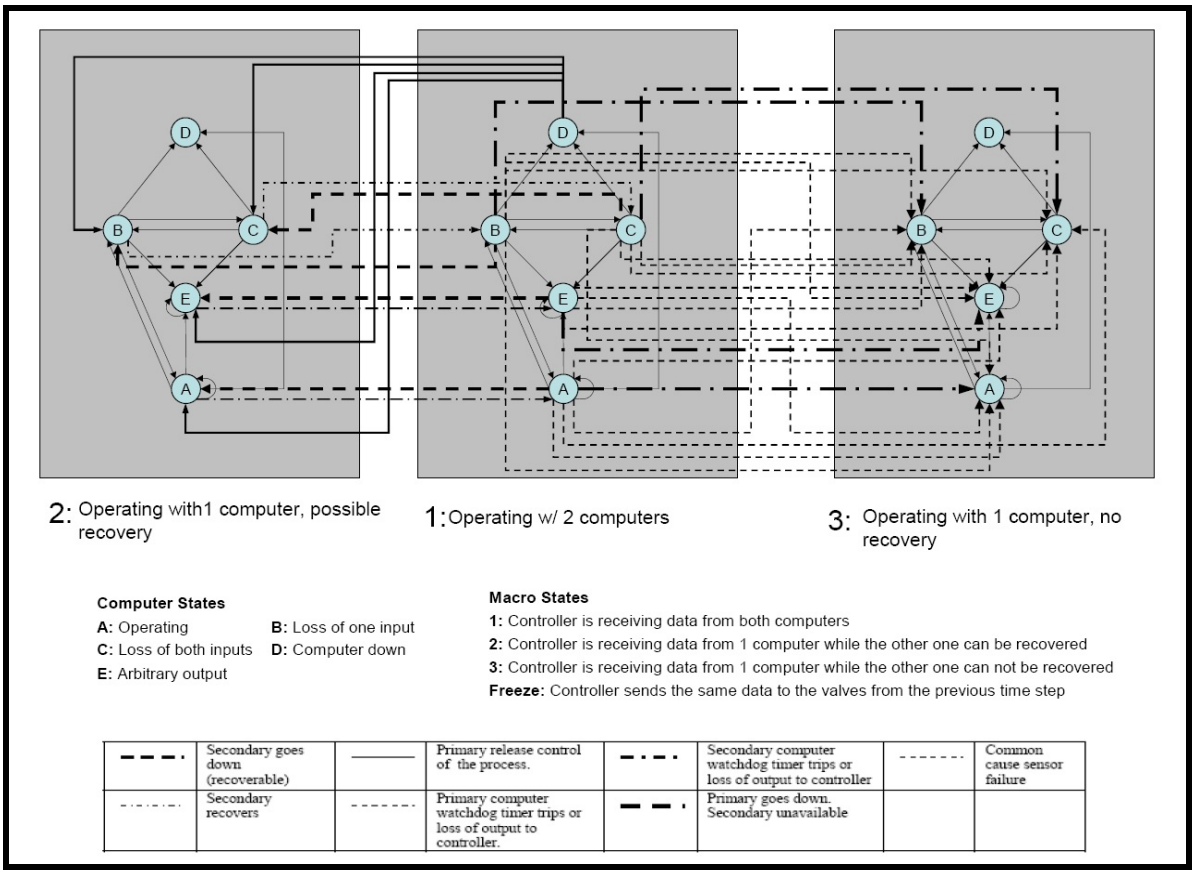

Figure 2.3.2 Inter-computer interactions of the DFWCS and definition of the macro-states (MSs)

Figure 2.3.3 shows all the possible controller-computer-actuated device interactions according to the FMEA charts presented in Section 2.3.1 and 2.3.2. The shaded circles represent signals to the actuated devices (e.g. MFV, BFV, FP) upon computer/controller failure, as well as the mechanical failure of the actuated device (Device Stuck). As indicated in the beginning of Section 2.3, mechanical failure of the actuated device leads to the device maintaining its current position for MFV and BFV or to zero flow for FP. The planes represent the communication status between the controller and actuated devices. The two-way transitions between Planes I and II are necessary to keep track of the computer from which the controller is receiving data when the communications between controller and actuated device are restored.

As presented in Fig.2.3.3, the following types of controller failures are under consideration:

- Arbitrary output: random data are generated and sent to the actuated device (i.e. pump or valves)
- Output high: output value is stuck at the maximum value (i.e. valve totally open or pump at the maximum speed)

- Output low: output value is stuck at the minimum value (i.e. valve totally closed or pump stopped)
- 0 vdc output: loss of communications between controller and actuated device

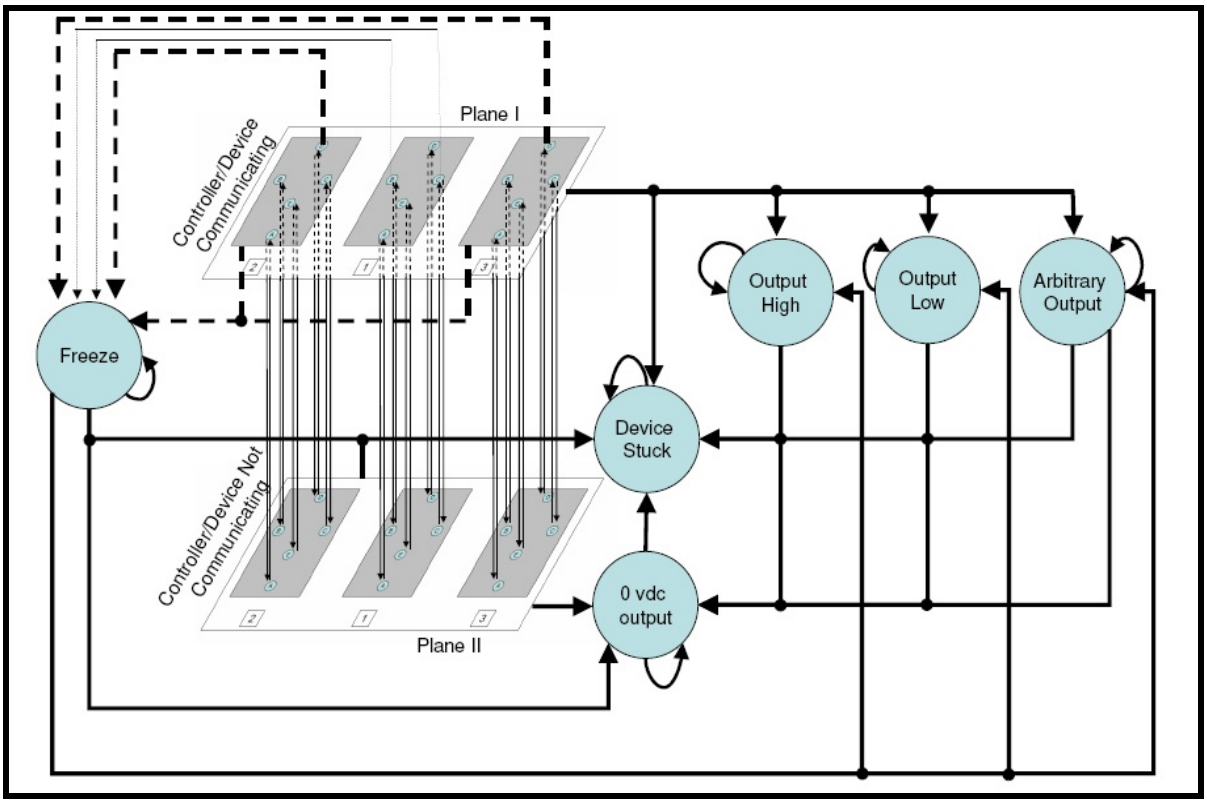

Figure 2.3.3 Computer-Controller-Actuated Device Interactions

Moreover, as a result of the failure of both computers, the controller can recognize the failure and send to the actuated devices (i.e. pump or valves) the old valid value (i.e. Freeze). If the controller does not recognize the failure, then it will simply pass on false information (Arbitrary Output) to the actuated device. Figure 2.3.3 also shows how the computer-computer interactions (presented in Fig. 2.3.2) integrate with computer-controller and controller-actuated device interactions. Device Stuck refers to mechanical failures and is independent of the failure modes of the computers and controllers. The behavior of the controller under normal and failed operation can be described as follows:

- When both MC and BC are down, the controller transits to the Freeze state. The actuated device remains in the position corresponding to the last valid value.
- If the controller is operating and an Output High or an Output Low or an Arbitrary Output failure occurs, the controller transits to the corresponding state and the actuated device assumes the highest, the lowest or an arbitrary position, respectively.

- If the controller is in the Freeze state and an Output High, Output Low or Arbitrary Output failure occurs, the controller transits to the corresponding state and the actuated device assumes the highest, the lowest or an arbitrary position, respectively.
- If a loss of output occurs when the controller is failed (i.e. the controller is sending Arbitrary Output, Output High or Output Low state), then the actuated device receives a 0 vdc as input which correspond to the lowest position.

Explanation of state transitions and groupings of transition by event type are given in Tables 2.3.7 and 2.3.8, respectively, for transitions in both Planes I and II in Fig. 2.3.3. The notation used is Start State-End State with states that include the macro-state followed by the state letter (see Fig. 2.3.2). For example, the transition in which both computers are operational and continue to function normally is denoted 1A-1A.

Table 2.3.7: Explanation of State Transitions

Transition	***Event***
1A-1A	Primary Computer continues to operate normally
1E-1E	Primary Computer continues to operate abnormally
2A-2A	Primary Computer continues to operate normally
2E-2E	Primary Computer continues to operate abnormally
3A-3A	Primary Computer continues to operate normally
3E-3E	Primary Computer continues to operate abnormally
1A-3A	Secondary Computer watchdog timer trips or loss of output to controller
1A-3A	Primary Computer watchdog timer trips or loss of output to controller
1A-3B	Primary Computer watchdog timer trips or loss of output to controller
1A-3C	Primary Computer watchdog timer trips or loss of output to controller
1A-3E	Primary Computer watchdog timer trips or loss of output to controller
1A-1B	Primary Computer loses one input
1A-1D	Primary Computer detects an internal failure via self-diagnostics and takes itself down
1A-1E	Primary Computer does not detect internal failure and produces arbitrary output
1B-1A	Primary Computer's input returns
1B-1C	Primary Computer loses second input
1B-1D	Primary Computer's input does not recover in one processing cycle
1B-1E	Primary Computer does not detect internal failure and produces arbitrary output
1B-2B	Secondary Computer detects an internal failure via self-diagnostics and takes itself down
1B-3B	Primary Computer watchdog timer trips or loss of output to controller

1B-3A	Primary Computer watchdog timer trips or loss of output to controller
1B-3B	Primary Computer watchdog timer trips or loss of output to controller
1B-3C	Primary Computer watchdog timer trips or loss of output to controller
1B-3E	Primary Computer watchdog timer trips or loss of output to controller
1B-F	Common Cause sensor failure
1C-1B	One input for Primary Computer returns
1C-1E	Primary Computer does not detect internal failure and produces arbitrary output
1C-2C	Secondary Computer detects an internal failure via self-diagnostics and takes itself down
1C-3A	Primary Computer watchdog timer trips or loss of output to controller
1C-3B	Primary Computer watchdog timer trips or loss of output to controller
1C-3C	Primary Computer watchdog timer trips or loss of output to controller
1C-3E	Primary Computer watchdog timer trips or loss of output to controller
1C-1D	Primary Computer inputs do not recover in 1 processing cycle
1C-3C	Primary Computer watchdog timer trips or loss of output to controller
1C-F	Common Cause sensor failure
1D-2A	Primary Computer releases control of process, Secondary computer is now the primary computer
1D-2B	Primary Computer releases control of process, Secondary computer is now the primary computer
1D-2C	Primary Computer releases control of process, Secondary computer is now the primary computer
1D-2E	Primary Computer releases control of process, Secondary computer is now the primary computer
1E-3E	Secondary Computer detects an internal failure via self-diagnostics and takes itself down
1E-3A	Primary Computer watchdog timer trips or loss of output to controller
1E-3B	Primary Computer watchdog timer trips or loss of output to controller
1E-3C	Primary Computer watchdog timer trips or loss of output to controller
1E-3E	Primary Computer watchdog timer trips or loss of output to controller
2A-2B	Primary Computer loses one input
2A-2D	Primary Computer detects an internal failure via self-diagnostics and takes itself down
2A-2E	Primary Computer does not detect internal failure and produces arbitrary output
2A-1A	Secondary Computer becomes operational
2B-2A	Primary Computer input returns
2B-2C	Primary Computer loses one input
2B-2D	Primary Computer input does not recover in 1 processing cycle
2B-1B	Secondary Computer becomes operational

2B-2E	Primary Computer does not detect internal failure and produces arbitrary output
2C-2B	One input for Primary Computer returns
2C-2E	Primary Computer does not detect internal failure and produces arbitrary output
2C-2D	Primary Computer inputs do not recover in 1 processing cycle
2C-1C	Secondary Computer becomes operational
2A-F	Primary Computer watchdog timer trips or loss of output to controller
2B-F	Primary Computer watchdog timer trips or loss of output to controller
2C-F	Primary Computer watchdog timer trips or loss of output to controller
2E-F	Primary Computer watchdog timer trips or loss of output to controller
2D-F	Controller freezes output
2E-1E	Secondary Computer becomes operational
3A-3B	Primary Computer loses one input
3A-3D	Primary Computer detects an internal failure via self-diagnostics and takes itself down
3A-3E	Primary Computer does not detect internal failure and produces arbitrary output
3B-3A	Primary Computer input returns
3B-3C	Primary Computer loses one input
3B-3D	Primary Computer's input does not recover in one processing cycle
3B-3E	Primary Computer does not detect internal failure and produces arbitrary output
3C-3B	One input for Primary Computer returns
3C-3D	Primary Computer inputs do not recover in 1 processing cycle
3C-3E	Primary Computer does not detect internal failure and produces arbitrary output
3A-F	Primary Computer watchdog timer trips or loss of output to controller
3D-F	Controller freezes output
3B-F	Primary Computer watchdog timer trips or loss of output to controller
3C-F	Primary Computer watchdog timer trips or loss of output to controller
3E-F	Primary Computer watchdog timer trips or loss of output to controller
1A-DS	Actuated Device stuck
2A-DS	Actuated Device stuck
3A-DS	Actuated Device stuck
1B-DS	Actuated Device stuck
2B-DS	Actuated Device stuck
3B-DS	Actuated Device stuck

1C-DS	Actuated Device stuck
2C-DS	Actuated Device stuck
3C-DS	Actuated Device stuck
1D-DS	Actuated Device stuck
2D-DS	Actuated Device stuck
3D-DS	Actuated Device stuck
1E-DS	Actuated Device stuck
2E-DS	Actuated Device stuck
3E-DS	Actuated Device stuck
AO-DS	Actuated Device stuck
OH-DS	Actuated Device stuck
OL-DS	Actuated Device stuck
1A-OH	Controller fails by outputting only high value
2A-OH	Controller fails by outputting only high value
3A-OH	Controller fails by outputting only high value
1B-OH	Controller fails by outputting only high value
2B-OH	Controller fails by outputting only high value
3B-OH	Controller fails by outputting only high value
1C-OH	Controller fails by outputting only high value
2C-OH	Controller fails by outputting only high value
3C-OH	Controller fails by outputting only high value
1D-OH	Controller fails by outputting only high value
2D-OH	Controller fails by outputting only high value
3D-OH	Controller fails by outputting only high value
1E-OH	Controller fails by outputting only high value
2E-OH	Controller fails by outputting only high value
3E-OH	Controller fails by outputting only high value
1A-OL	Controller fails by outputting only low value
2A-OL	Controller fails by outputting only low value
3A-OL	Controller fails by outputting only low value
1B-OL	Controller fails by outputting only low value
2B-OL	Controller fails by outputting only low value

3B-OL	Controller fails by outputting only low value
1C-OL	Controller fails by outputting only low value
2C-OL	Controller fails by outputting only low value
3C-OL	Controller fails by outputting only low value
1D-OL	Controller fails by outputting only low value
2D-OL	Controller fails by outputting only low value
3D-OL	Controller fails by outputting only low value
1E-OL	Controller fails by outputting only low value
2E-OL	Controller fails by outputting only low value
3E-OL	Controller fails by outputting only low value
1A-AO	Controller fails by outputting arbitrary value
2A-AO	Controller fails by outputting arbitrary value
3A-AO	Controller fails by outputting arbitrary value
1B-AO	Controller fails by outputting arbitrary value
2B-AO	Controller fails by outputting arbitrary value
3B-AO	Controller fails by outputting arbitrary value
1C-AO	Controller fails by outputting arbitrary value
2C-AO	Controller fails by outputting arbitrary value
3C-AO	Controller fails by outputting arbitrary value
1D-AO	Controller fails by outputting arbitrary value
2D-AO	Controller fails by outputting arbitrary value
3D-AO	Controller fails by outputting arbitrary value
1E-AO	Controller fails by outputting arbitrary value
2E-AO	Controller fails by outputting arbitrary value
3E-AO	Controller fails by outputting arbitrary value
1A-0V	Controller fails by outputting 0 vdc output
2A-0V	Controller fails by outputting 0 vdc output
3A-0V	Controller fails by outputting 0 vdc output
1B-0V	Controller fails by outputting 0 vdc output
2B-0V	Controller fails by outputting 0 vdc output
3B-0V	Controller fails by outputting 0 vdc output
1C-0V	Controller fails by outputting 0 vdc output

2C-0V	Controller fails by outputting 0 vdc output
3C-0V	Controller fails by outputting 0 vdc output
1D-0V	Controller fails by outputting 0 vdc output
2D-0V	Controller fails by outputting 0 vdc output
3D-0V	Controller fails by outputting 0 vdc output
1E-0V	Controller fails by outputting 0 vdc output
2E-0V	Controller fails by outputting 0 vdc output
3E-0V	Controller fails by outputting 0 vdc output
F-0V	Controller fails by outputting 0 vdc output
AO-0V	Controller fails by outputting 0 vdc output
OH-0V	Controller fails by outputting 0 vdc output
OL-0V	Controller fails by outputting 0 vdc output
OL-OL	Controller remains in the same state
OH-OH	Controller remains in the same state
F-F	Controller remains in the same state
0V-0V	Controller remains in the same state
DS-DS	Controller remains in the same state
AO-AO	Controller remains in the same state

Table 2.3.8: Grouping of Transitions by Event Type

Transition	***Event***
1A-3A, 1B-3B, 1C-3C, 1E-3E	Secondary Computer failed via watchdog timer trip or loss of output to controller
1A-3A, 1A-3B, 1A-3C, 1A-3E, 1B-3A, 1B-3B, 1B-3C, 1B-3E, 1C-3A, 1C-3B, 1C-3C, 1C-3E, 1E-3A, 1E-3B, 1E-3C, 1E-3E, 3A-F, 3B-F, 3C-F, 3E-F, 2A-F, 3B-F, 3C-F, 3E-F	Primary Computer watchdog timer trips or loss of output to controller
1A-1B, 2A-2B, 3A-3B	Primary Computer loses one input
1A-1D, 2A-2D, 3A-3D	Primary Computer detects an internal failure via self-diagnostics and takes itself down
1A-1E, 1B-1E, 1C-1E, 2A-2E, 2B-2E, 2C-2E, 3A-3E, 3B-3E, 3C-3E	Primary Computer does not detect internal failure and produces arbitrary output
1B-1A, 2B-2A, 3B-3A	Primary Computer's input returns

1B-1C, 2B-2C, 3B-3C	Primary Computer loses other input
1B-1D, 2B-2D, 3B-3D	Primary Computer input does not recover in 1 processing cycle
1A-2A, 1B-2B, 1C-2C, 1E-2E	Secondary Computer detects an internal failure via self-diagnostics and takes itself down
1B-F, 1C-F	Common cause sensor failure
1C-1B, 2C-2B, 3C-3B	One input for Primary Computer returns
1C-1D, 2C-2D, 3C-3D	Primary Computer inputs does not recover in 1 processing cycle
1D-2A, 1D-2B, 1D-2C, 1D-2E	Primary computer releases control of process, Secondary Computer now is the primary computer
2A-1A, 2B-1B, 2C-1C, 2E-1E	Secondary Computer becomes operational
2D-F, 3D-F	Controller freezes output
1A-DS, 1B-DS, 1C-DS, 1D-DS, 1E-DS, 2A-DS, 2B-DS, 2C-DS, 2D-DS, 2E-DS, 3A-DS, 3B-DS, 3C-DS, 3D-DS, 3E-DS, AO-DS, OH-DS, OL-DS	Actuated Device stuck
1A-OH, 1B-OH, 1C-OH, 1D-OH, 1E-OH, 2A-OH, 2B-OH, 2C-OH, 2D-OH, 2E-OH, 3A-OH, 3B-OH, 3C-OH,3D-OH, 3E-OH	Controller fails by outputting only high value
1A-OL, 1B-OL, 1C-OL, 1D-OL, 1E-OL, 2A-OL, 2B-OL, 2C-OL, 2D-OL, 2E-OL, 3A-OL, 3B-OL, 3C-OL, 3D-OL, 3E-OL	Controller fails by outputting only low value
1A-AO, 1B-AO, 1C-AO, 1D-AO, 1E-AO, 2A-AO, 2B-AO, 2C-AO, 2D-AO, 2E-AO, 3A-AO, 3B-AO, 3-CAO, 3D-AO, 3E-AO	Controller fails by outputting arbitrary value
1A-0V, 1B-0V, 1C-0V, 1D-0V, 1E-0V, 2A-0V, 2B-0V, 2C-0V, 2D-0V, 2E-0V, 3A-0V, 3B-0V, 3C-0V, 3D-0V, 3E-0V, F-0V, AO-0V, OH-0V, OL-0V	Controller fails by outputting 0 vdc output
1A-1A, 2A-2A, 3A-3A	Primary Computer remains operational
1E-1E, 2E-2E, 3E-3E	Primary Computer remains operating incorrectly
OL-OL, OH-OH, F-F, DS-DS, AO-AO	Controller remains in the same state

2.4 Application of a Safety Quantification Methodology to the Digital Feed Water Control System for Failure Data Generation

2.4.1 Background

As indicated in Section 1.2, digital I&C systems are functionally different from traditional analog and discrete designs, and almost always include the use of microprocessors, programmable

logic devices, A/D and D/A conversion electronics, displays, power conditioning electronics, real-time data buses, and multiplexing. Furthermore, these systems employ different fault isolation and containment techniques to achieve sufficient independence and fault tolerance than their predecessors (Section 2.3). Taken together, the introduction or consideration of new digital I&C systems into nuclear power plants raises concerns regarding the possibility that the fielding of these I&C systems may introduce unknown or unanticipated failure modes, or design errors in the software in the redundant channels of a safety system that could lead to a common mode failure of the safety system function (Section 1.2).

In order to prevent such situations from arising, there is a need for digital I&C systems to be dependable across a wide spectrum of perceived threats, faults, and failures; that is, these systems must be able to detect faults and initiate appropriate mitigation actions upon the detection of these faults to ensure overall safety of the I&C system. In recent years, significant effort has gone into improving design methodologies for safety critical systems, assessment methods, and updating regulatory industry standards and regulatory guidelines (e.g. RG 1.152 [102], IEC 61508 [103], RTCA-DO-254 [104] to ensure that digital I&C systems can be designed and assessed to the high safety requirement levels required for highly critical applications. The goal of a quantitative safety assessment methodology is to provide a generic systematic way of characterizing the dependability behavior of embedded systems (e.g. I&C systems) in the presence of faults [105-107]. We recognize that the key features of this methodology should state the safety metrics that are pertinent for a given domain, how those metrics are quantified with supporting data, the way the data for the metrics are obtained, and the domain in which the measures and metrics are valid and meaningful. In the Section 2.4.2, we introduce and discuss some preliminary concepts of dependable systems to set the context. Section 2.4.3 shows how fault injection can be used as dependability assessment method and Section 2.4.4 overviews of the quantification of this dependability assessment. The experimental setup for the design and implementation of the fault injection environment is described in Section 2.4.5. Section 2.4.6 gives example results from a fault injection campaign, including error classification and handling of common cause failures. Section 2.7 describes how the results of the injection experiments along with hardware failure data can be used to estimate the failure rates and/or failure probabilities on demand for specific failure modes for use with the DFM and the Markov/CCMT method.

2.4.2 Concepts of Dependable Systems

A dependable real-time system has the ability to provide its intended, expected and agreed upon functions, behavior, and operations in a correct and timely manner. Dependability may be defined as *the trustworthiness of a system such that reliance can justifiably be placed on the service it delivers* [106]. A service delivered by a system is its behavior, as it is perceived by its users. A user is another system (human or technical) which interacts with the former.

2.4.2.1 The Attributes of Dependability

The attributes of dependability are the primary means by which we specify the quantitative and qualitative requirements of a system. We define some basic terms and concepts related to dependability attributes [105]:

> **Definition 1: *Dependability*,** the property of a computer system such that reliance can justifiably be placed on the service it delivers, which is a qualitative system attribute that is quantified through the following terminologies.
> **Definition 2: *Reliability***, a conditional probability that the system will perform correctly throughout the interval [t_0, t], given that the system was performing correctly at time t_0, which is related to the continuity of service.
> **Definition 3: *Availability***, a probability that a system is operating correctly and is available to perform its functions at the instant time t, which is related to readiness for usage.
> **Definition 4: *Safety*,** a probability that a system will either perform its functions correctly or will discontinue its functions in a manner that does not disrupt the operation of other system or compromise the safety of any people associated with the system, which is related to the non-occurrence of catastrophic consequences on the environment.

2.4.2.2 Impairments to Dependability

Faults, errors and failures affect the system's ability to deliver its dependability attributes(e.g. safety, reliability, performance). Hence, they are called the *impairments* of dependability. From our experience, there are important reasons to establish and evaluate the hierarchal, causal relationships between impairments, together with their origins and effects. First, it enables systematic quantitative and qualitative modeling and analysis of the reliability and safety of the system. Second, it serves as a basis for specification and requirements of an assessment of a dependability metric. We begin by defining the meaning of these expressions more precisely [105]:

> **Definition 5**: A ***fault*** is a physical defect, imperfection, or flaw that occurs within some hardware or software component. A fault is the adjugated cause for an error.
> **Definition 6**: An ***error*** is the manifestation of a fault. Specifically, an error is a deviation from accuracy or correctness.
> **Definition 7:** A ***failure*** is the nonperformance or inability of the system or component to perform its intended function for a specified time under specified environmental conditions.

Implicit in the definitions of the above terms is a cause-effect relationship. The well known 3-universe model depicted in Fig. 2.4.1 shows the relationship between faults, errors, and failures.

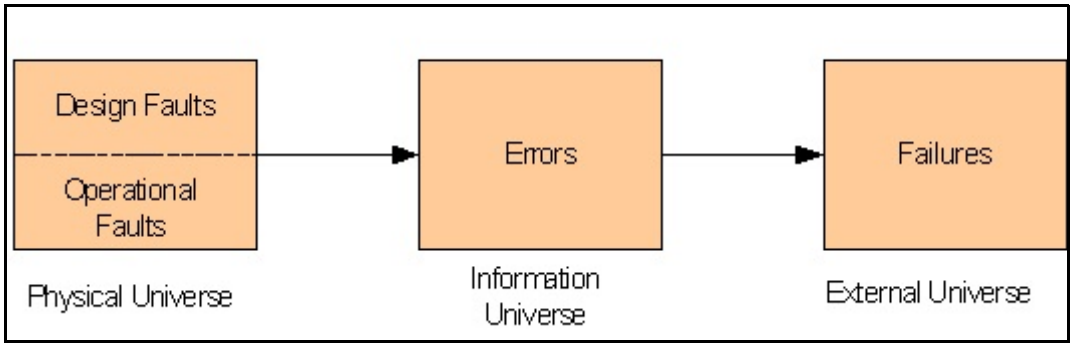

Figure 2.4.1 Cause-Effect Relationship Among Faults, Errors, and Failures using the 3-Universe Model

Essentially, Fig. 2.4.1 expresses a relationship that shows that failures are caused by errors, which are caused by faults. Associated with each term is a domain of effect; for example, faults are associated with the physical universe. Two categories of faults are possible in the physical universe: *operational faults and design faults*. *Operational Faults* include faults associated with semiconductor devices, mechanical elements, power supplies, and other physical entities that make up a system. These types of faults usually include fault types such as signal crosstalk in IC's, transistor failures in IC's, wear-out, oxidation, etc.

On the other hand, *design faults* include hardware and software flaws that are usually due to the inability to anticipate or fully consider certain interactions in hardware and software during system specification, design and implementation [105]. Design flaws or faults are activated when specific input stimulus and computer states are present, e.g. a unique execution path is taken by the computer-based machine element. Design flaws are not randomly occurring actions per se; they are deterministic events since they occur every time the same input and state conditions happen.

Errors are associated with the information universe, which contains units of information [105]. More precisely, information in a computer is characterized by symbols, and the interpretation and manipulation of those symbols. Errors can corrupt symbols, rendering them into different symbols, non-symbols or reconstitute the interpretation of symbols. Errors in the information universe are usually manifested as bit flips in the data and/or instruction symbols. Finally, Fig. 2.4.1 shows that *failures* are associated with the external universe, which is where the user of the system eventually sees the effects of faults and errors.

The linkages between physical, information, and external universe is an abstract point of view. Basically, the physical represents the hardware of the system, such as semiconductors, microelectronic devices, power supplies, data distribution networks, etc. Thus, a fault is physical defect or alteration of a component. The second universe basically represents the execution behavior of the digital system. Because execution of the digital system is carried out on units of information, we call it the information universe. Errors occur on data words within the computer, instructions, or transmission of information from place to another. The external

universe (or users universe) is where the user or another system see's the effect of faults and errors. This is where failures manifest. Failure is any deviation from the expected or desired service/behavior of the system.

2.4.3. Fault Injection as Dependability Assessment Method

2.4.3.1 Introduction

I&C systems in which safe operation and high reliability is required, demands strong verification of the fault tolerance and safety protection aspects of the I&C system in the presence of faults. This is necessary because the failure of the fault tolerance mechanisms could lead to failures of an I&C system. *Fault Injection* is defined as the dependability validation technique that is based on the realization of controlled experiments where the observation of the system behavior in the presence of faults is explicitly induced by the deliberate introduction (injection) of faults into the system [107]. Fault injection stresses the fault handling and management aspects of the system under assessment to ultimately collect crucial data via fault injection campaigns. These typically include, coverage of faults, error latency, recovery time. In most fault tolerant systems, the FDIR (Fault Detection, Isolation and Recovery) software or online diagnostic functions may account for as much as 40%-50% of the executable system software code [108]. This code is rarely exercised in the real world because faults are infrequent occurrence. This FDIR code is vital for system compliance to dependability, and can only be effectively tested and validated by realistic fault injection campaigns.

We should state clearly, that fault injection is but one crucial technique in a comprehensive framework for dependability assessment. Fault injection has it's limitations. It does not represent design flaws, specification flaws very well, or at all. For these impairments, the use of contemporary design assurance and fault avoidance methods, such as, formal specifications analysis, model checking, and dynamic code analysis methods can be used to minimize the impact of design faults. However, it is generally accepted and demonstrated that fault injection methods are useful in *finding* design flaws in fault tolerant systems. Fault injection is specifically designed to exercise and stress the FDIR mechanisms of the system under a variety of conditions – both nominal and off-nominal. A number of experience reports from academia, industry, and government where fault injection has found design flaws in the fault tolerance mechanisms of computer based safety critical systems [109]. While fault injection has its limitations, and these are well-known, we view it as a natural partner to contemporary design assurance techniques, testing and fault avoidance methods.

Fault injection techniques can be divided into three general categories; (1) fault injection at the hardware level on an operational prototype, (2) fault injection at the software level on an operational prototype, and (3) fault injection into a simulation of the system [110]. See [110] for detailed review of fault injection methods. The type of fault injection used in this effort is of type 2: fault injection at the Instruction Set Architecture of microprocessor based computer. Section 2.4.6 of this report describes the design and operation of the fault injection environment. The faults to inject for these fault injection mechanisms are typically selected at random from the

total fault space of the system. The data collected during fault injection is used to statistically and accurately estimate the crucial parameters of the analytical safety and reliability models.

Selecting the appropriate fault model is a crucial decision. It is frequently estimated that 80% or more of all hardware faults which occur on ground based computer applications are transient in nature [111]. The effects may last for milliseconds, or they may persist until the system is re-booted or refreshed. This number is increasing each year as semiconductor technology scales downward [112]. In recent years, the impact of Single Event Upset (SEU's) due to neutron particles and high intensity radiation fields on digital systems has become even more significant due to the aggressive scaling of the semiconductor manufacturing processes. Lower transistor features and lower voltages, high performance circuit designs, have led to higher performance and, also, have increased circuit sensitivity to particle and radiation induced errors. Contemporary deep sub-micron technology processes are much more sensitive to particle induced upsets than the previous generation of process technology. Thus, for this effort, the use of the transient fault model was selected based upon these facts and based upon historical and contemporary failure data (including operational data from the plant) of ground based systems. The other fault model that is of importance is the permanent fault model. The permanent fault model represents a physical defect in the hardware that always is present and is activated when certain input and state conditions arise. The use of the permanent fault model is limited in this effort.

2.4.3.2 Fault Injection Space

The purpose of this section is to describe the parameter space that characterizes fault injection. A fault is injected into a system by defining five parameters of the system fault. The external input and current system state is one fault injection parameter which is implied in most fault injection experiments. The external input is typical called the operational profile or simply the environment domain of the system. For instance, in a control system the plant is controlled by an active controller, the state of the plant is fed back to the controller at every sample instance. In this example, the space of the sensor inputs is what we call the operational profile of the controller. The controller uses this sensor information to computer the next command according to a control objective. The current external input and internal system state is represented by the variable σ. Another parameter which affects the definition of a fault injection experiment is the fault occurrence time (or fault start time) which is represented by the variable t. The duration of a fault, given by Δ, is another fault injection parameter. The location of the fault and fault type are the last two parameters which define a fault injection experiment. The fault type is given by the variable f_m, while the fault location is given by the variable l.

One of the main goals of fault injection is evaluate the error and fault handling mechanisms of a fault tolerant system. An extremely important parameter in the design and assessment of fault tolerant systems is *fault coverage, C*. The fault coverage available in a system can have tremendous impact on the reliability and safety of a system [113-115]. The intuitive definition of fault coverage is that it is a measure of the systems ability to perform fault detection, fault isolation, and fault recovery. In that respect, Type II failure events such as communication

failures and failures arising from competition for resources are captured in fault coverage. Fault coverage is mathematically defined as the conditionally probability that given the existence of a fault, the system detects and recovers. Thus,

$$C = Pr(\text{fault detected} | \text{fault existence}) \quad (2.4.1)$$

Common cause software failures in redundant systems (e.g. platform based) can be captured with coverage since system failure due to common cause is also part of uncoverage, *1-C*. A general mathematical expression for the system coverage as a function of the above fault injection parameters can now be derived as

$$C = c(t, \Delta, \sigma, l, f_m) = \Pr\{Y = 1 \mid (t, \Delta, \sigma, l, f_m)\} \quad (2.4.2)$$

where Y is a random variable which is 1 if the fault is covered and 0 if it is not, *C* is the fault coverage, and is the probability of a covered fault over the entire fault activation space. This probability can also be derived by taking the expected value of for the entire fault activation space. Unfortunately, it is difficult if not impossible to derive c(●) through analytical techniques. The coverage of the system is typically obtained by sampling this five dimensional space and obtaining a statistical point estimate. The evaluation of each sampled data point in this space via fault injection results in either a 1 or 0 value for *C*. The binomial function is used to represent the system response. The point estimate for the system fault coverage is obtained

$$\hat{C} = \frac{1}{n} \sum_{i=1}^{n} y(t_i, \Delta_i, \sigma_i, l_i, f_{mi})$$

where $\hat{C}$ is the point estimate for the system fault coverage, and *n* is the number of fault injection experiment.

From Eq. (2.4.3) we can see that the estimation of fault coverage via fault injection is dependent on five parameters. Thus, in a well designed fault injection tool environment, support for controllability and observability of these parameters needs to be available. We address this requirement in the design and implementation of our fault injection environment. The next section describes the methodology used to estimate the important parameters needed quantify a dependability metric such as safety, probability of system failure, reliability, and coverage. The methodology is driven in large part by the needs of Eq. (2.4.3). Specifically, the methodology addresses the representation and characterization of each parameter (among others) in a operational fault injection scenario.

2.4.4 Overview of the Quantitative Dependability Assessment Methodology

A distinguishing feature of the assessment methodology is the synergistic relationship between fault injection process and analytical dependability models. Dependability models allow for formal (meaning mathematical) tractable assessment of the I&C system based upon

measurable and observable parameters of the system under evaluation. The essence of the dependability model is to allow quantification (e.g. calculation) of safety or reliability based upon observable and measurable parameters of the system under assessment. Fault Injection is used to estimate certain crucial parameters, like coverage. As stated earlier, fault injection stresses the fault handling and management aspects of the I&C system under assessment to ultimately collect crucial data (e.g. such as error detection coverage) via fault injection campaigns. The general approach is to inject randomly selected faults into the system (on a working prototype, for example) to determine if the fault processing capabilities of the I&C system detect and mitigate the effects of the faults. This data collected during fault injection campaigns is used to statistically and accurately estimate the crucial parameters of the analytically safety model. These parameters typically include, fault class coverage, recovery and repair time, fault latency, error detection mechanism coverage, and error detection time [107]. Once these parameters have been accurately estimated, then they are instantiated into the dependability models to calculate the safety, reliability, or probability of failure of the system. A detailed report of the model based safety assessment process can be reviewed in [115]. Figure 2.4.2 shows the flow of the methodology. We present this methodology in a step by step basis to give the reader and practitioner a feel for the methodology.

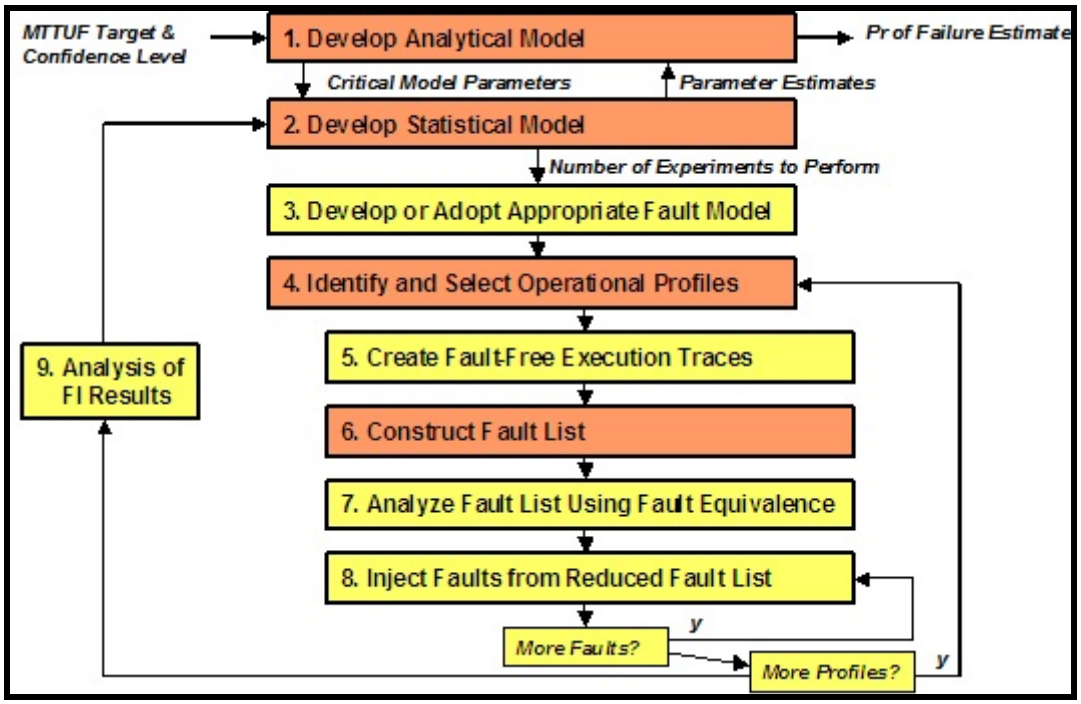

Figure 2.4.2 Operation of the Quantitative Dependability Assessment Process

Step 0: Defining or Selecting a Dependability Metric. The assessment process begins with defining or selecting the dependability metric of interest. The metrics that can be used in I&C systems include but are not limited to, system reliability, probability of coincident failure, system safety, probability of failure on demand, mean time to system failure, mean time to unsafe

system failure, and steady state unsafe system failure. For instance, an actuation system would be probably be more accurate characterized by probability of failure on demand, and a instantaneous availability metric rather than a MTTF metric. So, how the system is employed in the context of the plant is very important to the selection of an appropriate metric. In the case of the DFWCS, it is a control system that operates continuously and should have a significant degree of reliability over it's mission time. From the DFWCS requirements documents, the stated reliability requirement is 99.9% operational capability over a unstated operational mission time. The presumed operational mission time is 11 months (e.g. allowing for 1 month for outage). Since the DFWCS is continuously operating 24 hours a day, 7 days a week, 365 days a year, the metric that best characterizes the dependability of this system is probability of system failure over the mission time (e.g. 11 months).

Probability of system failure = 1 - R(t) = 1-.999 =.001 failures/11 months

On a hourly scale.

.001 system failures/(1)(8000 hours) =1.2x10-7 system failures/hour.

This is a very low probability of system failure, as it should be. A very low probability of system failure infers a high degree of fault detection and recovery coverage by the system. In the DFWCS documentation there were no quantitative requirements for the fault coverage capabilities of the various fault detection mechanisms of the DFWCS. Only qualitative FMEA information was available and this is most typical of current I&C systems. However, it is possible to deduce how effective each fault detection mechanism or component needs to be in order to meet the stated unreliability requirement of .001 system failure per year. In this case, we have inferred the coverage numbers by modeling the system via Markov methods and calculating the minimum fault coverage required for each component, given the desired unreliability requirement of 0.001 failure/year. The result is that the minimum coverage required to meet the unreliability requirement is about .99995 for each component. This value will be used later to help estimate the number of fault injections trials needed for a narrow interval estimate of coverage.

Step 1: Development of the Analytical Dependability Model of the System. An analytical safety or reliability model is developed from the system architecture and inter-component dependencies. The input to the model development process comes from a variety sources. These include architectural specifications, software specifications, fault tolerance specifications, operating systems specifications, control laws, and hazards analysis documents. In fact, this methodology may use several models to determine the compliance of a system with respect to its dependability requirement. These models are typically represented as dynamic fault trees, Markov models, dynamic flow graph models, finite state models to model functional behavior (Section 1.2.2), or combinatoric models.

The dependability metric of interest (such as probability of a unsafe failure) influences the critical parameters of the model. These model parameters often include fault coverage values

for components, recovery rates, and critical failure rates for the various components of the system. Equally important is stating the assumptions the models make in light of incomplete knowledge of the systems. These assumptions must be assessed in regard to the model and the parameters of the model for the purpose of exploring the sensitivity of the model predictions to the stated assumptions. See [116] for more details.

The development of the analytical models of the DFWCS are found in Section 2.3 of this report. As an example of an analytical model, is the main computer of the DFWCS. This model is a Markov process model showing several modes of failure (see Fig. 2.4.3 below). The model is intended to be illustrative of the typical parameters used in a analytical model.

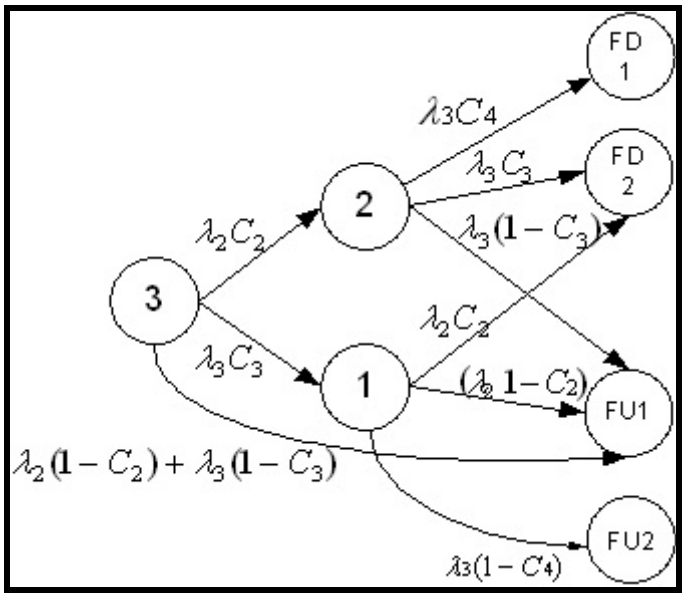

Figure 2.4.3 Markov model of main computer of the DFWCS.

State FD1 represent the failure of the computers (both main and back up) in such away that the loss or inappropriate control command from the was detectable by the PID Controllers. In this case, the remaining functioning system (e.g. the PID controllers) reach a “fail-safe state” by switching to a manual mode and setting the outputs of the PID controllers to nominal output levels and annunciating the failure to the operator. State FD2 represents the failure of the controllers to establish proper feedback through the analog input backplane. Again, switch over to manual mode is the proper fail-safe corrective action.

State FU1 represents a failure where the loss of set point or inappropriate control action is undetected for substantial period of time resulting in the improper control of the valves or pumps. FU2 is a failure where the loss of feedback, improper measurement is not detected resulting in improper or loss of control of the valves and pumps for unacceptable amount of time.

The important parameters in this model are the following:

λ_n : The rate of failure for the component as given in failures/hour
C_n : The coverage of a specific fault by the fault detection mechanisms of the DFWCS

We have previously defined the parameters that characterize coverage in Eq. (2.4.3). However, λ_n is a empirical parameter typically based on failure data from the plant, commercial device failure rate databases, and proprietary vendor databases. It can not be estimated by a fault injection campaign. There are other means to estimate the device failure rate based upon accelerated life testing methods where the component is placed in a stressful environment and forced to fail prematurely. These methods are expensive and typically only conducted by the manufacturer of the device. In the next section we will examine the issues of gathering failure rate data for components.

Step 2: Statistical Model development. The purpose of the statistical model is to provide a formal basis for a estimating the critical model parameters that are required by the analytical model. Once the critical parameters have been identified in Step 1, a sound mathematical basis for estimating these parameters must be used to gain confidence in the model prediction capability. This statistical model supports four specific needs of the methodology:

1 Quantify and characterize the uncertainty of model parameters.
2 Characterize and define the assumptions of model parameters.
3 Statistically estimate based on the assumptions of the model and model parameters the number of observations are required to estimate a parameter to a known confidence level.
4 Calculate the number of fault injection trials in a fault injection campaign required to calculate the coverage estimate of the component

Generally speaking, safety and reliability analyses are often confronted with limited or no historical “hard data” on the failure rates of components. In these cases, engineering judgement needs to be used. How to use such judgement depends, however, on incorporating uncertainty in our engineering judgement. It is a fact of life that failure data has measure of uncertainly associated with it. Estimating the degree of this uncertainty is largely based on how much data is available to the engineers, and what methods the engineers to estimate uncertainty. In recent years, Bayesian approaches that allow the use of engineering judgement to establish subjective uncertainty measures related to what the true values of the statistical quantities are have become more mainstream. Some methods combine the frequentist and the Bayesian approach. The combined classical and Bayesian approach is also referred to as the probability of frequency framework [117]. With this approach, combined uncertainty is related to two levels, i.e. the prediction of events by the model and the values of the probabilities and failure rates. In this effort we used a combination of methods to reach a consensus on the failure rate data. Below we describe the process we used in this methodology to estimate the failure rates of the various components of the DFWCS. This process is not be inferred as the

only way, or the preferred way, but it represents a way to deal with information when it there is uncertainty involved.

Our approach to acquiring the hardware failure rate of the components was to use three key pieces of information:

1 Acquire and analyze the actual failure data from the DFWCS.
2 Use a commercial failure data base.
3 Conduct interviews with selected vendors to acquire “off-the record” failure data information.

Failure data was supplied to us by the plant engineers on all of the observed permanent failures of the system since its installation in the plant 13 years ago. Overall, the component failure records are very robust. Several key components of the DFWCS had not failed at the time of this study (namely the PID controllers). In that respect, the next step was to use a high quality commercial data bases of hardware failure data information (e.g. the PRISM database[118]) and utilize this collected information to set the base failures rates of the components of the DFWCS. Where real DFWCS failure data exists from plant failure logs, the use of a Bayesian updating method to adjust the failure rates to the observed failures in the benchmark I&C system was employed. The following assumptions are made with this process:

1 The data contained in the RAC database and the field data are beyond the infant mortality part of the failure.
2 The failure rate data contained in the databases represents the stable failure period of the failure curve.

Final step was to interview key vendors about their equipment reliability in order to obtain more credible information about the failure rate calculations. It should be noted that vendors are often reluctant to share actual failures with clients, and when they do, it is usually on an off-the-record basis. Based on our conversations with two vendors, the failure rates of the CPU’s closely matched the observed failures at the plant. No hard data was forthcoming on the PID controllers, but we were told that a failure rate of 10e-6/hour was not unreasonable. All failures were permanent failures, and did not include transients in the reports and failure logs. Transient failures were noted on the CPU and watchdog timer modules of the DFWCS. That is, failures for which a reboot or re-start would correct the problem. There were at least 10 such events noted in the event logs. All vendors told us that they recommended assigning a uncertainty factor of 10 to nominal failure calculations.

Table 2.4.1 shows the failure rate estimation for key components of the benchmark DFWCS. At this point time we have not conducted an uncertainty analysis on the variability of the failure rate as a function of system failure.

Component NO	Component Name	Failure Rate	Estimated Parameter (per hour)
Component 1.1(A/B)	Power Level Sensor	λ_{11}	4.5×10^{-7}
Component 1.2(A/B)	Steam Flow Sensor	λ_{12}	1.87×10^{-6}
Component 1.3(A/B)	Water Flow Sensor	λ_{13}	1.5×10^{-6}
Component 1.4(A/B)	Water Temp Sensor	λ_{14}	1×10^{-6}
Component 1.5	Water Level Sensor	λ_{15}	1×10^{-6}
Component 2	Main Controller	λ_{2}	3.65×10^{-5}
Component 3	Backup Controller	λ_{3}	3.65×10^{-5}
Component 4	Main Flow Valve PID	λ_{4}	1×10^{-6}
Component 5	Bypass Flow Valve PID	λ_{5}	1×10^{-6}
Component 6	Spare PID	λ_{6}	1×10^{-6}
Component 7	Main Flow Valve	λ_{7}	1.2×10^{-6}
Component 8	Bypass Flow Valve	λ_{8}	1.2×10^{-5}
Component 9	Feed Water Pump PID	λ_{9}	1×10^{-6}
Component 10	Feed Water Pump	λ_{10}	3.13×10^{-5}

Table 2.4.1 Failure Rate Estimations for DFWCS Main Components

Another important parameter that must be estimated in the analytical models is coverage. Coverage represents the ability of the I&C fault detection mechanisms to detect a fault, given a fault is present. Coverage is very important parameter in estimating the system safety, reliability, unreliability, and other safety related metrics. Thus, to accurately estimate the coverage of a system, a fault injection statistical experiment must be designed to estimate the parameter based on finite number of fault injection trials. Several statistical approaches to estimating the number of fault injections required to obtain an accurate coverage estimate and confidence level is discussed in[116]. In this work we use a single sided confidence interval to estimate the upper bound on the number fault injection trials needed to accurately estimate the coverages.

The estimation of the fault coverage can be modeled as a binomial random variable X_i,

where $X_i = \begin{cases} 1\forall \\ 0\forall \end{cases}$

The fault injection experiments are performed to generate X_1, ……X_n, where each X_i is assumed to be independent and identically distributed. The expected value of the random variable is *E (X)* = *C*, and the variance of the random variable is Var (X) = C(1-C).

Given the statistic

$$S_n = \sum_{i=1}^{n} X_i, \tag{2.4.4}$$

the probability of $S_n = j$ is

$$P(S_n = j) = \begin{bmatrix} n \\ j \end{bmatrix} C^j (1-C)^{n-j}, 0 \le j \le n$$

If S_n out of n faults are observed to be covered, then C, the lower side of the confidence interval, satisfies the following equation

$$P(S_n > s_n) = 1-\gamma \tag{2.4.6}$$

where $P(S_n > s_n)$ is the probability S_n is greater than or equal to s_n, given that the fault coverage value equals C_l, and *g* is the confidence coefficient. The single-sided confidence interval is calculated as

$$1-\gamma = \sum_{j=s_n}^{n} P(S_n = j) = \sum_{j=s_n}^{n} \begin{bmatrix} n \\ j \end{bmatrix} C^j (1-C)^{n-j}, 0 \le j \le n$$

Now, consider the case where all the faults are covered. In this case,

$$1-\gamma = \sum_{j=s_n}^{n} \begin{bmatrix} n \\ j \end{bmatrix} C^j (1-C)^{n-j} = C_l^n$$

Rearranging Eq. (2.4.13) and solving for *n*

$$n = \frac{\ln(1-\gamma)}{\ln C_l} \tag{2.4.9}$$

where n defines the number of fault injections needed to satisfy a given confidence level.

For instance, for the DFWCS estimating the number of fault injections to establish a coverage

factor of .99995 at a 99% confidence level, then using Eq. (2.4.9)

n = 92,101 fault injections are required.

As the coverage value increases (e.g. *1- C* = 10^{-7}) an extremely large number (millions) of faults need to be injected in order to obtain an estimate at a high confidence level. It is not possible to carry out this large a number of fault injections within a practical time constraint without some form variance reduction methods. Therefore, methods that reduce the sample size have to be explored. Fault expansion is one technique that has emerged in an attempt to achieve such a goal [114]. The key idea of fault expansion technique is to identify the *equivalence class* to which an injected fault belongs after the fault is injected and the outcome is determined. The faults in the same class are "equivalent" because they affect the system in an identical way. Therefore, if the injected fault is detected, all faults in its equivalence class will be detected when injected. As a consequence, we only need to inject one fault per equivalence class. Thus, the fault expansion method intends to yield the desired fault coverage interval estimate with a smaller number of fault injections. With fault expansion methods employed 92,000 fault injections is easily obtainable over a reasonable period of time (e.g. weeks).

Step 3: Development of a High Level Processor Fault Model. In Eq. (2.4.3) (the coverage equation) the parameters Δ, f_m and *l* are respectively defined earlier as the fault duration, fault type, and fault location. The parameter f_m, is crucial parameter because it represents the type of fault(s) that are representative of the faults that the systems is expected to encounter, and tolerate. Selecting the appropriate fault model is a crucial decision. It is frequently estimated that 80% or more of all hardware faults which occur on ground based computer applications are transient in nature. For this task, we developed a high level processor fault model to represent both transeunt and permanent faults. The purpose of this model is to accurately represent the types of faults that can occur in a computer based I&C system. The fault model is used to generate the fault space, F. F is usually a multi-dimensional space whose dimensions can include fault characteristics such as location, time, and value, as shown in Fig. 2.4.4. Here, time represents the time of occurrence and duration, location is where the fault occurs within the system under analysis, and value represents the form of the corruption. Note that value can be something as simple as a mask (e.g. corrupting a memory or register location with a overwrite operation), or something more complex that is state dependent.

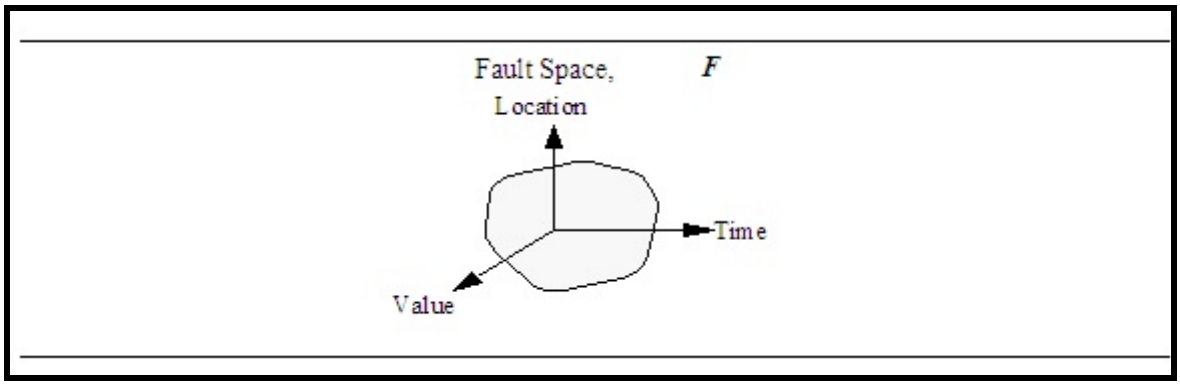

Figure 2.4.4 Fault Space Characterized by Location, Value, and Time

In general, completely proving the sufficiency of the fault model used to generate and characterize F is usually very difficult. The fault modeling of the applied processor is obviously the most problematic, due to the large fault space of a complex processor and the interactions with software. It is more traditional to assume that the fault model is sufficient, justifying this assumption to the greatest extent possible with experimental data, historical data, or results published in literature. This is the approach we took, and we used in house processor modeling and simulation to support the justification of our proposed processor fault model used in this research effort.

In support of this task, we developed a behavioral-level generic processor fault model that represents the faulty behavior of a general-purpose, implementation-independent microprocessor. This work builds on at least 20 years of research to develop a behavioral-level fault model that accurately represents the faulty behavior of a generic RISC like processor. In the past early research focused on using the fault models for developing test scenarios for detecting the various faulty behaviors defined by the fault model. Our efforts to develop a generic processor fault model builds on [114].

The generic processor model considered performs a basic fetch-execute instruction cycle typical of a von Neumann architecture. As such, it contains a control unit, which operates as a synchronous finite state machine, and a data-path, consisting mainly of combinational logic and some storage elements, which performs the information processing within the processor. The data-path contains a register file that contains general-purpose and special-purpose registers, a program counter, an arithmetic and logic unit (ALU), and a fetch and decode logic block. In addition, the processor contains an interface that allows for communication with entities external to the processor. And finally, the processor contains internal signals that allow for communication between the data-path and control unit.

The effectiveness of the generic processor fault model was demonstrated by developing another processor model, a gate-level model of an actual 32-bit RISC processor (with a structure similar to the generic processor) and performing simulation-based fault injection

experiments to demonstrate that all faulty behaviors that are produced by gate-level faults in the gate level processor model (for instance, stuck-at faults, bridging faults, open faults, etc.) can be represented by the higher level behavioral fault model. Any unusual faulty behavior discovered as a result of these fault injection experiments was used to augment the behavioral-level generic processor fault model.

Once the generic processor fault model was developed, an analysis was performed to ensure that the experimental environment that is used to perform the fault injection experiments fully supports the fault model. The fault model we verified is shown below in Fig. 2.4.5. The model follows the basic premise of a programmers model. That is, the fault model involves the modification of the software executing on the I&C system under analysis in order to provide the capability to modify the system state (both processor registers and memory) according to the programmer's model view of the system. The experimental fault injection environment we developed to inject faults into the processors of the DFWCS is based on a software implemented fault injection method employing a in-circuit emulator to control and trace the execution flow of the processor. The details of the implementation of the fault injection environment are in Section 2.4.6. In addition, we compared the generic behavioral level processor to the Intel 486-DX5 which is the resident processor in the benchmark digital feed water control system. The purpose of this exercise was to determine how functionally equivalent the 486 processor was to our generic behavioral processor model. We found that the two processors had strong functional equivalence, and therefore the generic processor fault model could apply to the 486 processor. See [119] for more details.

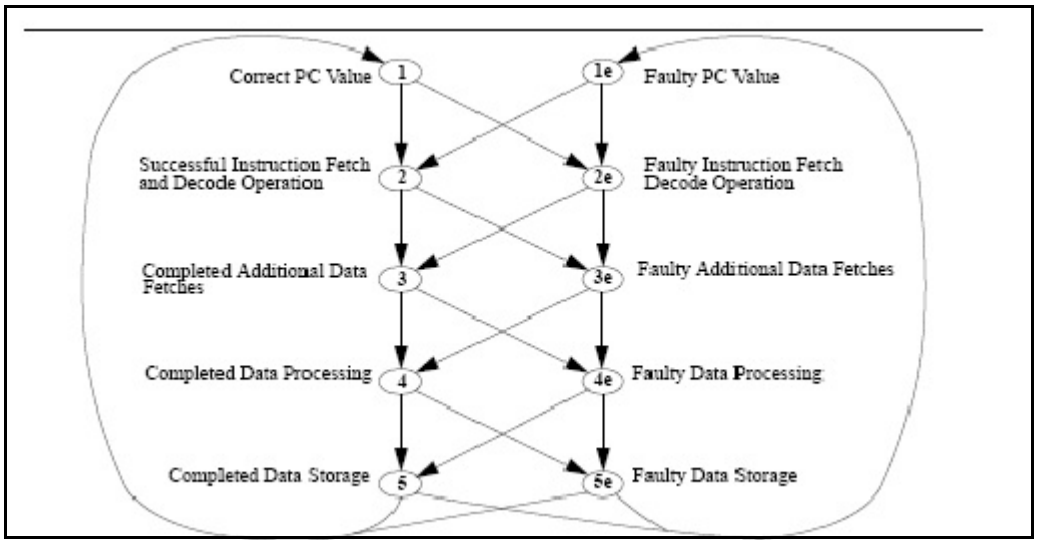

Figure 2.4.5 Instruction Set level Behavioral Fault Model

Step 4: Selection of the Operational Profile. Referring to Eq. (2.4.3) again, the last parameter in the coverage equation is σ. The σ represents the input and state space of the system under fault injection or the input operational domain of the system. We also call σ the

operational profile – the input space of a target system. Operational profiles to be used in the fault injection experiments must be selected to be representative of the system under various modes of operation and configuration. System configurations may invoke different hardware and software modules, and it is important that the experiments include sufficient combinations of these to ensure a thorough evaluation of their behavior in the presence of faults. Similarly, each configuration will generally have many modes of operation, and not all of these provide appropriate times during which to perform fault injection experiments. For example, a typical fault injection scenario would include a startup sequence used to bring the system to a well defined state prior to the period during which faults may be injected. From a statistical point of view, since the startup time is negligible compared to the operational time, the system startup is not a representative operational profile.

Most real-time hardware/software systems operate on an event-triggered basis; that is, when an event occurs, it is immediately serviced by the system. Of course, the event itself can generate other events to service. If there is no event from the outside environment, only a reduced set of software and hardware resources may be used (cyclic idle tasks, diagnostics, and so forth). This portion of the operational profile will be referred to as a system light workload. A transient fault that occurs during a system light workload has a high probability of not being activated as an error. If activated as an error it may not produce a failure until an input/output activity starts. Also a permanent fault that occurs during a system light workload has a high probability of not being activated as an error by the idle tasks, but it has a high probability of being activated as an error and detected by the system diagnostics. This assumes that the observation interval is longer than the time in which diagnostics are completed. Therefore, a system light workload, followed by some input/output, has to be in the selected operational profiles to properly exercise the system.

If there are many events from the outside environment, a majority of the software and hardware resources are used. This portion of the operational profile will be referred to as a system heavy workload. In general, when testing the functionality of a given system, the operational profiles should be selected to exercise as much of the system as possible (assuming that exhaustive testing is infeasible due to time and resource limitations). This becomes especially important in a safety evaluation effort using fault injection since it has been shown that certain operational profiles can mask faults within the system [120]. A transient fault that occurs during a system heavy workload has a high probability of being activated as an error, and it could quickly produce (if not detected) a failure due to the high input/output activity. Also, a permanent fault that occurs during a system heavy workload has a high probability of being activated as an error before being detected by some diagnostic. It also has a certain probability of being activated as an error and detected by the system diagnostics, assuming that the observation interval is longer than the time in which diagnostics are completed.

In summary, the operational profile shown in Fig.2.4.6 is considered to be representative of the types of operational profiles that need to be used for the fault injection experiments in order to properly exercise complex hardware/software systems. As such, operational profiles like the one shown in Fig. 2.4.6 are used during the fault injection experiments, the results of which are

then used to statistically estimate the fault coverage for the various system components. Figure 2.4.6 operational profiles consist of the following phases:

1. A system start-up (Phase 1)
2. A system light workload (Phase 2)
3. A system heavy workload (Phase 3)
4. A short system light workload (Phase 4)

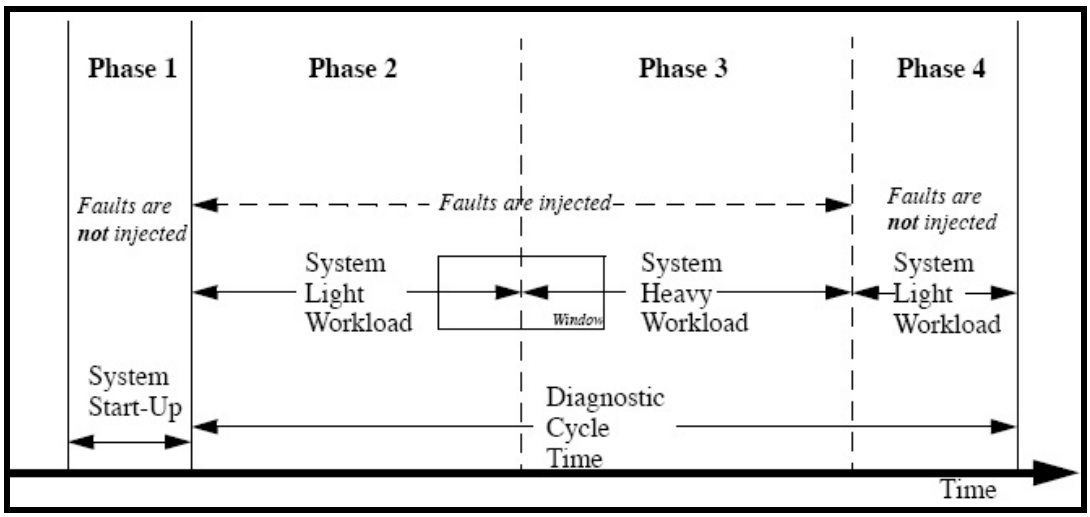

Figure 2.4.6 Typical Operational Profiles Applied During Fault Injection

Faults are injected only during Phase 2 and Phase 3, but not during Phase 1 and Phase 4 due to the justifications presented above. It is worth noting that the window shown in Fig. 2.4.6 can be seen as the operational profile to be applied for a given fault injection experiment. Sliding the window left and right, produces a set of operational profiles with many different stress conditions.

The operational profile selection for the benchmark DFWCS is based upon actual plant data collected over a period of time. At present our profile on the benchmark DFWCS is more representative of a light to heavy workload, a transition from low to high power operation. This mode of operation invokes a substantial amount of the control processing.

Step 5: Creation of Fault free Execution Traces. For each operational profile that is selected, a fault-free execution trace must be created to support the analysis efforts in the safety evaluation process. The fault free trace is considered the correct operation of the system in the presence of no faults. The trace will contain information, such as the sequence of instructions

that are executed during a given period of observation, as well as the process state information that is visible (that is, the information that is accessible by a programmer using the programmer's model view of the processor). The fault free trace is compared to the faulted trace after fault injection to determine if the fault was detected and properly mitigated by the system. The experimental environment that will be used to perform the fault injection experiments is constructed in such a manner as to support the creation of the fault-free execution traces as well. In general, an execution trace requires information that is typically accessible using tools such as data loggers, software debuggers, or in this case, in-circuit emulators (ICE). In the assessment of the DFWCS we use three types of data logging to create fault free traces. The first is a data dump routine that executes on the main and backup computers of the DFWCS. This level of information contains the status of all process variables and set points used in the application control laws. The second type of data collection is an external data logger that tracks selected sensor inputs and actuator commands. The third type of data collection is a assembly level instruction trace stored by the ICE machine.

Step 6: Fault List Construction. One of the main issues when setting up fault injection experiments is how to build the list of faults to be injected. Solutions depend on the goal of the experiments: if they aim at measuring the fault coverage attained through detection mechanisms, the list has to be representative of the whole set of possible faults which can occur in the system. On the other side, if the experiments aim at locating faults which could be critical for the system behavior, special techniques have to be used to target only these faults. In our case, we needed to identify faults that were tied to failure modes of interest, like computer output fails high. These failure modes were derived from the information in Section 2.3, and by analyzing the results of the FMEA presented in Chapter 2.

Our fault injection experiments are shown in Fig. 2.4.7 as a *Set of Experiments*. The set of experiments will not be an exhaustive set due to the size of the fault space (which is assumed to be infinite), but it is assumed that the set of experiments is a representative sample of the fault space to the extent of the fault model and to the confidence interval established by the number of samples each fault injection experiment that is performed, there are three possible outcomes, or events, that can occur as a result of injecting the fault into the system under analysis. First, the fault could be covered or detected. A fault being covered means that the presence and activation of the fault (which produces an error) has been correctly mitigated by the system. The second possible outcome for a given fault injection experiment is that the fault is uncovered or undetected. An uncovered fault is a fault that is present and active as with a covered fault (and thus producing an error). However, the fault detection mechanisms in the I&C system is somehow insufficient and cannot identify the incorrect system behavior. As such, any discrepancies that are produced as a result of the fault are not identified by the onboard fault detection mechanism or diagnostics, and thus the fault is uncovered.

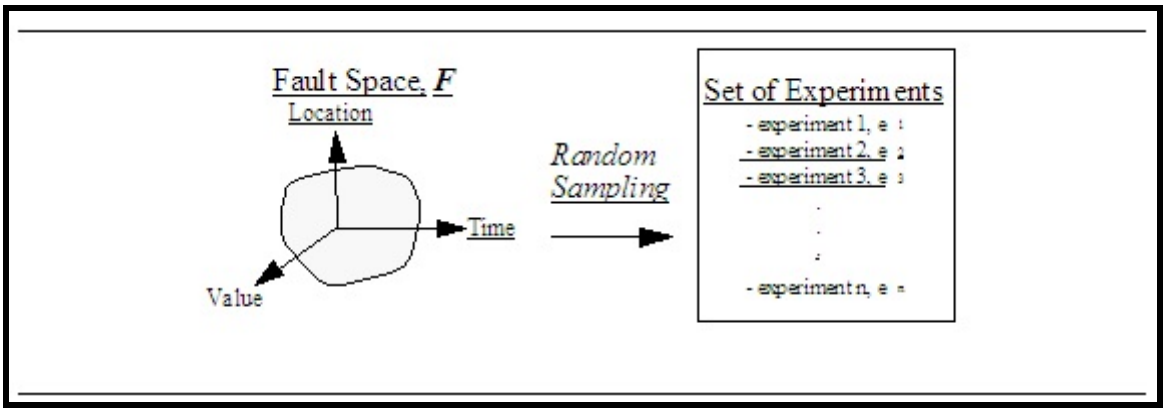

Figure 2.4.7 Generation of the Fault Experiments from the Fault Space

The final possible outcome for a given fault injection experiment is that the fault causes no response from the system. A no-response fault is one that is present, but is not active. For instance, a fault could be present in a given portion of memory that is never accessed by the system. Or, if it is active, it is not producing an error due to the fact that its effect is being masked by the system. For instance, a stuck-at-1 fault could be present on a given signal line whose value is always a logic one. These no-response faults are typically referred to as latent or dormant faults. During a system evaluation using fault injection, it is desirable to reduce or eliminate no-response faults. No-response faults provide no additional information with regard to computing the system coverage. In essence, the results from fault injection experiments for no-response faults can be discarded without affecting the coverage estimate. Note that in order to discard the fault, it must be shown that the fault is truly a no-response fault.

In addition, no-response faults require the maximum amount of time to perform the fault injection experiment. This is due to the fact that the experiment is not terminated prematurely by a fault mitigation mechanism (such as a hardware watchdog timer or a software diagnostic) in response to an active fault. In general, reducing, or ideally eliminating, the number of fault injection experiments involving no-response faults will help to minimize the time and effort needed to perform the fault injection-based evaluation, which in general is a very resource-intensive process.

To this end, several researchers have addressed the problem of no-response faults by constraining the fault space from which the set of experiments is generated by performing some sort of algorithmic processing of the fault space (using the fault-free execution trace that was mentioned above) [51]. By pre-processing the fault list, it is possible to eliminate wasteful no-response faults. The faults that remain after preprocessing the fault list are guaranteed to be response faults. Finally, these faults are used to generate the set of experiments by randomly selecting from the portion of the reduced fault space to ensure that every fault that is injected will produce an error. Thus, no unnecessary fault injection experiments are performed, and the efficiency of the evaluation procedure is improved significantly, even when taking into account the overhead associated with the extra processing required to identify the portion of the fault space that does not contain any no-response faults. In our work with the benchmark DFWCS, the entire list of faults was pre-processed to ensure that they were no-response faults.

Fault Expansion.

As indicated earlier, the concept of fault equivalence and expansion plays a critical role in the fault list generation process. The concept of fault equivalence and it's role in fault injection can best be described by Fig. 2.4.8. First, consider a program running on a computer. For each variable s_i used by the program, and for each write operation on it, we can define a *window period* the longest period starting with the write operation on the variable, ending with a read operation on the same variable, and not including any other write operation apart from the initial one. Referring to Fig. 2.4.8, the number of faults contained in the window period of opportunity is infinite if one considers time as a continuous variable. Digital systems, however, are designed based on the concept of discrete time units. Thus, the number of faults contained in the window of opportunity is measured based on this fundamental discrete unit of time. The discrete unit of time is derived from the minimum time required for the system to reach the next system state. This fundamental time unit is referred to as a system clock cycle. In most cases, this discrete unit of time is the inverse of the processor clock frequency. Some systems, however, update the state of the system faster than the processor clock frequency. Thus, the system clock cycle may or may not be equivalent to the duration of the system processor clock. The number of system cycles contained in the window of opportunity is the number of equivalent faults for this particular fault injection experiment. This quantification assumes that the fault occurrence is an independent event which can occur at any time independent of the system clock. The effects of a fault occurrence event, however, are observed and propagate through the system based on the system clock.

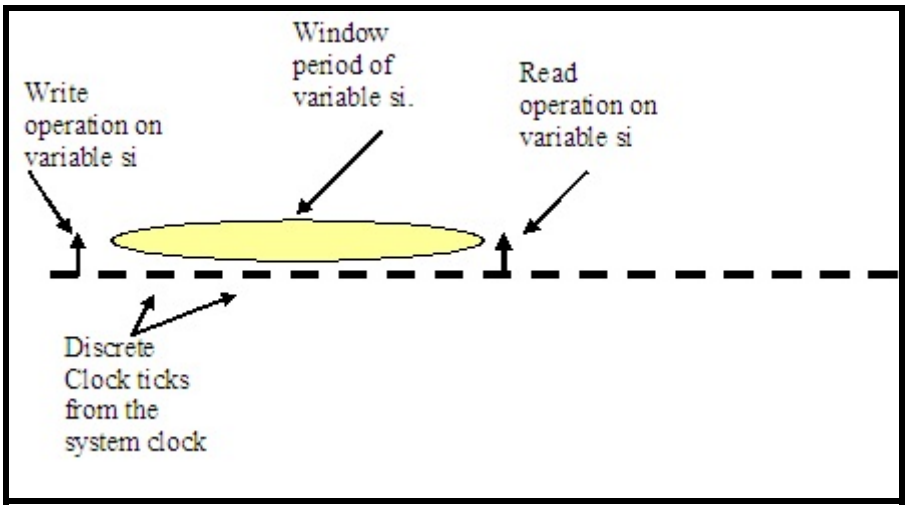

Figure 2.4.8 Fault Equivalence concept.

As example of fault expansion, if there were 1000 clock ticks between the write of variable s_i, and the read of variable s_i, then there would be the possibility of 1000 faults in the window period that would be equivalent. We only have to select one fault from the fault set of 1000 to

count all 1000 faults. This is the power of the expansion concept. Stating this concept mathematically,

$$C_s = [\frac{t_a - t_b}{T_C}] \quad (2.4.10)$$

Where C_s is the number of equivalent faults, t_a is the time instance of the first write operation in the window, and t_b is the last read/write operation in the window and *Tc* is the time period associated with one system clock of the system under test. See reference [121] for complete description of fault expansion and coverage estimation using expanded faults sets.

2.4.5 Experimental Setup: Design, Implementation of the Fault Injection Environment

2.4.5.1 Overview

This section describes the experimental setup used to perform the fault injection experiments for the purpose of evaluating the fault detection mechanisms and responses of the DFWCS. The fault injection tool was developed specifically to support precise controllability of the fault injection parameters given Eq.(2.4.3). That is, fault type, fault locations, fault time, fault value, and fault duration. Furthermore, the environment for designed to support the processor fault models described in Section 2.4.4. Recall that the fault model adopted for this project is a bit flip model. Faults are emulated by injecting single or multiple bit flip error patterns into the memory maps, registers, busses, and I/O registers of the microprocessor of the DFWCS computer modules. There are many different methods and techniques for implementing fault injection [110]. For this work, we developed a unique method for fault injection that employs an ICE as the heart of the fault injection environment. We found that the ICE has the potential to provide very high controllability and observability, approaching that of detailed simulation based fault injection.

An ICE machine is a tool used by designers of embedded systems to debug embedded software. Debugging embedded system software is particularly challenging because embedded systems usually lack suitable user-interface devices such as keyboards and displays. Under such circumstances, ICE machines provide a ‘window’ into the system through which the designer can effectively control and observe an embedded system at a very low level. In-circuit emulators usually have a pod that plugs directly into the socket where a CPU chip is inserted. There is interface circuitry which provides a connection between an ICE machine and a terminal PC. This terminal can be used to run an interactive user interface application using which a designer can monitor the embedded system being designed. For the rest of this discussion, this PC or terminal which runs the user interface application will be referred to as the *host machine*. The machine which the ICE emulates will be referred to as the *target machine*. The execution of the target machine can almost completely be controlled and observed from the host machine. This feature is particularly attractive for using ICE machines for fault injection purposes. A few significant capabilities of most in-circuit emulators when used with an interfacing application are described below.

Memory and register view: When the emulator is halted, the processor state (contents of CPU registers, memory, memory management status registers, bus state, timing details, and status flags) can be viewed in the interface application running on the host machine.

Software execution breakpoints: Breakpoints can be defined anywhere in the code where the designer wants to halt the execution to look at the contents of the processor state.

Hardware breakpoints: This is more of a feature of the processor that is being emulated than of the emulator. These are registers used by the processor for allowing the programmer to enable various debug conditions. They are usually used by storing a memory address in them, which when accessed by an application causes the emulator to halt. These registers can be enabled or disabled from the interface application running on the host machine. For instance, the Intel 486 processor of the DFWCS has 6 debug registers that can store "state" information at time during the execution of the program.

Disassembly: The interface application is capable of disassembling machine code and matching it to source level symbolic information (variable and function names) while stepping through code. This gives the designer a layer of abstraction, giving an effect of stepping through source code as opposed to machine or assembly code.

Trace collection: Execution times of each instruction can be logged in terms of either clock ticks or absolute time (nanoseconds/microseconds).

Script execution: This is a capability that is usually built into the interface application running on the host machine. There is usually a shell prompt where the programmer can give shell commands for various operations such as for halting the processor, restarting it, and so on. A sequence of shell commands could be executed to attain a complex operation. These commands can be saved in a script file and executed at once giving rise to a very useful automation tool.

The main advantage of ICEs for fault injection is the capability to alter processor and memory state easily and view the results of the alteration which is the main motivation behind using them for fault injection experiments. When the processor is halted, the memory values and register contents can be viewed and altered as desired by the user. This ability has been made use of for implementing a fault injection environment. The processor execution halted at a desired time, followed by an alteration to processor state (memory, registers, or status flags), followed by resumption of execution can be considered a fault injection. Given the script execution capability, it is possible to automate injection of a larger number of faults. Accordingly, this capability to alter the process state of a executable program conforms nicely with the generic processor fault model that was described in Section 2.4.4.

Collecting information regarding response of the system after fault injection is to be done at an abstraction level that is far above than that of the processor. This is because we need to observe how the whole *system* responds to a fault. Hence, the ability to collect data at regular intervals after each fault injection is a capability the emulated target machine would need to have. Information available at the level of the in-circuit emulator is very low level and global effects may not be noticeable from the (emulated) processor level. Consider for example a variable in a control system for a heating unit called temperature. If the fault injection reduced its value down to 0 degree Celsius, it could respond in a way that resets some variables in memory. From the view of a processor it may not have any semantic meaning, but to the whole system it might show up as a response which (say) turns on the heating system. Hence it is important that there is capability to know the response of a fault injection that 'reduced the temperature down to 0' (variable/CPU level) gave a response which 'turned on heating' (system/global level).

2.4.5.2 The DFWCS Experimental Test Bed

All experiments were conducted on a duplicate lab version of the *DFWCS* application that was installed at the power plant. Figure 2.4.9 shows the architectural configuration of the DFWCS. It consists of two identical industrial PC based computers called the *main* and *backup* machines (e.g. CPU's). They run Intel 80486 DX4 processors at 120 MHz. These machines are equipped with plasma display units (*PDU*s) which act as status display monitors. They are touch screens which display status information about the DFWCS application and can be used to give commands to it through an interactive menu it displays. The main and backup machines are connected to an interface backplane. The controllers are devices that take as their input the position commands from the main and backup CPU's and produce outputs to the main feed regulator valves and bypass valves.

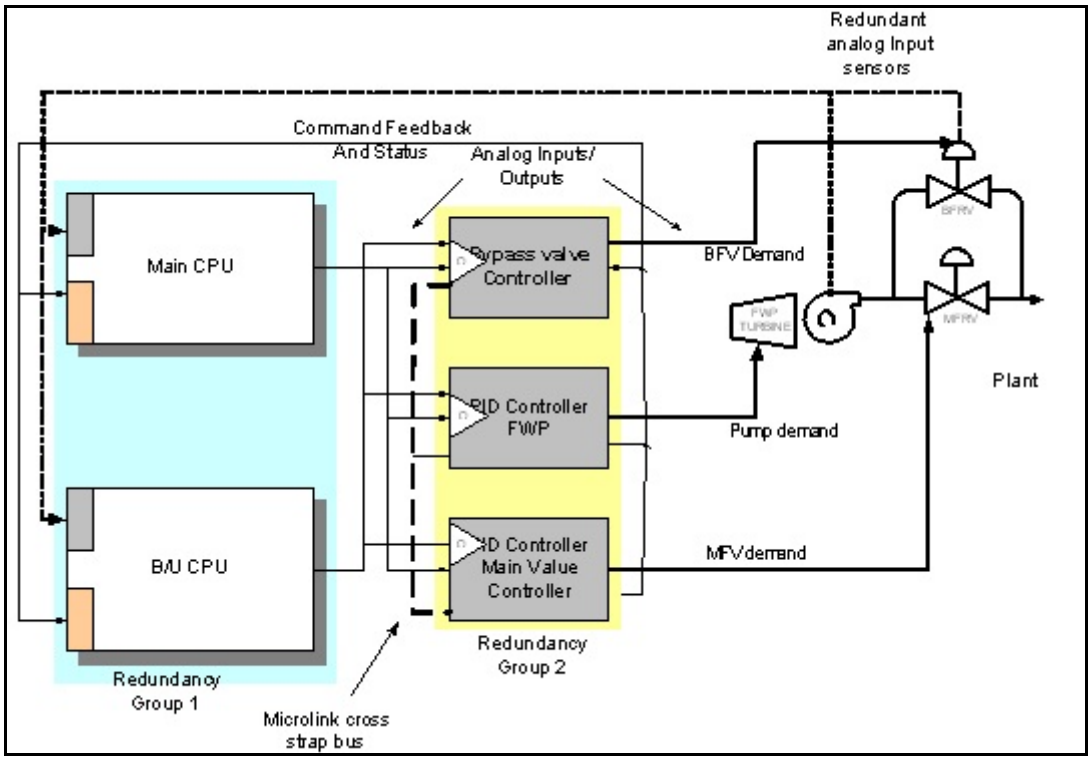

Figure 2.4.9 Architectural View of the Benchmark DFWCS.

The DFWCS needs an elaborate reset sequence for startup, only after which the system is available for use and for fault injection. This reset sequence is strictly time constrained, and is handled by another terminal which is called the *sequencer*. This terminal gives ASCII commands to the CPU at the specific times defined by the reset sequence. The CPU transforms them into plant parameters and connects to various elements of the system through the back plane.

The Microtek Powerpack In-Circuit Emulator EA-486 was the emulator used for fault injection experiments. It is an emulator for Intel 80486 processors. Its main component is a plug that can be inserted into the pin grid array (PGA) socket of the 486. This is connected to interface circuitry which can be connected to a terminal host over the serial port. The host machine is a Windows 2000 machine running on an Intel Pentium III processor. This runs the interface application and is referred to as the source level debugger (*SLD*).

In all experiments that were conducted, the main CPU was used as the target. The ICE machine was plugged into the CPU socket of the main machine. This was interfaced to the host machine through the serial port COM 2 in the main CPU. A trigger line was connected between the sequencer and the ICE machine. This is the line used by the sequencer to inform the emulator that the system is ready for fault injection. Fig. 2.4.10 below summarizes the integration of the emulator into the DFWCS environment.

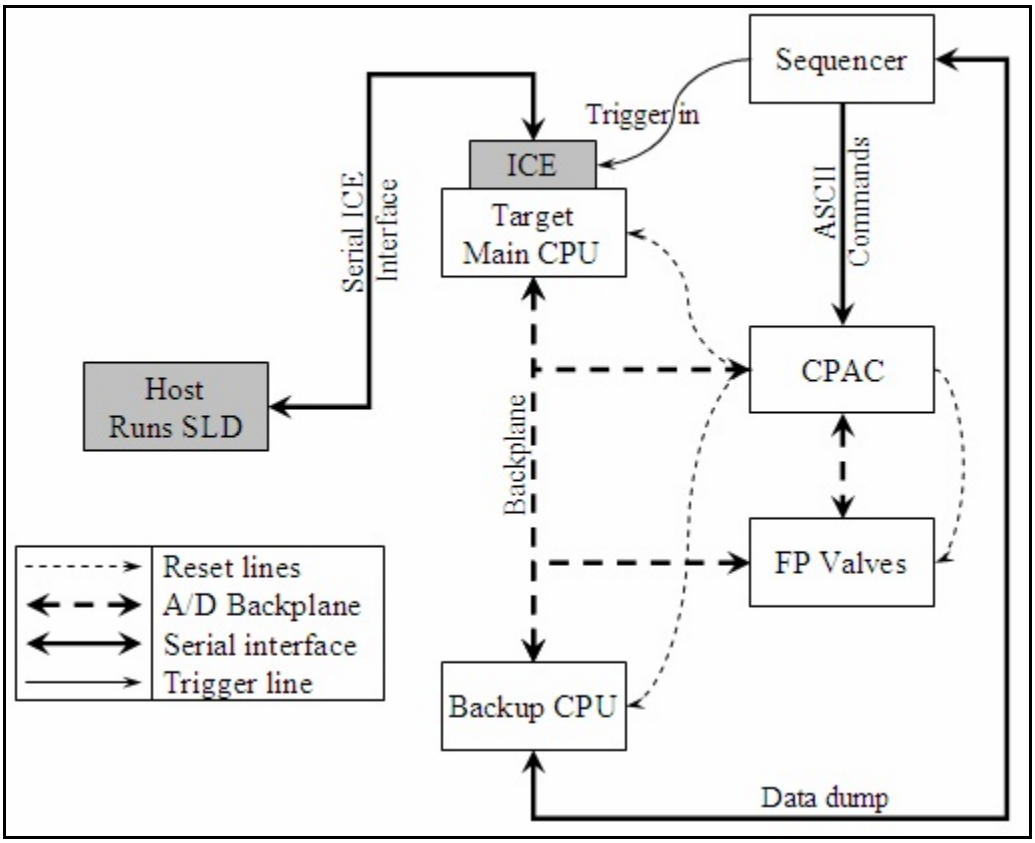

Figure 2.4.10 Integration of the In-circuit Emulator into DFWCS Lab

2.4.5.3 Identifying Potential Fault Injections Locations and Values for the DFWCS Application

The approach we take for the first set of fault injection campaigns is to replicate as best as possible the conditions of failure modes given by the models in Section 2.3, and listed Table 2.3.1 for the main computer. We applied the malicious fault list generation method [121] to select faults that would have specific impact on the operation of the DFWCS. The application source code was manually searched for identifying variables that could be potential candidates for fault injection according to the malicious fault list generation criteria. Namely, variables in modules that corresponded to input processing, output processing, control law, and acceptance tests were the ones mainly chosen. A compiler generated memory map was then used to find out memory addresses of each of these variables. This had to be done manually because the SLD was not capable of matching symbolic information for 16 bit applications. This process gave a list of memory addresses used by the application for which a corruption could bring about significant consequences. A fault list was compiled based on the address of each

variable together with a masking bit pattern for the corruption. The fault types that were created for the list typically include:

Fault types:

- Force the contents of a memory location(s) or CPU register(s) where critical program variables reside to a all zero state = 0000h
- Force the contents of a memory location(s) or CPU register(s) where critical program variables reside to a all one's state = FFFFh
- Force the contents of a memory location(s) or CPU register(s) where critical program variables reside to a random assignment state = random bit vector assignment
- Force the contents of a memory location(s) or CPU register(s) where critical program variables reside to invert state = Corrupted Contents Inverted from original contents.
- PC and Stack registers corrupted.
- Memory management registers corrupted to any value desired.

Fault Duration:

The different fault types listed above can be activated for different periods of time. The smallest period of time is 2 instruction cycles (~300 ns). The largest period of time (tested to date) is about 5 seconds.

Fault location:

Any of the fault types listed above can be activated at any writable location in the state of the processor. This includes process memory, code segments, data segments, registers, status registers, heap memory, OS code and data segments, process state locations, etc.

Fault Injection Timing:

Since the ICE machine can halt on any instruction boundary, data or variable reference the timing of the fault placement can be very precise. The time granularity is measured in either clocks cycles or instruction cycles, both of which are in the nanosecond time scale.

2.4.5.4 Fault Injection Automation

A small C application was written to generate a script containing SLD commands based on the fault list. This script runs on the SLD application, and handles the automation of fault injections. The sequencer machine goes through the elaborate reset sequence of the various components of the system, and signals the ICE machine when the system is ready for a fault injection. This is accomplished by sending a trigger to the trigger in pin of the ICE machine. This event is

detected by the SLD script that is running on the host machine. The SLD script has been configured to halt the target machine and prepare for injection of a fault from the fault list. A fault in the fault list contains information such as the segment and offset address of the location to be corrupted, number of bytes to be corrupted and a masking bit pattern to be logically applied to the contents of the memory location. These pieces of information are used to define an *event* on the host machine that causes the emulator to halt when a write occurs at the memory location defined by the event. After this, execution of the target is resumed which causes the main CPU to execute normally. This eventually causes an event of 'write to the memory location' to occur at some point of time in the near future. This implies that the code has written (possibly) a new value to the location.

2.4.5.5 Data Collection

Figure 2.4.10 shows a data dump interconnection (serial link) between the sequencer and the backup machine. The sequencer sends a command to request a data dump to the backup machine, and it responds by sending the data dump back on the same serial connection. This functionality is exercised once before the fault injection and once after the fault injection. It has been configured so that each data dump is stored as a separate file with filenames defined by time stamps. The data dump also contains information regarding the fault that was injected. This piece of information is extracted from the fault list and inserted into the file that is logged. Discrepancies between the pre fault injection data dump and post fault injection data dump can be analyzed for generating statistical data after a large amount of data has been collected.

2.4.6 Results From a Fault Injection Campaign

Fault injection campaigns were performed on the DFWCS to ascertain the effectiveness of the fault tolerance mechanisms, classify the error responses of the systems to injected faults, and in general exercise the methodology presented to find its strengths and weaknesses. Referring to Fig. 2.4.10, the process of fault injection begins with the system being reset so that the experiment starts from a "good" state. Once the DFWCS main processor is running in known good state, we collect operational data from the DFWCS for the fault free run. This data is used for comparison against the operational data collected during fault injection trial. Once the faulted data is collected and stored by the sequencer computer in Fig. 2.4.10 the DFWCS is reset again to start a new fault injection trail.

Once the response was logged and found to be a detected fault. We can compute the fault equivalence coverage of the detected fault. For this fault the time between the first write to the I/O memory register for the variable FW flow and the last read from the register containing the variable FW flow was 5720 system clock cycles. Thus, the expansion factor of this fault is 5720.

2.4.6.1 Error Classification

The fault injection environment along with the data acquisition software is able to observe a wide variety of errors. We currently classify the errors into four categories:

1. **Detected errors**: Errors that are detected by the DFWCS fault tolerance mechanisms and correct runtime recovery was initiated.
2. **Other errors**: Errors that were detected by the DFWCS, but caused the processor to hang or crash. Recovery was not possible until the processor was rebooted.
3. **Sustained Undetected errors or uncovered errors**: Errors that were not detected by the fault tolerance mechanisms causing a failure to occur, that is the DFWCS produces an incorrect control signal.
4. **Spurious undetected errors**: Errors that temporarily cause the system outputs to deviate from expected performance, but the system eventually stabilizes, or detects the error and corrects it.

2.4.6.2 Common Mode Failures

A possible approach to quantify the likelihood of CMFs from an integrated hardware/software perspective is described in [122]. This modeling scheme represents the system in the information universe to obviate the need to distinguish between hardware and software when developing the system model. One key contribution of this research was to recognize that dependable embedded digital systems often employ error containment regions (ECR's) for fault tolerance. An ECR is defined as a collection of data produced by either hardware or software, which is unaffected by any arbitrary error outside that given region. Conversely, errors that originate outside a given ECR cannot affect that ECR. Two existing modeling methodologies, the DFM [3] and the data flow [122] were used to represent and to analyze the system's information universe. The DFM derives the test vectors, the location for error injection and the partitioning of the ECR's. Data flow provides the simulation environment for the model that determines the effect that the various errors have on the system and provides the data to derive the coverage estimate for CMF.

The basic idea for quantification is to use multiple fault injections across a span of ECR's in an embedded systems. The method of using fault injection for analysis CMF coverage is an extension of the basic single point fault injection methods described earlier. The CMF fault injection includes injecting faults into multiple redundant channels at key times of execution, or multiple faults into a single channel at key times of execution. Deriving the list of potential CMF requires analysis of the hardware/software system, specifically the interactions between the various components inside a single channel, and across different channels. The methods used for the estimation of coverage for CMF are no different than for single mode fault injection. The one requirement that is crucial is the need for precise timing measurement which requires the establishment of a global clock on an operational platform. Establishing such a global clock is

not a difficult task since most fault injection environments have a precision clock that can be used as a global clock.

2.4.7 Estimation of Failure Mode Rates and Failure Mode Probabilities on Demand

The failure rate data for the DFWCS listed in Table 2.4.1 are for the failure rates of the DFWCS components irrespective of their failure modes. The failure data also do not take into account the fault detection, detection or or remediation capability of digital systems. In that respect, the data in Table 2.4.1 or data hardware data from existing data bases (e.g. [118]) cannot be used directly either by the Markov/CCMT method or the DFM. A possible way to estimate the needed data for these methods would involve the following steps:

1. *Construct a list of faults that are known to cause the system*. For example, suppose we want to estimate λ_{12} in Table 2.4.1 as it relates to the specific DFWCS under consideration. Then, we would construct a list of faults that are known to cause the system to lose input. This process is called malicious fault list generation [121]. The malicious fault list should include faults that the system is expected to detect which may include, permanent, transient, and common mode failures.
2. *Inject each fault on the malicious fault list into the system and observe the system response to the fault.* The ratio of the detected faults to the total fault injection set defines the coverage C_{fm} for device f in the failure mode f_m (see Eq.(2.4.2)). The injection set will include injections at different times t, with different durations Δ, locations l, and internal and external system states σ. The uncoverage ($1\text{-}C_{fm}$) is then the probability that the system will not detect the fault m in device f so that the fault will propagate though the system and $\lambda_{fm}(1\text{-}C_{fm})$ is the failure rate of device f in the mode m. Using the example of Step 1 above, λ_{12} ($1\text{-}C_{12}$) will be the failure rate for loss of input. For digital systems which are activated/de-activated upon demand$1\text{-}C_{fm}$ will be the probability of failure upon demand.
3. Repeat Steps 1 and 2 for all the failure modes f_m.

Step 2 is performed under the assumption that all the injection set parameters are equally likely to occur. Since is the failure rate is irrespective of the failure mode, the frequency of occurrence of any one failure mode from the set of all possible failure modes f_m is less than total failure rate of the device f, and subsequently the approach is conservative.

Another possible approach is based on a new metric called Mean Time To Unsafe Failure (MTTUF) [123]. This metric and it's mathematics were developed in response to the concern about using failure rate data when there is sparse component failure data to make a credible basis for predicting overall system failure. The MTTUF metric represents the average or mean time that a system or component will operate safely before a failure occurs, and produces an unsafe or undetected system state. The MTTUF represents the evaluation of safety or coverage of component as a function of time in the limiting case as time approaches infinity. Thus, it is a steady state metric. The use of the metric in this case would be to estimate the rate

of undetected failure from a operational state to a failure mode state. This approach would require knowing the Mean Time To Failure (MTTF) of a specific failure mode of the system. MTTF information is more likely to be available for the system and it's components rather than failure rate information. However, obtaining MTTF information on specific failure modes would present a challenge.

2.4.8 Initial Conclusions

This section has summarized the quantitative safety assessment process, and more specifically the key steps in the numerical quantification process - the development of analytical models by which safety and reliability can calculated from, the acquisition and generation of data for the key parameters in the models to be introduced in latter sections. The application of this methodology to an actual I&C system (e.g. the benchmark DFWCS) has shown the viability of the approach. Significant findings to this point are summarized below:

1. Careful attention to model parameter uncertainty is required, especially with parameters where data is scarce, such as new device failure rates. The use of modern uncertainty analysis techniques could help to quantify the uncertainty in relation to a model's predictive capability (Section 2.4.5).
2. Data acquisition from the target system should be designed to collect data to support model parameters at different lvels of information. This ensures that subtle failures, whose effects might not be visible at one level of data collection, may be visible at another level of information flow (Section 2.4.7).
3. The development and design of the generic processor fault model was found to be compliant with the DFWCS CPU architecture (e.g. the Intel 486 and the AMD 586) thus the generic processor fault model was representative of the behavioral fault model we used for fault injection on the DFWCS (Section 2.4.4).
4. The operational profile model developed for fault injection was compliant with the operational profile used in the DFWCS fault injection campaigns (Section 2.4.4).
5. Failure data needed for the DFM and Markov/CCMT method can be estimated by fault injection experiments (Sections 2.4.6.2 and 2.4.7)
6. Statistical models can be developed to estimate with narrow confidence intervals the key coverage parameters of specific failure modes (Section 2.4.4)
7. Malicious fault lists can be generated to include a comprehensive list of faults that cause a specific failure mode to occur on the target system (Section 2.4.4)
8. Fault equivalence models can be applied to capture a large set of equivalent faults, reducing the variance of the estimate of fault coverage
9. Novel fault injection techniques can use commercially available test equipment to emulates both transient and permeant faults (Section 2.4.6)
10. Failure mode specific faults can be injected into the target system to successfully collect key data.

2.5 An Example Initiating Event For Illustration

The PRA models that are used in this study (see Section 5.2) are taken from NUREG-1150 [124]. Since these models are directed to Level 2 PRA, they assume that the reactor is shutdown in all the initiating events. In that respect, the following initiating event is used to illustrate how the reliability models constructed with the DFM (Section 3) and the Markov methodology (Section 4) can be incorporated into an existing PRA:

1. Turbine trips
2. Reactor is shutdown
3. Power *P* is generated from the decay heat
4. Reactor power and steam flow rate reduce to 6.6% of 3000 MW_{th} (or 1500 MW_{th}/SG) 1 second after reactor shutdown
5. Feedwater flow is at nominal level
6. Off-site power is available
7. Main computer is failed.

Assumption 1 represents the initiating event. Assumptions 2 and 3 are necessitated by the available PRA models in NUREG-1150. Assumption 4 implies that a 3000 MW_{th} plant is being considered that has operated at this power level for more than 1 year (within 3% of actual values using shutdown heat generation relations given in [125]) and Assumption 5 implies that the reactor was at full power at the time of shutdown. Assumption 6 allows modeling of the actuation (or failure upon demand) of motor operated valves. Assumption 7 reduces the system state space for reliability model construction to provide clarity in illustrations.

Since we are in the low power mode following the plant trip, the BFV is being utilized. Then from Section 2.2.2, we have the following equations as the control laws for SG*n* (*n* = 1,2):

Level: $$\frac{dx_n}{dt} = A(f_{wn} - f_{sn}) \tag{2.5.1}$$

Level error : $$\tau_5 \frac{dE_{Ln}}{dt} = r_n - C_{Ln}(t) \tag{2.5.2}$$

Compensated water

level : $$\tau_2 \frac{dC_{Ln}}{dt} = -C_{Ln}(t) + x_n(t) + \tau_1 A(f_{wn} - f_{sn}) \quad (2.5.3)$$

Compensated power: $$C_{pn}(t) = p(0)e^{-t/\tau_4} + \frac{(1+\tau_3)}{\tau_4}\int_0^t du\ p(t-u)e^{-u/\tau_4}$$

BFV demand $$\sigma_{Bn}(t) = \mu_{Bn}\alpha_{Bn} + \mu_{Bn}C_{pn}(t) + \beta_{Bn}(h_{wn})E_{Ln}(t) \quad (2.5.5)$$

BFV position(%) $$\tilde{S}_{Bn} = \begin{cases} \sigma_{Bn} & \text{main or backup CPU up} \\ \eta_{Bn} & \text{both main and backup CPU down} \end{cases}$$

In Eq.(2.5.1), f_{wn} = 0 if BFV is failed closed. Otherwise, f_{wn} is obtained from the solution of

$$\frac{4.73\mathrm{L}(100/\tilde{S}_{Bi})^2 f_{wn}^{1.852}}{\mathrm{C}^{1.852}\mathrm{D}^{4.87}} = 136 + 6.3\times10^{-6} f_{wn} - 4.6\times10^{-11} f_{wn}^2 \quad (2.5.7)$$

where D is the diameter of inlet pipe to the BFV (in feet) and f_{wn} is in ft^3/s. The L is a fitting parameter. Equation (2.5.7) uses the pump and valve models given in NUREG/CR-6465 [4] and assumes that pump head is equal to the head loss in the valve. In general, the steam flowrate f_{sw} is obtained from the solution of Eqs. (A.29) through (A.41a) in Appendix A along with Eqs. (2.2.14) and (2.2.15). For the example initiating event, it is assumed that steam generation rate f_{sn} follows the primary system decay heat generation rate, i.e.

$$f_{sn}(t) = 0.066\times 21028\left[\frac{1}{(10+t)^{0.2}} - \frac{1}{(3.15\times10^7 + t)^{0.2}}\right]$$

Equation (2.5.8) is taken from [125] and time t is in seconds. In addition to Assumption 1 through 7 above, Eq. (2.5.8) assumes that: $f_{sn}(0)$ = 2102.8 ft^3/s, the reactor has operated for 1 year and stating point of the analysis is 10 seconds after the turbine trip. Table 2.5.1 shows the data used for this example initiating event.

For the data in Table 2.5.1, Eqs. (2.5.1) through (2.5.8) become

$$\frac{dx_n}{dt} = \frac{f_{wn}(\tilde{S}_{Bn})}{109} - \frac{0.066\times 21028}{109}\left[\frac{1}{(10+t)^{0.2}} - \frac{1}{(3.15\times 10^7 + t)^{0.2}}\right] \tag{2.5.9}$$

$$\tau_5 \frac{dE_{Ln}}{dt} = -C_{Ln}(t)$$

$$\tau_2 \frac{dC_{Ln}}{dt} = -C_{Ln}(t) + x_n(t) + \tau_1\left\{\begin{array}{l} f_{wn}(\tilde{S}_{Bn}) - \\ \quad 0.066\times 2102.8\tau_1\left[\frac{1}{(10+t)^{0.2}} - \frac{1}{(3.15\times 10^7 + t)^{0.2}}\right] \end{array}\right\}$$

$$C_{pn}(t) = 0.066\times 1500 e^{-t/\tau_4} + 0.066\times 1500\frac{(1+\tau_3)}{\tau_4}\int_0^t du\left[\frac{1}{(10+u)^{0.2}} - \frac{1}{(3.15\times 10^7 + u)^{0.2}}\right] e^{-(t-u)/\tau_4} \tag{2.5.12}$$

$$\sigma_{Bn}(t) = 1/15 * C_{pn}(t) + 12/54 * 100 E_{Ln}(t)$$

$$\frac{146.53 f_{wn}^{1.852}}{\tilde{S}_{Bn}^{\ 2}} = 136 + 6.3\times 10^{-6} f_{wn} - 4.6\times 10^{-11} f_{wn}^2 \tag{2.5.14}$$

Table 2.5.1 Data Used for the Example Initiating Event

Variable	Value
$f_s(0)$	0.066*2102.8 ft^3/s
$x(0)$	0 ft
$E_{Ln}(0)$	0 ft
$C_{Ln}(0)$	0 ft
$C_{pn}(0)$	0.066*1500 MW_{th}
$P(0)$	0.066*1500 MW_{th}
S_{Bn}	0 %
L/D	2
C	140

D	0.5 ft
μ_{Bn}	1/15
α_{Bn}	0
r_n	0
β_{Bn}	100X12/54
A	1/109.0 ft^{-2}

Figure 2.5.1 shows that the solution of f_{wn} as a function of $\tilde{S}_{Bn}$ is practically indistinguishable from its quadratic function representation through

$$f_{wn} = 0.0014\tilde{S}_{Bn}^2 + 1.2681\tilde{S}_{Bn} - 1.2019$$

.

Also, since

$$\frac{1}{(10+t)^{0.2}} >> \frac{1}{(3.15\times 10^7 + t)^{0.2}}$$

we have

$$\frac{dx_n}{dt} = \frac{0.0014\tilde{S}_{Bn}^2 + 1.2681\tilde{S}_{Bn} - 1.2019}{109} - \frac{0.066\times 2102.8}{109}\frac{1}{(10+t)^{0.2}} \qquad (2.5.17)$$

$$\tau_2\frac{dC_{Ln}}{dt} = -C_{Ln} + x_n(t) + \tau_1\left(\frac{0.0014\tilde{S}_{Bn}^2 + 1.2681\tilde{S}_{Bn} - 1.2019}{-\frac{0.066*3505.1}{(10+t)^{0.2}}}\right)$$

$$-\tau_5\frac{dE_{Ln}}{dt} = C_{Ln} \qquad (2.5.$$

$$\sigma_{Bn}(t) = 1/15\left[0.066*1500e^{-t/\tau_4} + 0.066*1500\frac{(1+\tau_3)}{\tau_4}\int_0^t du\,\frac{e^{-(t-u)/\tau_4}}{(10+u)^{0.2}}\right] + 1200E_{Ln}(t)/54$$

$$\tilde{S}_{Bn} = \begin{cases}\sigma_{Bn} & \text{main or backup CPU up}\\ \eta_{Bn} & \text{both main and backup CPU down}\end{cases}$$

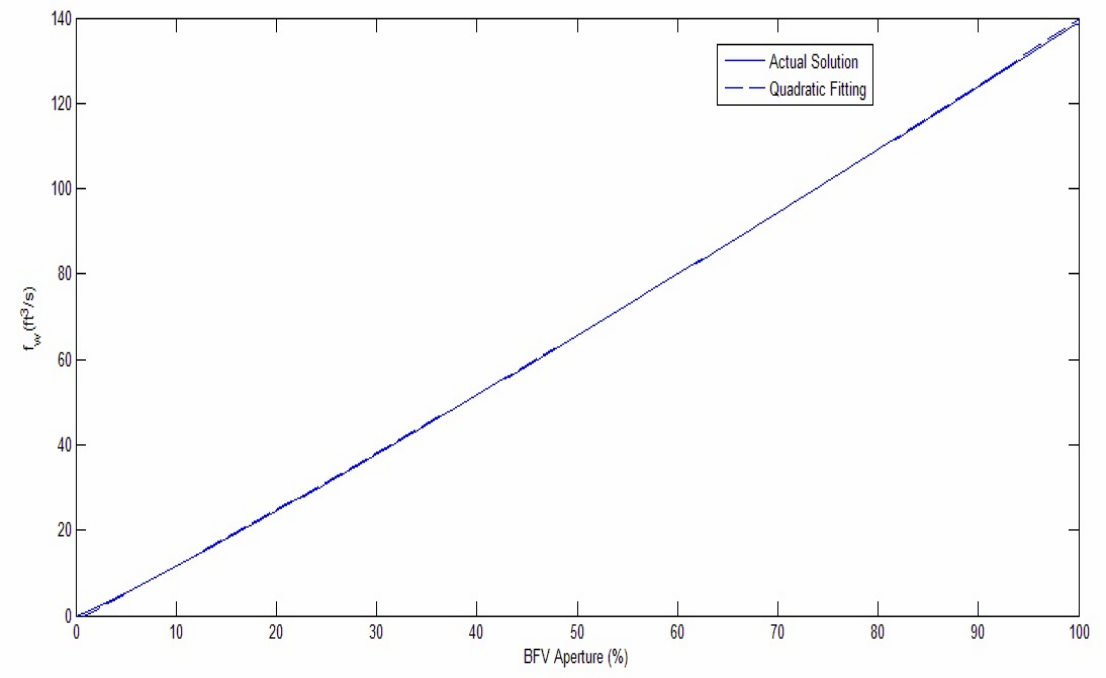

Figure 2.5.1 The Solution of f_{wn} from Eq.(2.5.15) as a Function of $\tilde{S}_{Bn}$

Laplace transform of Eqs. (2.5.17)-(2.5.20) following linearization around normal operating conditions shows that the transfer function has one real (Root 1) and two complex conjugate roots (Root 2 and Root 3). Figures 2.5.2 and 2.5.3 show that the system is unconditionally stable for $10 \leq \tau_1 \leq 100$ s and $10 \leq \tau_2 \leq 100$ s.

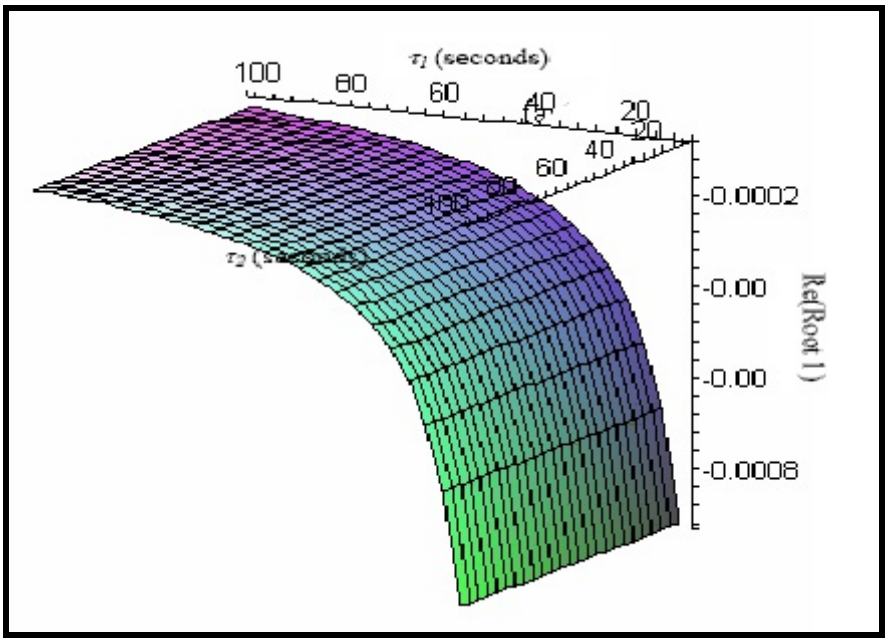

Figure 2.5.2 Real Part of Root 1 of the Transfer function of Eqs.(2.5.17), (2.5.18) and (2.5.19) Following Linearization Around E_{Ln} = 0

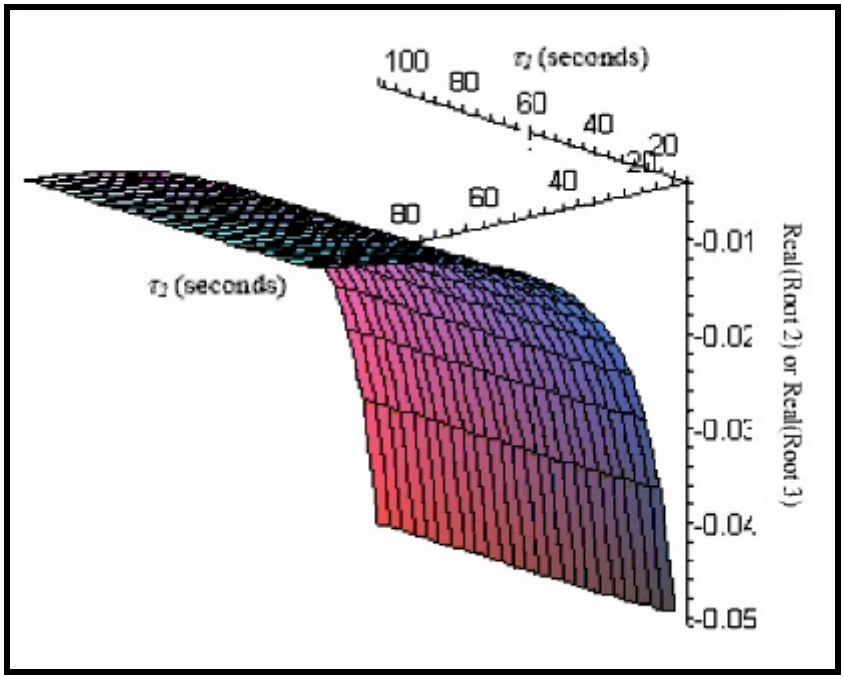

Figure 2.5.3 Real Part of Root 2 or Root 3 of the Transfer Function of Eqs.(2.15.17), (2.15.18) and (2.15.19) Following Linearization Around E_{Ln}=0

Figures 2.5.4 - 2.5.6 show the behavior of Eqs. (2.5.17) - (2.5.20) for τ_i=10 seconds (i=1,...,5), $x_n(0) = C_{Ln}(0) = E_{Ln}(0) = 0$ and $\tilde{S}_{Bn}$ =100. Both level x_n and compensated level C_{Ln} stabilize around their nominal value within 100 seconds following the initiating event, while level error E_{Ln} shows a steady decrease after 100 seconds. This behavior is consistent with Eq. (2.5.10) which determines $E_{Ln}(t)$ from the difference between the setpoint r_n and $C_{Ln}(t)$ (see Eq. (2.5.2)). The compensated level $C_{Ln}(t)$ anticipates the behavior of the difference between steam outflow and feedwater inflow into the SG (see Eq. (2.5.11). Since steam outflow follows the power generated in the primary system and power decreases with time (see Eq. (2.5.8)) so does the difference between the actual level x_n and compensated level $C_{Ln}(t)$.

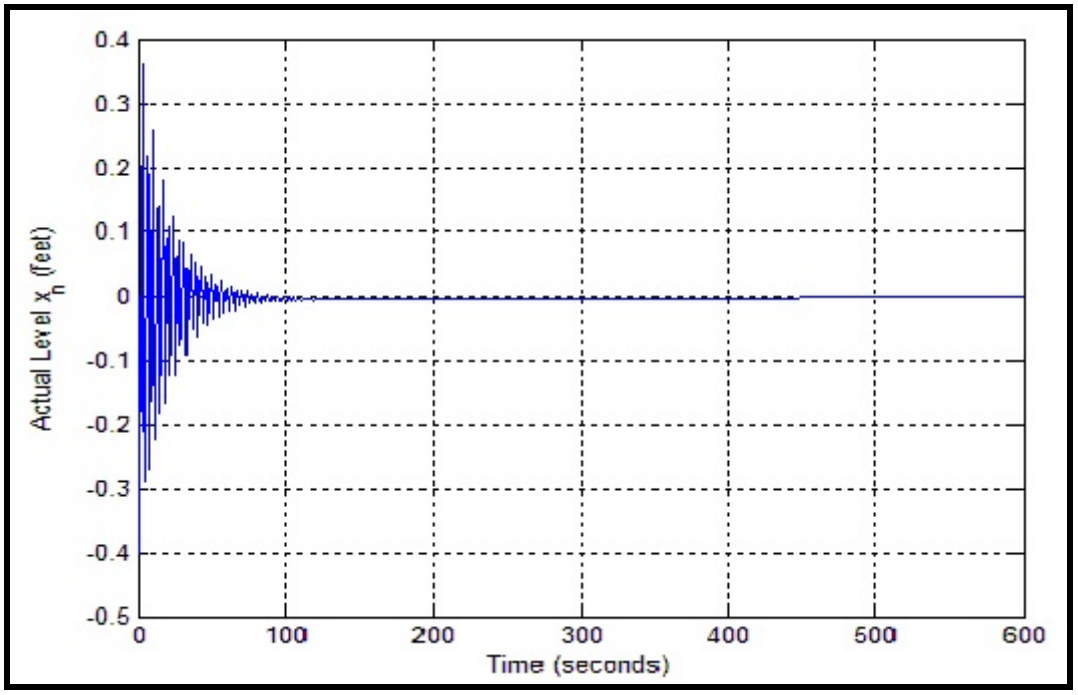

Figure 2.5.4 Variation of Actual level with Time for the Example Initiating Event

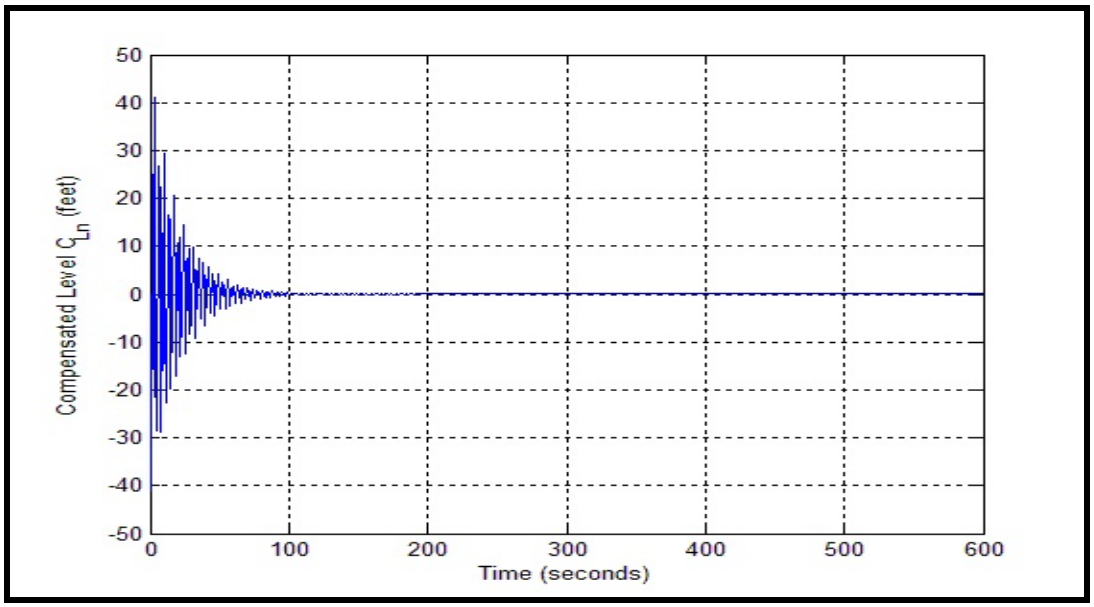

Figure 2.5.5 Variation of Compensated Level with Time for the Example Initiating Event

The anticipation of the behavior of the difference between steam outflow and feedwater inflow into the SG is also the reason why the magnitude of variation of $C_{Ln}(t)$ is different from $x_n(t)$ initially. The difference decreases as f_{wn} approaches f_{sn} and subsequently both $C_{Ln}(t)$ and $x_n(t)$ approach their target value of zero.

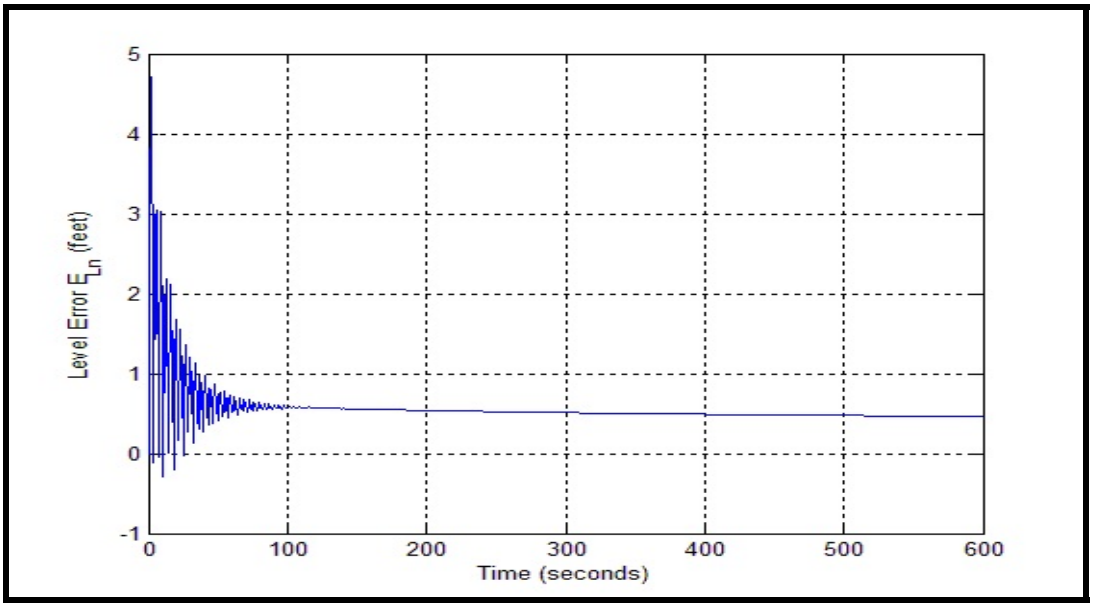

Figure 2.5.6 Variation of Level Error with Time For the Example Initiating Event

Figure 2.5.7 shows that the exact timing of the failure of a system component can have an impact on the resulting system failure. In particular, Fig. 2.5.7 depicts the evolution of the level variable under two distinct scenarios starting both from the same initial conditions as those in Figure 2.5.4. In one case, the BFV fails stuck at the current position at time t = 43 sec. In the other case, the BFV fails stuck at time t = 44 sec. The first scenario results in the level failing low (x_n < - 2.0 feet), while the second scenario results in the level failing high (x_n > 2.5 feet). This example is important because, for a system similar to the digital feedwater control system in an operating PWR, it illustrates:

- what has been reported in the literature on the possible sensitivity of the system failure mode to the exact timing of component failures [126], and,
- that an analysis that considers only the order of events and ignores their exact timing may result in the failure to identify possible failure modes.

It is also important to note that while such sensitivity of the system failure mode to the exact timing of component failures may be also true for the traditional analog I&C systems and is usually not considered under the current ET/FT approach to PRA, it is not clear that this approach is still allowable under the discrete time nature of digital I&C systems. For example, the position of the level and the magnitude of the other variables used by the controller (e.g. compensated level, level error) at a sampling point in time are no longer the same at the next sampling point. The DFM and the Markov/CCMT methodology both account for such uncertainties by representing the system dynamics in terms of mappings between process variable intervals in discrete time (see Chapters 3 and 4, respectively).

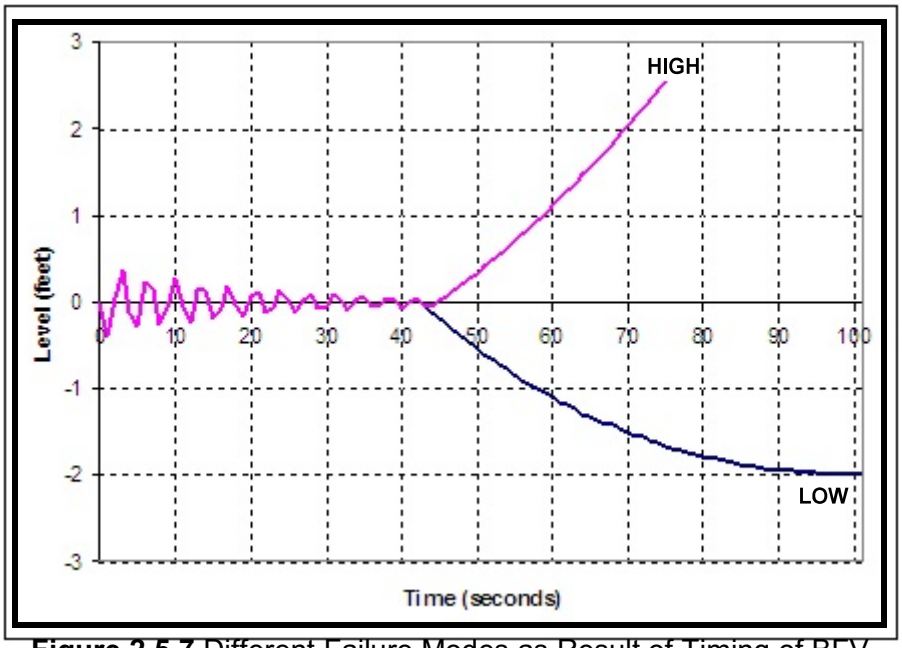

Figure 2.5.7 Different Failure Modes as Result of Timing of BFV Failure

Figures 2.5.8 - 2.5.10 present another interesting issue. These figures display the same data shown in Figures 2.5.4 - 2.5.6 except that they include a longer time interval (t = 0..1200 seconds). The system seems to exhibit instability around time t = 880 seconds where the three variables start oscillating again. The level and the compensated level quickly settle again around their nominal value, and the level error seems to make a jump before resuming its slow descent. It is important to note that while, in principle, such a dependence of the nature of the Top Event on the timing of failure could be found by a careful FMEA, the FMEA would amount to going though the steps to generate the cell-to-cell-transition probabilities of Markov/CCMT methodology (Section 4.2.4) or the decision tables of DFM (Section 3.1.1) since it would still require considering possible locations of the system in the discretized state-space of Eqs. (2.5.1) - (2.5.7) with respect to the example initiating event transitions (Section 2.5.1).

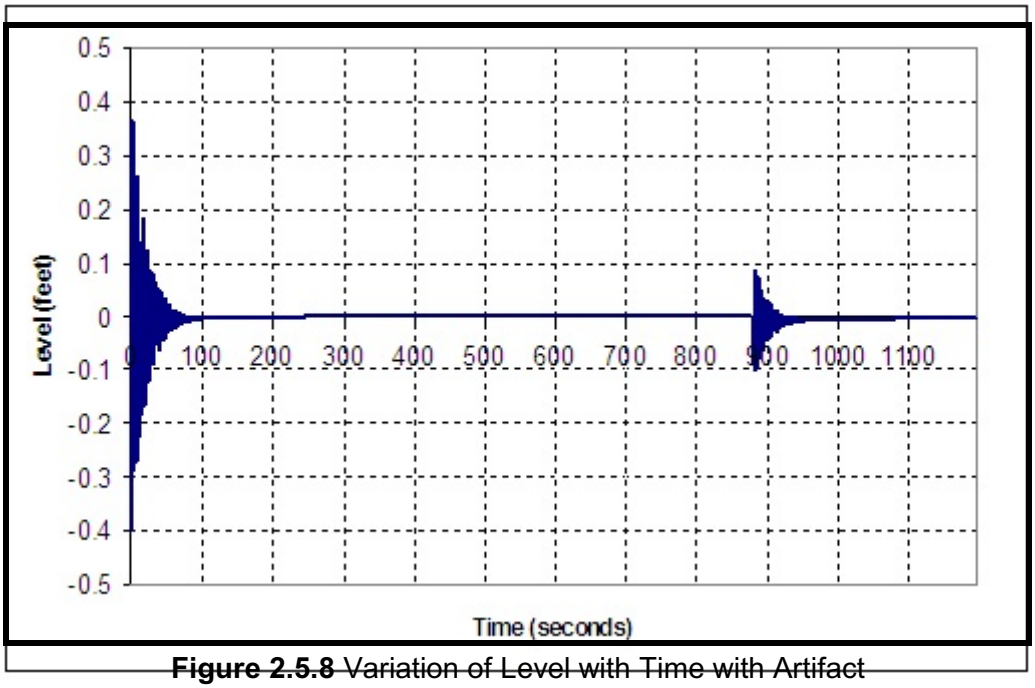

Figure 2.5.8 Variation of Level with Time with Artifact

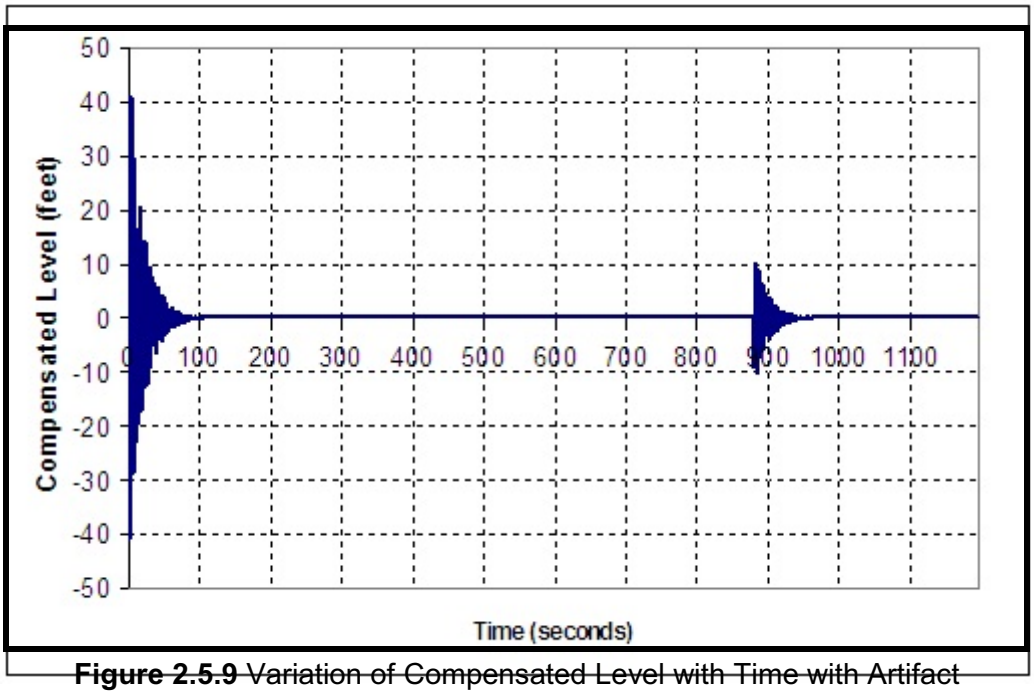

Figure 2.5.9 Variation of Compensated Level with Time with Artifact

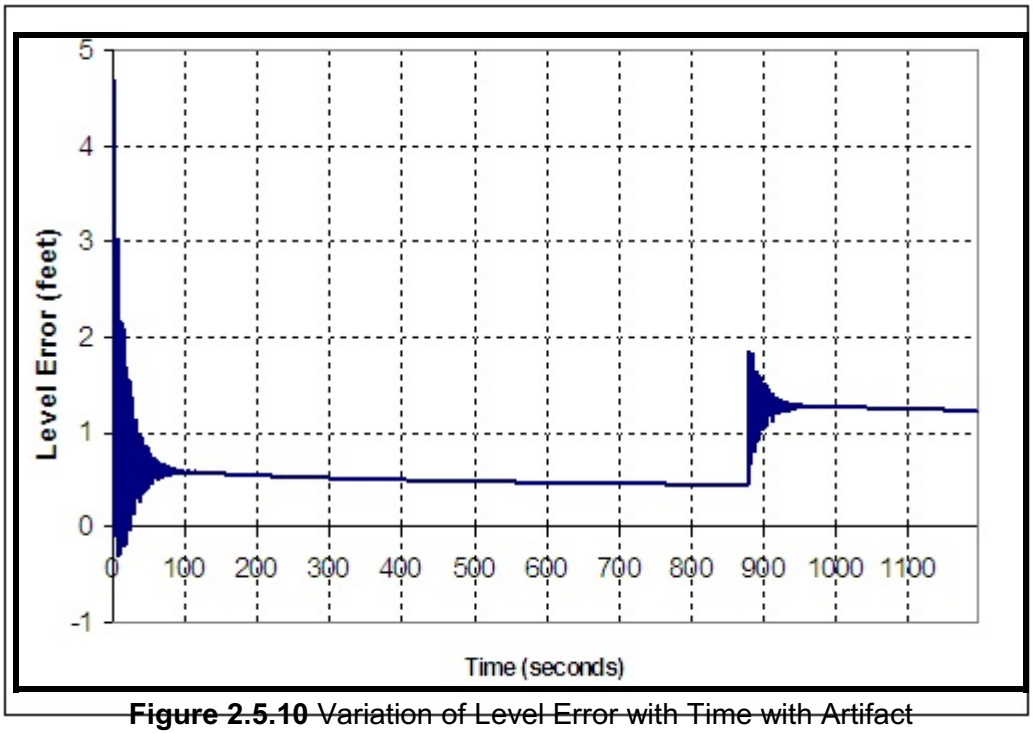

Figure 2.5.10 Variation of Level Error with Time with Artifact

This behavior may be caused by an actual instability in the system and its corresponding model. Such instabilities have been observed in nuclear plants[7]. However, in this case, it is an artifact that is the result of a numerical error in the digital control algorithm simulator. The algorithm uses Gauss-Legendre quadrature to evaluate the integral in Eq. (2.5.20). The integral is computed repeatedly with an increasing number of points until the absolute value of the difference between two consecutive estimates of the integral is below a given threshold (10^{-6}). At time t = 880 seconds, the first two estimates of the integral are both below the threshold itself, so that the absolute value of their difference is also below the threshold. This causes the algorithm to stop its iteration and return the wrong value for the integral. Figures 2.5.11 and 2.5.12, respectively, show the correct integral in the range $0 \leq t \leq 1200$ seconds, and the integral calculated by the faulty algorithm in the same time interval. The numerical problem presented here would probably be avoided by an experienced, qualified programmer. However, this example is important because it illustrates the kind of pitfalls that can arise in the presence of digital systems and software control algorithms.

It should be reiterated at this point that this example initiating event is chosen for ease of illustrating the implementation of the dynamic methodologies under consideration and is not indicative of the limitations of the Markov/CCMT methodology or DFM, as well as

[7]Neutron flux oscillations with scram following recirculation pump trip in La Salle 2, Illinois on 3/9/1988. Power oscillations after a turbine trip with pump runback in Oskarshamn 2, Sweden on 2/25/1999. Feedwater oscillations in Harris plant, North Carolina during start-up at 7% power on 1/2/2002. Also see [127].

being representative of the possible types of interactions relevant to the benchmark DFWCS which may lead to errors not easily identifiable by conventional methods. For example, power is not assumed to be constant in time which leads to the artifact described above. Similarly, the BFV position is a function of the BC and BFV controller states and may reflect history dependence which again is not representable by the conventional ET/FT methodology. As also indicated earlier, another reason for the choice of the initiating event is to achieve a reduction of the system state space for reliability model construction to provide clarity in illustrations. All the features of the benchmark system can be modeled by both the Markov methodology and the DFM (see Sections 3 and 4). Section 2.5.1 below reduces the state transitions of the DFWCS described in Section 2.3 to those relevant to the example initiating event.

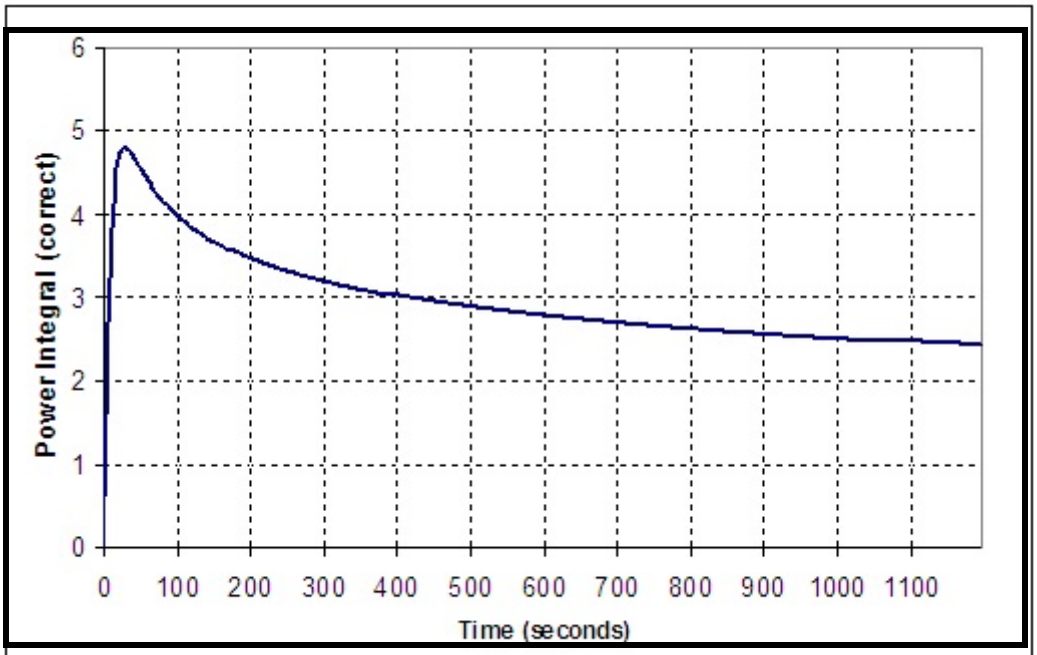

Figure 2.5.11 Correct Evaluation of the Integral in Eq. (2.5.20)

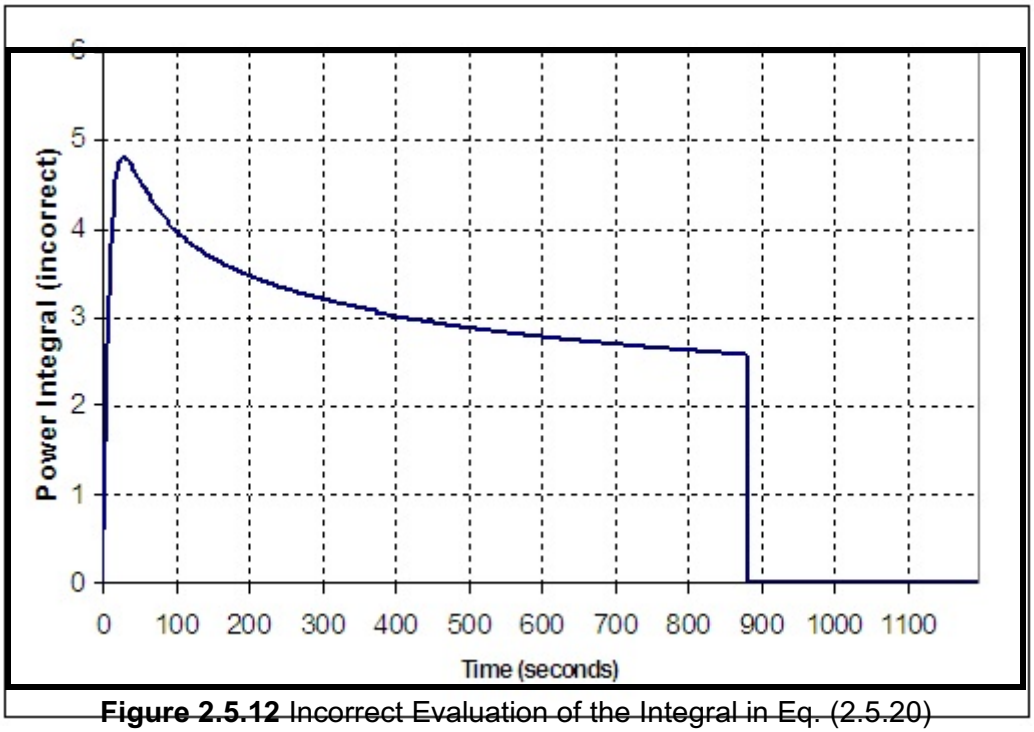

Figure 2.5.12 Incorrect Evaluation of the Integral in Eq. (2.5.20)

2.5.1 Example Initiating Event Transitions

As presented above, the components involved in the example initiating event are BC, BFV and BFV controller. In order to simplify the overall system presented in Fig. 2.3.3 to clarify the illustration of reliability model construction with the Markov/CCMT methodology and DFM, the following assumptions have been made:

1. Loss of both inputs only (and not possibly one).
2. Only the BFV controller failure can generate arbitrary output. If BC generates arbitrary output due to internal failure, it is recognized by the BC and BC transits to State D.
3. Loss of communications between the sensors and BC and between BC and BFV controller cannot be recovered.
4. The BFV controller cannot fail in Output High mode.
5. FP cannot fail.

The BFV controller Output Low mode failure is included in 0 vdc Output in Fig. 2.5.13 since they have the same effect on the actuated device (i.e. valve totally closed). Also, since the MC has failed (not recoverable) and only the BC computer is operating, the computer-computer connections (see Fig. 2.3.2) are reduced to the one presented in MS 3 only. Moreover, due to Assumptions 1 and 2, Loss of One Input and Arbitrary Output (Fig. 2.3.1) are not considered in intra-computer interactions .

From Assumption 3, the Plane 2 that represent the loss of output of the controller in Fig. 2.3.3 is no longer needed. Subsequently, Loss of Output leads the controller to transit to the 0 vdc Output directly. Figure 2.5.13 shows the possible states of the DFWCS components for the example initiating event which are defined more explicitly below:

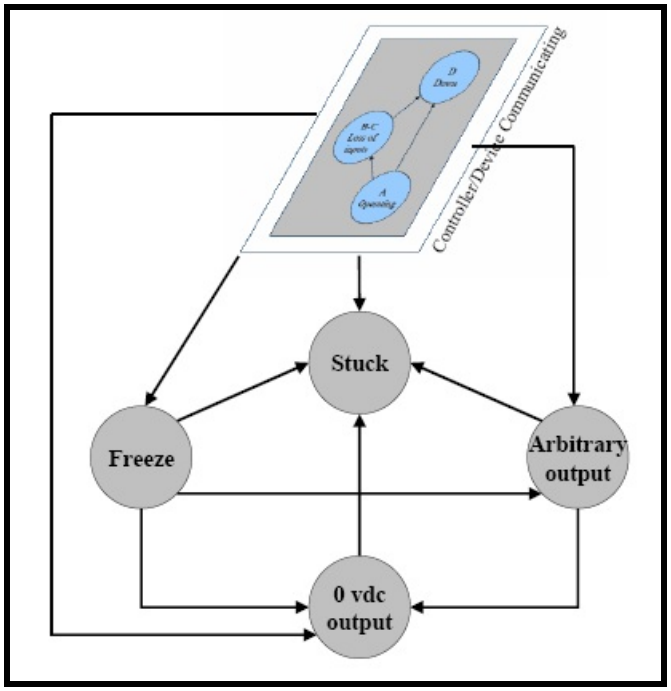

Figure 2.5.13 Example Initiating Event Transitions

- **Controller/Device Communicating**: This state specifies that the combined BFV controller and BFV are operating correctly and contains the following BC states:
    - State A: BC operating correctly
    - State B-C: BC does not receive any signals from the sensors (BC detects 0.0 vdc in the sensor reading)
    - State D: BC down

- **Freeze**: BFV controller recognizes that BC is down and maintains the position of the BFV.

- **Stuck**: BFV remains stuck in the same position due to mechanical failure.

- **Arbitrary Output**: BFV controller is sending random data to the BFV due to internal failure (software/firmware/hardware).

- **0 vdc Output**: 0.0 vdc on the line which connects controller and valve. This can be due to both a loss of communication between BFV controller and BFV or also due to the Low Output mode failure of the BFV controller.

A list of the possible transitions between these states are presented in Table 2.5.2.

Table 2.5.2 Possible Transitions for the Example Initiating Event

Failure mode	Transitions
BFV stuck due to mechanical failure	3, 5, 6, 7
BFV controller drifts low or 0 vdc on the line	1, 9, 10
BC down due to internal failure (e.g. watchdog timer), power loss or loss of output	2
Random output from the BFV controller to the BFV	4, 8

The BFV position is a function of the BC and BFV controller states and may reflect history dependence. In this respect, the allowed possible combinations of the component states are as presented in Table 2.5.3.

Table 2.5.3 BFV Position as Function of the System State for the Example Initiating Event

n	BFV Controller	BC	BFV Position
1	Controller/Device Communicating	State A	Calculated from Eqs. (2.5.17)-(2.5.21)
2	Controller/Device Communicating	State B-C	Maintained
3	Controller/Device Communicating	State D	Maintained
4	Freeze	---	Maintained
5	Arbitrary Output	---	Random in the range 0-100%
6	0 vdc Output	---	Closed
7	Stuck	---	Maintained

3. DESCRIPTION OF THE DYNAMIC FLOWGRAPH METHODOLOGY

This chapter presents the features of the DFM. Section 3.1 discusses the modeling features, Section 3.2 presents the analytical modules, and Section 3.3 outlines the quantification element. Section 3.4 uses the benchmark system to illustrate the model construction, analysis and quantification processes. Section 3.5 discusses the application of DFM to the example initiating event discussed in Section 2.5.

The DFM is a software analytical toolset developed to support PRA. It has been demonstrated in pilot U.S. NRC and NASA applications [4, 128-130]. DFM combines multi-valued logic modeling and analysis capabilities that are well suited for systems constituted of components that have multiple degraded states and exhibit dynamic behavior. The multi-valued logic algorithms implemented in DFM is especially suited to analyze non-coherent logic structures. An example of a non-coherent logic structure is a fault tree with "NOT" gates. Three DFM features of note enable it to support the modeling and analysis of dynamic systems:

- The capability to model and analyze feedback loops and time transitions,
- The ability of the deductive and inductive modules to analyze detailed multi-valued logic models to find interactive failure modes.
- The capability to quantify the Top Events analyzed by the deductive analysis module.

The essential steps in applying DFM in a PRA framework are:

1. Construct a DFM model to represent the system of interest.
2. Analyze the DFM model.
3. Quantify the results.

In applying DFM, the system of interest is first represented in a digraph (directed graph [131]) model. The DFM model is enriched with the explicit identification of the cause-and-effect and timing correspondences among the significant states of the parameters that are best suited to describe the system behavior. Once such a model has been produced, automated deductive or inductive algorithms that are built into the methodology can be applied to it. The deductive procedures are applied to identify how system level states (which may represent specific conditions of interest, be they success, anomaly or failure states) can be produced by any combinations and sequences of basic component states. This is accomplished by backtracking through the DFM model of the digital control system in a systematic, specified manner, and by expressing the conditions that cause the system events of interest in the form of timed prime-implicants. Conversely, inductive procedures can be applied to the same model, to determine how a particular basic component state can produce various possible sequences and system-level states. If a deductive analysis was executed, the quantification module can

estimate the probability of the Top Event from the probability of the events identified in the prime implicants. Thus, DFM can provide the multi-state and time-dependent equivalent of both fault tree analysis (FTA) and failure mode and effect analysis (FMEA), with the advantage that, once the DFM model of a system has been developed, the DFM system model already contains all the information necessary for the automated execution of these analyses for any system condition of possible interest. This can be compared, for example, with the execution of FTA, in which each system Top Event requires a separate manual input of AND/OR gate information.

3.1 DFM Model Construction

A DFM model is a graphic network that links key process parameters to represent the cause-and-effect and the time-dependent relationships. In particular, for a digital control system, both the controlled/monitored process and the controlling software itself are represented in the DFM model. A DFM model is an integration of a "time-transition network", a "causality network" and a "conditioning network", which is built by using detailed multi-state representations of the cause-and-effect and time-varying relationships that exist among the key system and software parameters.

The networks mentioned above are constructed from the DFM modeling elements. These modeling elements, as well as the manner in which they are assembled to form the three networks of a DFM model, are discussed below.

3.1.1 DFM Modeling Elements

A DFM model makes use of certain basic modeling elements to represent the temporal relations and the logical relations that exist in the system and the associated software. More specifically, a DFM model integrates a "time-transition network" which describes the sequence in which software subroutines are executed and control actions are carried out, a "causality network" that shows the functional relationships among key hardware and software parameters, and a "conditioning network" which models discrete software behavior due to conditional switching actions and discontinuous hardware performance due to component failures.

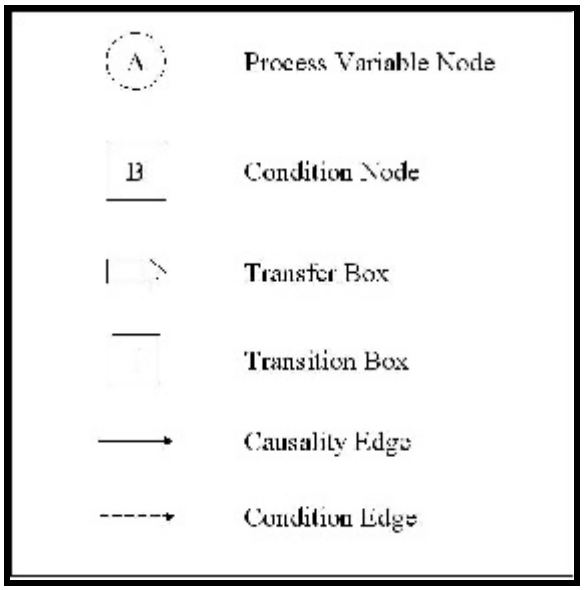

Figure 3.1.1 DFM Model Elements

The building blocks of these three types of networks are process variable nodes, condition nodes, causality edges, condition edges, and transfer/transition boxes and their associated decision tables. These basic modeling elements are shown in Figure 3.1.1.

3.1.1.1 Process Variable Nodes

Process variable nodes represent physical and software variables that are required to capture the essential functional behavior, continuous or discrete, of the digital control system. A variable represented by a process variable node is discretized into a number of states. The reason for the discretization is to simplify the description of the relations between different variables. The choice of the states for a process variable node is often dictated by the logic of the system. For instance, it is natural to set a state boundary at a value that acts as a trigger point for a switching action or a value that indicates the system is progressing towards failure. The number of states for each variable must be chosen on the basis of the balance between the accuracy of the model and the complexity introduced by higher numbers of variable states.

3.1.1.2 Causality Edges

Causality edges are used to connect process variable nodes to indicate the existence of a cause-and-effect relationship between the variables described by the nodes. The precise nature of the functional relationship (or the transfer function) is described by a transfer box that is always directly associated with each causality edge (please see discussion in Section 3.1.1.3 below).

3.1.1.3 Transfer Boxes and Associated Decision Tables

A transfer box represents a transfer function between process variable nodes. The quantification of the transfer function, i.e., the manner in which the states of the input process variable nodes are correlated with those of the output process variable nodes, is described by decision tables associated with each transfer box.

A decision table is associated with each transfer box and is used to quantify the relationships between its input and output process variable nodes. This table is a mapping between the possible combinations of the states of the input process variable nodes and the possible states of the output process variable nodes. Decision tables are an extension of truth tables in that they allow each variable to be represented by any number of states. These tables have been used in earlier developments to model components of engineering systems [132-134].

Because each transfer box input or output variable is a vector of states, and each combination of input states maps to a state of each of the output variables, each decision table is actually a multi-dimensional matrix whose dimension is equal to one plus the number of its inputs. For simplicity and convenience of representation, all decision tables can be reduced to a two-dimensional form. In this simplified form, there will be a column for each input variable and a column for each output variable of interest.

Decision tables can be constructed from empirical knowledge of the system, from physical equations that govern the system behavior, or from available software code and/or pseudo code. Building decision tables with empirical knowledge and/or the pseudo code provides a means of modeling the intended behavior of a system, and thus allow analysis to be performed on the specifications or the design concept, even before the system exists. On the other hand, using physical equations and running module testing to fill the decision table rows with detailed input/output state mappings creates a model reflecting the actual behavior of the system, thus enabling the actual system to be verified. The accuracy of the decision tables is crucial for the analysis because it directly correlates to the fidelity of the model (its ability to predict system behavior). Hence, to keep decision tables from growing too big, a judicious selection of the number of states into which each node is discretized should be made, without at the same time losing too much of the more detailed system-behavior information.

3.1.1.4 Condition Edges

Unlike causality edges, condition edges are mostly used to represent true discrete behavior in the system. They link parameter nodes to transfer boxes, indicating the possibility of using a different transfer function to map input variable into output variable states.

3.1.1.5 Condition Nodes

Condition nodes, like process variable nodes, represent physical or software parameters. However, condition nodes are used in DFM to more explicitly identify component failure states, changes of process operation regimes and modes, and software switching actions. Condition nodes represent variables that can affect the logic superstructure of the digital control system by modifying the causal relations between the process variable nodes. Condition nodes that are linked to causality edges and upstream process variable nodes are at the same time process variable nodes as well as condition nodes, but condition nodes whose states are not determined by other upstream process variable nodes are treated in DFM as "random variables", i.e., as variables that can be assumed to be in any of their possible states. In the latter case, a distribution of "relative frequency" of the associated states could also be assumed, for purposes of probabilistic quantification. It should be noted that the effect of a condition node on an output variable is modeled through a decision table, as is the case for a process variable node. The reason for having the added modeling elements of condition nodes and condition edges is to offer a clear distinction between continuous and discontinuous behavior in a system.

3.1.1.6 Transition Boxes and Associated Decision Tables

Transition boxes are similar to transfer boxes in that they connect process variable nodes to indicate cause-and-effect relationships. Condition nodes can be associated with transition boxes to represent discontinuous behavior between the input and output process variable nodes. Decision tables are again used to describe the relationships between the input and output process variable nodes. However, transition boxes differ from transfer boxes in the essential aspect that a time lag or time transition is assumed to occur between the time when the input variable states become true and the time when the output variable state(s) associated with the inputs is(are) reached. This time delay is a characteristic of the transition which is being modeled and is treated as an attribute of the transition box. Transition boxes are routinely used to model the execution of software routines and the handling of interrupts, which often play an important role in the execution flow of digital control systems software. They can, of course, also be used to model hardware time transitions..

3.1.1.7 DFM Model Construction and Integration

To construct a DFM model for a digital control system, the first step is to select the hardware components and the software/firmware functions that are to be included in the model. Following that, the controlled/monitored process parameters and software variables that capture the essential behavior of these components and software/firmware functions are identified and represented as process variable nodes. These process variable nodes are then linked together

by causality edges through transfer boxes or transition boxes to form an integrated "causality" and "time-transition" network. Discrete behaviors such as component failures and logic switching actions are then identified and represented as condition nodes, which are tied to transfer boxes and transition boxes expressly to show how a "conditioning network" of discrete actions and events actually interacts with and affects the integrated "causality" and "time-transition" network. The parameters represented by the process variable nodes and condition nodes are discretized into meaningful states, and decision tables are constructed to relate these states. The decision tables can be constructed by empirical knowledge of the system, from the equations that govern the system behavior, or from available software code and/or pseudo code. In particular, when modeling a system that includes actual software, module testing (which itself constitutes the basic first step of standard software testing procedures) becomes an integral part in the creation of the decision tables that mimic the actual behavior of the software. The completed DFM model then reflects the essential causal, temporal, and logical behavior of the digital control system. The example discussed in Section 3.4 will illustrate how these steps are carried out.

For typical control systems, the associated DFM model is complex in general. However, the complexity does not make the tasks of quality assurance and quality control for the model particularly difficult. This is because the model can be checked piece-wise. The model that represents the whole system can be first partitioned into subsets, each subset representing a specific function of the system. These subsets can be further partitioned if necessary. Model checking can be done for these partitions to ensure the accuracy in the model construction process.

Additionally, for a complex system, modularization of the modeling step can be used to manage complexity. Conceptually, the modularization step is analogous to having transfer gates in fault trees. For instance, to model a system with *N* major sub-systems, it is useful to develop a top-level DFM model to represent the causal and temporal relationships between these major sub-systems. The parameters in the top-level model corresponding to these major sub-systems are then expanded in second level DFM models to characterize their detailed behavior. If these sub-systems are made up of components that can be decomposed, the model hierarchy can be further expanded into lower levels. It is important to point out that the modularization step is desirable but not essential. The user can construct a single model to represent the full complexity of the system. However, a model hierarchy that represents complexity progressively is generally easier to construct, follow and analyze.

3.2 DFM Model Analysis

The analysis of a DFM system model constructed according to the rules described above can be conducted by tracing sequences of events either backward from effects to causes (i.e., "deductively"), or forward from causes to effects (i.e., "inductively") through the model structure. Inductive and deductive analyses can be combined to analyze the system within the context of 1) design verification, 2) fault analysis, or 3) automated test sequence generation. These

analysis techniques will be discussed in Sections 3.2.2 to 3.2.4. Section 3.2.1 presents an overview of the DFM deductive and inductive analysis procedures.

3.2.1 Deductive Analysis and Inductive Analysis

Once a DFM model is constructed, deductive analysis can be carried out to identify root causes for different Top Events, and inductive analysis can be executed to visualize event sequences as a result of different initial conditions.

3.2.1.1 Deductive Analysis

The deductive engine backtracks the time and causality of the DFM model to identify timed prime implicants for Top Events of interest. These time prime implicants, characterized by the combinations and sequences of basic variable states, represent the full set of minimal conditions that could lead to the Top Event. Prime implicants are the multi-valued logic equivalent of minimal cut sets in traditional fault tree analysis. The DFM prime implicants are logically compatible with cut sets generated by automated PRA tools such as SAPHIRE. Hence, DFM results can be exported into the SAPHIRE environment.

3.2.1.1.1 Multi-Valued Logic and Prime Implicants

Deductive DFM analysis can be thought of as multi-valued logic extension of fault tree analysis. As DFM models represent physical variables (e.g., pressure, temperature, voltage, etc.), binary logic (in which only two states may be used to characterize each variable space) is, in general, not sufficient for an adequate representation of the behavior of the system. DFM models thus employ Multi-Valued Logic, wherein each variable space may be discretized into an arbitrary number of states. A DFM deductive analysis expansion, therefore, would contain non-binary primary events (or certain equivalent binary expressions containing groups of mutually exclusive binary primary events, which may be defined ad-hoc to signify whether the assertion that a given multi-valued variable is in any one of its states is true or false). The DFM prime implicants derived as a result of deductive analysis are, in general, non-coherent. An intuitive, rather than formal way, of understanding this is by noting that DFM variable states are not ordered in such a way that higher states always indicate "increasingly-faulted" conditions and lower states always indicate "increasingly-nominal" conditions. Thus, as a basic variable changes from a lower to a higher state, the system-state indicator variable of choice for the particular analysis of interest may be going in the opposite direction, i.e., from a higher to a lower state.

The Top Event of a deductive DFM analysis can still be expressed in disjunctive form (the form of a disjunction of conjunctions of primary events), but the multi-valued logic analogue of the minimal cut sets encountered in binary fault trees are known as prime implicants [135, 136]. A prime implicant is any monomial (conjunction of primary events) that is sufficient to cause the

Top Event, but does not contain any shorter conjunction of the same events that is sufficient to cause the Top Event. The prime implicants of a function are unique and finite; however, finding them is a more challenging task than finding binary logic minimal cut sets.

DFM uses decision tables to map the combinatorial states of transfer box inputs to their outputs. Decision tables allow each variable to be represented by any number of states, and they have been applied in fault tree analysis in the past to model component behavior. Given the state of a transfer box output node, the decision table gives the complete sets of inputs that could have caused it. Since a decision table is, itself, essentially a disjunction of conjunctions of states, it is possible to generate prime implicants from the table.

When referring to prime implicants in the context of a DFM analysis, another important observation is that the presence of the time element in the DFM modeling framework introduces the possibility of prime implicants that would not be possible in ordinary time-invariant logic. In the latter, in fact, a prime implicant of the form:

<variable A = 2 AND variable A = 3>

would not be possible, and, if found in the course of a time-invariant analysis, would have to be eliminated by application of explicit "physical consistency rules." In the application of DFM to time-dependent systems, however, if a time-transition has been encountered and the prime implicant is thus "time-stamped" to indicate:

<variable A = 2 @ time t = T1 AND variable A = 3 @ time t = T2>,

then the logical inconsistency no longer exists, and the prime implicant can be considered possible (unless of course it violates a "dynamic consistency rule," which still applies in time-dependent logic; please refer to Section 3.2.1.1.3). All prime implicants identified in a DFM analysis are conjunctions of primary events with associated time stamps.

DFM, therefore, represents a significant advancement beyond conventional fault tree analysis. In particular, a conventional fault-tree produces cut-sets for one, and only one, binary Top Event, with no associated time dependent information. The DFM representation is one or two orders of magnitude more powerful, because it produces multi-valued logic and time-dependent prime implicants for a very large number of possible Top Events. A DFM Top Event can in fact be chosen to be any state among all the possible states of any of the variables, or even any combination of states of separate variables. This is in addition to the fact that, once a DFM system model has been constructed, it can be used repeatedly to investigate many different Top Events.

In many digital control systems, there are feedback or feedforward characteristics. This can cause a node to be traced back to itself in the fault tree construction. Consistency rules must be applied when these situations are encountered. Inconsistent rows are then pruned from the transition tables. Two major classes of consistency rules have been identified which are the "physical" consistency rules and the "dynamic" consistency rules.

3.2.1.1.2 Physical Consistency Rules

Physical consistency rules are applied to eliminate physically impossible conditions from the deductive analysis. An example of this would be a system parameter taking on two different values at the same time step in the timed fault tree. This class of consistency rule is similar to the consistency rules applied in conventional static fault tree analysis. If the same variable appears twice in the same row of a transition table, but in different states at the same time step, then that particular row must be pruned from the transition table due to physical inconsistency.

3.2.1.1.3 Dynamic Consistency Rules

In a deductive analysis, it is sometimes advantageous to define dynamic consistency rules to prune out conditions that are not compatible with the dynamic constraints of the system of interest. Eliminating the incompatible conditions will reduce the number of intermediate events and prime implicants generated, thus, making the analysis more efficient. For instance, dynamic consistency rules can be defined to constrain:

- The direction of change of certain parameters. For example, if repair is not available, a component, once it enters into a failed state, remains in that state, or
- The rate of change of certain parameters.

3.2.1.2 Inductive Analysis

In addition to the deductive engine, the inductive engine can be executed to determine how a particular set of basic variable states (the initial condition) produces various sequences and system level states. Starting from a set of initial conditions, the inductive engine follows the causality and timing represented in the model to determine the resulting sequence of events.

Thus, in the deductive and inductive engines, DFM provides the multi-state and time-dependent equivalent of ET/FT analysis and failure mode and effect analysis. The substantial advantage is that once the DFM system model has been developed, the same model can be analyzed deductively and inductively an unlimited number of times by automated execution. This is more efficient compared to the integration of the former classical techniques.

3.2.2 Design Verification

Deductive and inductive DFM analysis techniques can be applied to verify a system design against system level requirements. In this type of design verification analysis, the goal is to show that the system designed satisfies requirements that describe desirable system properties. Deductive analysis or inductive analysis can be carried out, depending on the characteristics of the requirement statements being checked. If the requirement statements specify desirable system properties, such as a particular condition must occur as a result of some triggering conditions, an inductive analysis can be executed to show whether this is indeed the case. The initial condition is defined to be the triggering condition, and the subsequent conditions identified by the Analysis Engine through a propagation of the system model are checked to see if the condition that is specified in the requirements can be reached. On the other hand, if the requirement statements specify undesirable system properties, such as certain conditions that must not occur after some prior conditions, these type of statements can be easily verified by a deductive DFM analysis. The Top Event condition is defined as the conjunction of the undesirable condition and the prior condition, separated by a number of time steps specified in the requirements. If the Analysis Engine does not find any prime implicant for this Top Event, this means that no pathway exists by which the undesirable condition can result from the prior condition, and hence the requirement is satisfied.

3.2.3 Failure and Fault Analysis

Besides as tools for system design verification, deductive and inductive DFM analysis techniques can also be applied to identify potential faults in the system or investigate the effects of basic component failure modes on the system performance. The first type of failure and fault analysis uses the deductive analysis technique. A Top Event is defined as an undesirable system level condition, and the prime implicants identified by the Analysis Engine represent the potential faults that could lead to this system level condition. The second type of failure and fault analysis uses the inductive analysis technique, and can be referred to as an automated FMEA. Combinations of basic component failure modes are defined in the initial condition and the boundary condition, and the Analysis Engine propagates these through the system model to see their effects downstream in subsequent time steps. The reader should note that if a combination of basic failure modes is being investigated, the individual failure modes do not need to occur in the same time step. The initial condition and the boundary condition can be defined to represent special failure profiles, where the failure events follow in sequential order. Examples of DFM being applied in failure and fault analysis will be provided in Section 3.5.2.

3.2.4 Automated Test Vector Generation

Testing is traditionally one of the most important activities carried out to assure that a given design, in its actual implementation, complies with certain assigned constraints and

specifications in the areas such as peak performance, safety and reliability. In relation to software, testing is often performed by feeding inputs, generated randomly or according to some pre-ordained scheme, into the software and observing the produced outputs to verify correct behavior, or to uncover faults and errors. In practice, to achieve some minimum level of assurance, a large number of input cases are needed, making testing both expensive and time consuming. For the more complex systems in existence today, the monetary and time cost of testing has skyrocketed and assurance by testing alone is already impossible to achieve in true quantitatively defensible terms. Thus, although in all likelihood testing will remain an important pillar in ensuring system and software dependability, it is already clear that analytical tools should be developed and used to complement and guide the testing process in more effective and efficient ways than afforded by the random or heuristic input generation techniques commonly employed today.

In this regard, DFM can be used to analyze the software and system design and to identify special input combinations that can distinguish between the normal states and the faulted states of specific components. The automated test vector analysis procedure in DFM is an extension of the Automatic Test Vector Generation (ATVG) procedures used in the testing of digital circuits. More specifically, this DFM procedure is the multi-valued logic equivalent of the Boolean Difference based procedures formulated for binary circuit ATVG. The goal of the digital circuit ATVG procedure is to identify special test input combinations, such that changing these inputs in specific ways will cause the observable circuit output to change only if the circuit is free of faults. On the other hand, if the specific type of fault for which the test is carried out is present, the observable output will remain unchanged. The DFM automated test vector analysis procedure is capable of handling multi-valued logic functions modeled with DFM, enabling the procedure to be applied to test complex software and control systems which exhibit analog-equivalent behavior. In this DFM-based ATVG procedure, the Boolean Difference method for binary logic is reformulated, and is translated into a procedure for reducing the prime implicants associated with specially defined Top Events. The reduced set of prime implicants defines the special input combinations which can be used to test for specific faults. Like their binary circuit counter parts, these inputs have the characteristic of causing the observable outputs to change if a specific fault is absent, but to remain constant if that fault is present.

3.3 Quantification of Deductive Analysis Results

The quantification module is used to quantify results obtained in a deductive analysis. It estimates the probability of the Top Event based on the probability estimates of the basic events that make up the prime implicants. Suppose a deductive analysis yields n prime implicants, PI#1 through PI #n, as shown in Eq. (3.1):

$$\text{Top Event} = \text{PI \#1} \vee \ldots \vee \text{PI \#n} \qquad (3.1)$$

$$\text{PI \#I} \not\subset \text{PI \#j, for any I} \neq \text{j}$$

This set of prime implicants is first converted into a set of m mutually exclusive implicants, MEI #1 through MEI #m, as shown in Eq. (3.2). These mutually exclusive implicants can be thought of as the multi-valued logic equivalent of cut sets that do not yield any cross product term. Thus, the sum of the probabilities of these mutually exclusive implicants yields the probability of the Top Event, as shown in Eq. (3.3).

$$\text{Top Event} = \text{MEI \#1} \vee \ldots \vee \text{MEI \#m} \tag{3.2}$$

where $\text{MEI \#I} \wedge \text{MEI \#j} = \varnothing$ for any $I \neq j$ and

$$P(\text{Top Event}) = P(\text{MEI \#1}) + \ldots + P(\text{MEI \#m}). \tag{3.3}$$

The current toolset is able to process point estimate inputs, i.e., if the probabilities of the basic events are point estimates. The extension of the quantification module to incorporate failure rates and uncertainty is currently under development.

3.4 Benchmark System Application

This section describes the application of DFM to model and analyze the benchmark system. Section 3.4.1 presents the DFM model construction process. Section 3.4.2 discusses the analyses performed and the results obtained.

3.4.1 Benchmark System DFM Model

The DFM model of the benchmark DFWCS (see Fig. 2.1.2) encompasses both the digital controllers and the process being controlled (i.e., the steam generator and the feedwater system). This DFM model is shown in Fig. 3.4.1. In this figure, the sub-model for the digital control system is shown as a black box. It is expanded in its full detail in Fig. 3.4.2.

The process variable nodes represent the key process variables for the controller and the controlled process, and the condition nodes represent the components with the different failure modes. These process variable nodes and condition nodes are explained in Table 3.4.1.

These process variables nodes are each discretized into a finite number of states. The process variable nodes are linked together to model the temporal and causal behavior of the coupled system. The condition nodes are next linked to represent the discrete jump in behaviors when components fail. For example, transfer box 26 on the top right portion of Fig. 3.4.1 shows that LM, the measured SG level is a function of L, the actual steam generator level. The relationship is conditioned upon LS, the state of the level sensor. When LS is normal, the relationship is nominally proportional. However, if LS failed high, LM will be in a high state regardless of L.

The decision tables corresponding to the process portion are constructed by executing the simulator, and those corresponding to the controller portion are developed by executing the software modules.

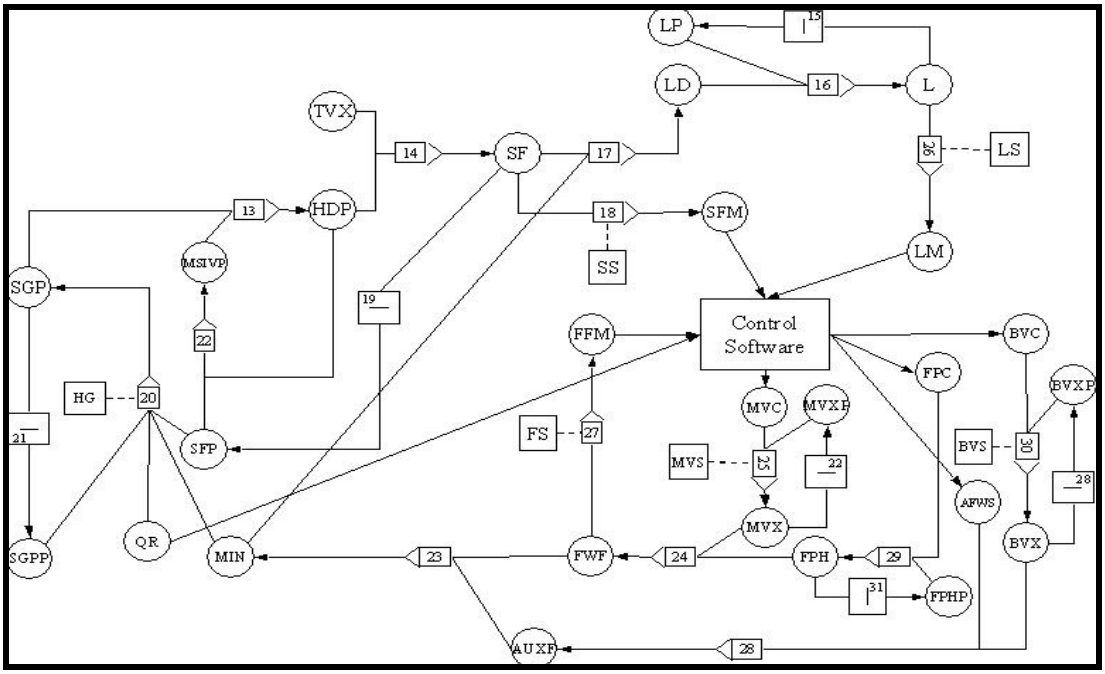

Figure 3.4.1 DFM Model of the Benchmark System

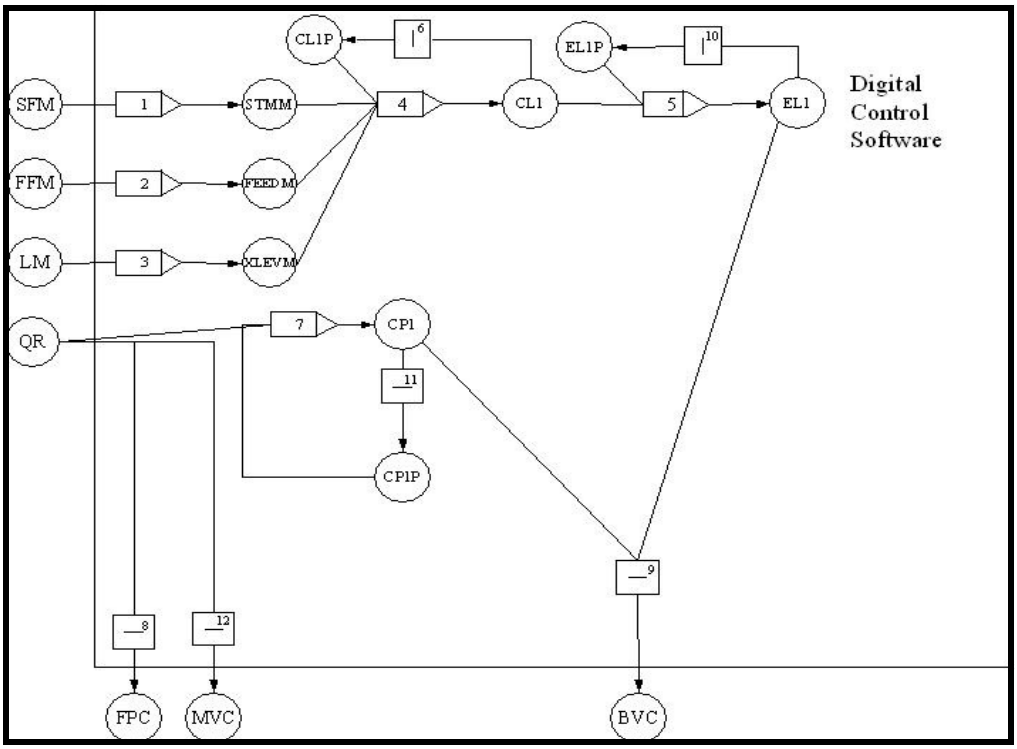

Figure 3.4.2 DFM Model of the Digital Feedwater Control System

Table 3.4.1 Description of the Nodes in the DFM Model

Node	Description

AFWS	Auxiliary feedwater system command
AUXF	Auxiliary feed flow
BVC	Bypass valve command
BVX	Bypass valve position
BVXP	Bypass valve position in the previous cycle
CL1	Compensated water level
CL1P	Compensated water level in the previous cycle
CP1	Compensated power
CP1P	Compensated power in the previous cycle
EL1	Level error
EL1P	Level error in the previous cycle
FEEDM	Software representation of the measured feed flow
FFM	Measured feed flow
FPC	Feed pump command
FPH	Feed pump head pressure
FPHP	Feed pump head pressure in the previous cycle
FS	State of the feed flow sensor suite
FWF	Feed flow
HDP	Steam header pressure
HG	Vapor state at the top of the steam generator
L	Steam generator level
LD	Change in the steam generator level
LM	Measured steam generator level
LP	Steam generator level in the previous cycle
LS	State of the steam generator level sensor
MIN	Total mass flowrate into the steam generator
MSIVP	Main steam insolation valve position
MVC	Main feed valve command
MVS	State of the main feed valve
MVX	Main feed valve position
MVXP	Main feed valve position in the previous cycle
QR	Reactor power
SF	Steam flow

3.5 Example Initiating Event Application

This section will illustrate the DFM model construction and analysis steps using the example initiating event discussed previously in Section 2.5.

3.5.1 DFM Model for the Example Initiating Event Application

The DFM model of the example initiating event encompasses the backup computer, the bypass flow valve, the bypass flow valve controller, the inputs and outputs for the backup computer, and the control law and logic for maintaining the steam generator level. This DFM model is shown in Figure 3.5.1. The process variable nodes represent the key components in the simplified system. These process variable nodes are explained in Table 3.5.1.

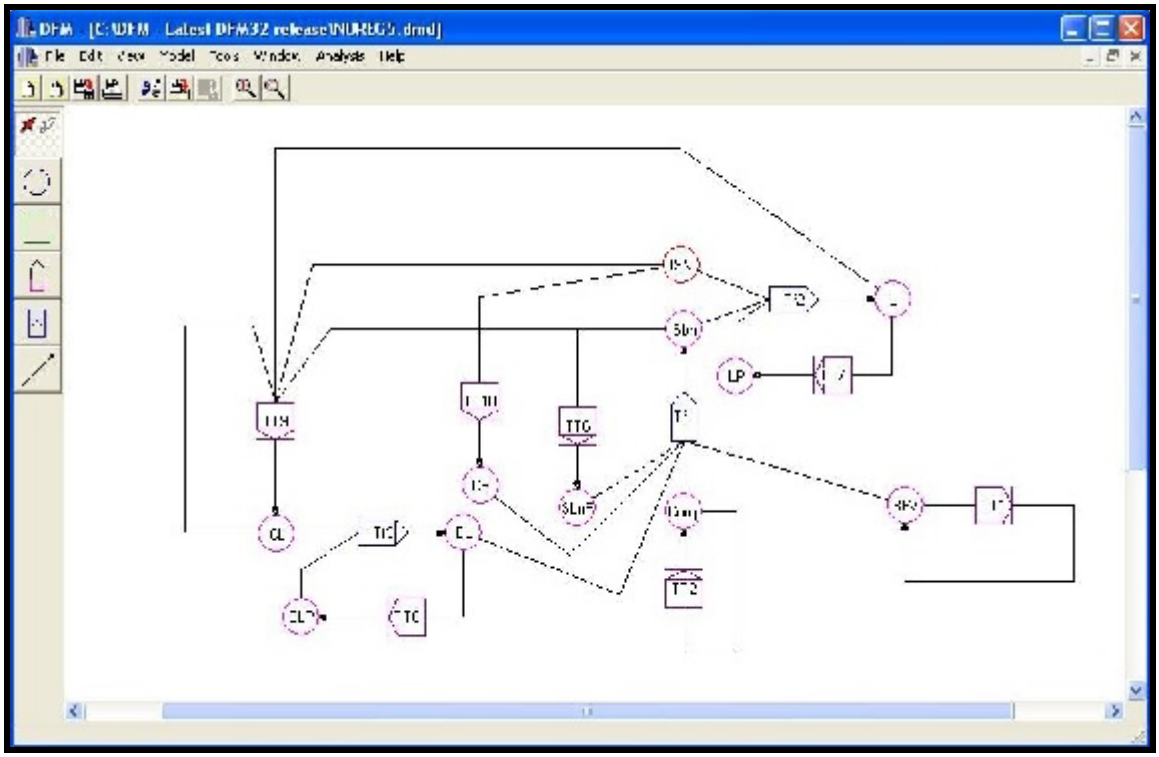

Figure 3.5.1 DFM Model for the Example Initiating Event

These process variable nodes are each discretized into a finite number of states. The discretization of these process variable nodes are shown in Table 3.5.2 to Table 3.5.12. In constructing this DFM model, it was assumed that once the BFV + Controller or the computer enters a failure state, it will remain in that state. Hence, the states F-S, Frz, Arb and Zero for

node BFV (in Table 3.5.2) and the states LossIn and Down for the node Comp (in Table 3.5.4) were assumed to be sink states.

The process variable nodes defined are linked together to model the temporal and causal behavior of the system under the example initiating event condition. For example, the transfer box Tf1 on the bottom center portion of Figure 3.5.1 shows that with the main computer out of commission, the bypass flow valve position is a function of the backup computer, the BFV, the BFV controller, the compensated power, the compensated level error and the previous BFV position. The relationship between the nodes are summarized in the decision tables. The decision tables for the transition boxes TT9 and TT10, and the transfer boxes Tf1, Tf2 and Tf3 are developed from the control equations in Eqs. (2.5.1) to (2.5.8). The decision tables for the other transfer boxes and transition boxes are developed based on the logic behavior of the system.

Table 3.5.13 and Table 3.5.14 show examples of the decision tables developed for this model. Table 3.5.13 is the decision table for transfer box Tf2. It shows how the BFV position (Node Sbn), the steam flow (Node fSN) and the steam generator level (Node LP) at the previous time step will influence the steam generator level (Node L) at the current time step. On the other hand, Table 3.5.14 is the decision table for transition box TT7. It updates the state of the steam generator level. Thus, the decision tables for the transfer boxes and the transition boxes can model the dynamic behavior of parts of the system.

The reader should note that one time step in the DFM model is meant to represent a number of clock cycles sufficient to see changes in the states of the process variables. Hence, the DFM model compresses time when it undergoes the deductive and inductive analyses. When the analysis determine that a process variable will change in 1 time step, the DFM tool does so without keeping track of the detail evolution millisecond by millisecond. This makes the analysis very efficient and preempts any combinatorial explosion of states. It is also important to point out that time is an implicit variable rather than a explicit variable in this DFM model. Time is implicitly related through the nodes that correspond to the steam flow.

Table 3.5.1 Description of the Nodes in the Simplified DFM Model

Node	Description
BFV	State of the bypass flow valve and controller
CL	Compensated level
Comp	State of the backup computer
CP	Compensated power
EL	Level error
ELP	Previous level error

fSN	Steam flow
L	Steam generator level
LP	Steam generator level in the previous time step
Sbn	Bypass flow valve position
SbnP	Previous bypass flow valve position

Table 3.5.2 Discretization of the Node BFV

State	Description
OP	Bypass flow valve/controller is operating
F-S	Bypass flow valve failed stuck
Frz	BFV controller is frozen
Arb	BFV controller fails in the arbitrary state
Zero	BFV controller fails in the zero state

Table 3.5.3 Discretization of the Node CL

State	Description
-1	[-500, -100)
0	[-100, 100)
+1	[100, 500]

Table 3.5.4 Discretization of the Node Comp

State	Description
OP	Backup computer is operating
LossIn	Loss of inputs to the backup computer
Down	Backup computer is down

Table 3.5.5 Discretization of the Node CP

State	Description
0	[0, 40)
1	[40, 80)
2	[80, 150]

Table 3.5.6 Discretization of the Node EL

State	Description
-1	[-1000, -1.587)
0	[-1.587, 4.203)
+1	[4.203, 1000]

Table 3.5.7 Discretization of the Node ELP

State	Description
-1	[-1000, -1.587)
0	[-1.587, 4.203)
+1	[4.203, 1000]

Table 3.5.8 Discretization of the Node fSN

State	Description
0	[0, 38)
1	[38, 94)
2	[94, 140]

Table 3.5.9 Discretization of the Node L

State	Description
-2	< -2.0
-1	[-2.0, -0.17)

0	[-0.17, 0.17)
+1	[0.17, 2.5]
+2	> 2.5

Table 3.5.10 Discretization of the Node LP

State	Description
-2	< -2.0
-1	[-2.0, -0.17)
0	[-0.17, 0.17)
+1	[0.17, 2.5]
+2	> 2.5

Table 3.5.11 Discretization of the Node Sbn

State	Description
0	[0, 30)
1	[30, 70)
2	[70, 100]

Table 3.5.12 Discretization of the Node SbnP

State	Description
0	[0, 30)
1	[30, 70)
2	[70, 100]

Table 3.5.13 Decision Table for the Transition Box Tf2

Sbn	fSN	LP	L
0	0	-2	**-2**
0	0	-1	**-1**

0	0	0	**0**
0	0	+1	**+1**
0	0	+2	**+2**
0	1	-2	**-2**
0	1	-1	**-2**
0	1	0	**-1**
0	1	+1	**0**
0	1	+2	**+1**
:	:	:	**:**

Table 3.5.14 Decision Table for the Transition Box Tt7

L@ current t	**LP@ next t**
-2	**-2**
-1	**-1**
0	**0**
+1	**+1**
+2	**+2**

3.5.2 Example Initiating Event DFM Analysis

Once the DFM model of the example initiating event is laid out and the decision tables constructed, the different kinds of analyses described in Section 3.3.2 to Section 3.3.4 can be carried out. Failure and fault analysis procedures are illustrated in the following sub-sections. When more information become available, such as the design specifications and/or the control software, the other two types of analyses, design verification and automated test vector generation for the software modules, can also be executed. Two deductive analysis examples are provided in Section 3.5.2.1, and two inductive analysis examples are presented in Section 3.5.2.2.

3.5.2.1 Example of Deductive DFM Analysis

For the example initiating event, failure and fault analyses using the deductive technique was carried out to find out the combination of component states that could lead to desirable or undesirable events of the Main Feedwater System.

For the failure and fault analysis example, to find out the prime implicants for a high steam generator level, the following Top Event was defined**:**

L = +2 @ t = 0 ^

L = +1 @ t = -1 ^

L = 0 @ t = -2 ^

ELP = 0 @ t = -2 ^

CL = 0 @ t = -2

This Top Event specified the progression of the steam generator level from 0 to 2, given nominal values of the level error and compensated level in the control software. In the deductive analysis of this Top Event, the Top Event is expressed as a transition table shown in Table 3.5.15. The header row shows the nodes and their associated time stamp and row 1 shows the combination of the states for the nodes of interest.

Table 3.5.15 Transition Table for the Top Event

L **t = 0**	**L** **t = -1**	**L** **t = -2**	**ELP** **t = -2**	**CL** **t = -2**
+2	+1	0	0	0

In this deductive analysis, the model was tracked backwards in time and causality. With the analysis time set to 0, the decision table for transition box TT7 was used to expand the transition table shown in Table 3.5.15. In particular, this expansion spelled out the combinations of fSN, Sbn and L @ t = -1 that give rise to L = 2 @ t = 0. The result of the expansion was the transition table shown in Table 3.5.16.

Table 3.5.16 Transition Table for after the first expansion

Sbn **t = -1**	**fSN** **t = -1**	**LP** **t = 0**	**L** **t = -1**	**L** **t = -2**	**ELP** **t = -2**	**CL** **t = -2**

-	0	+2	+1	0	0	0
1	0	+1	+1	0	0	0
2	0	+1	+1	0	0	0
1	1	+2	+1	0	0	0
2	1	+1	+1	0	0	0
2	-	+2	+1	0	0	0

To continue the deductive analysis, the causality shown in the model is backtracked. For the transition table shown in Table 3.5.16, the column corresponding to Sbn @ t = -1 was next expanded with the decision table for transfer box Tf1. This process was repeated after the whole model was traversed backwards for 2 time steps. The prime implicants shown in Table 3.5.17 were identified. In formal logic terms, these prime implicants describe the combinations of basic events that could cause the Top Event, but none of these prime implicants is contained in another. These prime implicants can be thought of as multi-valued logic equivalent of minimal cut sets in a fault tree analysis.

Top Event = Prime Implicant #1 $\vee$ ... $\vee$ Prime Implicant #10,

and Prime Implicant #I $\not\subseteq$ Prime Implicant #j

For the Top Event of interest, prime implicants #1 to #4 and prime implicants #6 to #9 identified the conditions that the BFV failed stuck, loss of inputs of the computer, the downing of the computer, or the freezing of the BFV controller, together with a steam flow-feed flow mismatch (feed flow > steam flow) will cause the steam generator level to rise. This is due to the fact that any of the failure will cause the feed flow to remain the same, while the steam flow gradually decreases. In particular, the BFV stuck at 44 s condition described in Section 2.5 falls under prime implicant #1. On the other hand, prime implicants #5 and #10 identified the condition corresponding to the BFV failure in the arbitrary state.

Table 3.5.17 Prime Implicants for High Steam Generator Level

#	Prime Implicant

1	L = 0 @ t = -2 ELP = 0 @ t = -2 CL = 0 @ t = -2 SbnP = 1 @ t = -2 fSN = 0 @ t = -2 BFV = F-S @ t = -2
2	L = 0 @ t = -2 ELP = 0 @ t = -2 CL = 0 @ t = -2 SbnP = 1 @ t = -2 fSN = 0 @ t = -2 Comp = LossIn @ t = -2
3	L = 0 @ t = -2 ELP = 0 @ t = -2 CL = 0 @ t = -2 SbnP = 1 @ t = -2 fSN = 0 @ t = -2 Comp = Down @ t = -2
4	L = 0 @ t = -2 ELP = 0 @ t = -2 CL = 0 @ t = -2 SbnP = 1 @ t = -2 fSN = 0 @ t = -2 BFV = Frz @ t = -2
5	L = 0 @ t = -2 ELP = 0 @ t = -2 CL = 0 @ t = -2 SbnP = 1 @ t = -2 fSN = 0 @ t = -2 BFV = Arb @ t = -2

6	L = 0 @ t = -2 ELP = 0 @ t = -2 CL = 0 @ t = -2 SbnP = 2 @ t = -2 fSN = 1 @ t = -2 BFV = F-S @ t = -2
7	L = 0 @ t = -2 ELP = 0 @ t = -2 CL = 0 @ t = -2 SbnP = 2 @ t = -2 fSN = 1 @ t = -2 Comp = LossIn @ t = -2
8	L = 0 @ t = -2 ELP = 0 @ t = -2 CL = 0 @ t = -2 SbnP = 2 @ t = -2 fSN = 1 @ t = -2 Comp = Down @ t = -2
9	L = 0 @ t = -2 ELP = 0 @ t = -2 CL = 0 @ t = -2 SbnP = 2 @ t = -2 fSN = 1 @ t = -2 BFV = Frz @ t = -2
10	L = 0 @ t = -2 ELP = 0 @ t = -2 CL = 0 @ t = -2 SbnP = 2 @ t = -2 fSN = 1 @ t = -2 BFV = Arb @ t = -2

If the probabilities for the basic event nodes (those that are not downstream of transfer boxes) in Figure 3.5.1 are defined, the Top Event can be quantified using the procedure outlined in

Section 3.3. The set of prime implicants is first converted into a set of mutually exclusive implicants. These mutually exclusive implicants can be thought of as the multi-valued logic equivalent of a cut sets that do not yield any cross product term. Thus, the sum of the probabilities of these mutually exclusive implicants yields the probability of the Top Event.

Top Event = MEI #1 $\vee$ ... $\vee$ MEI #m,

where MEI #I $\wedge$ MEI #j = $\varnothing$

P(Top Event) = P(MEI #1) + ... + P(MEI #m)

It is important to note that once a single DFM model is constructed, it can be analyzed for many different Top Events. For example, the same DFM model can be analyzed for the Top Event:

L = -2 @ t = 0 ^

L = -1 @ t = -1 ^

L = 0 @ t = -2 ^

ELP = 0 @ t = -2 ^

CL = 0 @ t = -2

This Top Event specified the progression of the steam generator level decreasing from 0 to -2, given nominal values of the level error and compensated level in the control software. For this particular Top Event, the 11 prime implicants shown in Table 3.5.18 were identified. Prime implicants #1 to #4 and prime implicants #7 to #11 identified the conditions that the BFV failed stuck, the computer loss of inputs, the downing of the backup computer, or the freezing of the BFV controller, together with a steam flow-feed flow mismatch (steam flow > feed flow) will lead to low level in the SG. This is due to the fact that any of these failures will cause the feed flow to remain the same, while the steam flow slowly decreases. In particular, the BFV stuck at 43 s condition described in Section 2.5 falls under prime implicant #1. On the other hand, prime implicants #5 and #11 identified the condition corresponding to the BFV controller failing in the arbitrary state, whereas prime implicant #6 identified the condition corresponding to the BFV controller failing in the zero state.

Table 3.5.18 Prime Implicants for Low Steam Generator Level

#	Prime Implicant

1	L = 0 @ t = -2 ELP = 0 @ t = -2 CL = 0 @ t = -2 SbnP = 0 @ t = -2 fSN = 1 @ t = -2 BFV = F-S @ t = -2
2	L = 0 @ t = -2 ELP = 0 @ t = -2 CL = 0 @ t = -2 SbnP = 0 @ t = -2 fSN = 1 @ t = -2 Comp = LossIn @ t = -2
3	L = 0 @ t = -2 ELP = 0 @ t = -2 CL = 0 @ t = -2 SbnP = 0 @ t = -2 fSN = 1 @ t = -2 Comp = Down @ t = -2
4	L = 0 @ t = -2 ELP = 0 @ t = -2 CL = 0 @ t = -2 SbnP = 0 @ t = -2 fSN = 1 @ t = -2 BFV = Frz @ t = -2
5	L = 0 @ t = -2 ELP = 0 @ t = -2 CL = 0 @ t = -2 fSN = 1 @ t = -2 BFV = Arb @ t = -2

6	L = 0 @ t = -2 ELP = 0 @ t = -2 CL = 0 @ t = -2 fSN = 1 @ t = -2 BFV = Zero @ t = -2
7	L = 0 @ t = -2 ELP = 0 @ t = -2 CL = 0 @ t = -2 SbnP = 1 @ t = -2 fSN = 2 @ t = -2 BFV = F-S @ t = -2
8	L = 0 @ t = -2 ELP = 0 @ t = -2 CL = 0 @ t = -2 SbnP = 1 @ t = -2 fSN = 2 @ t = -2 Comp = LossIn @ t = -2
9	L = 0 @ t = -2 ELP = 0 @ t = -2 CL = 0 @ t = -2 SbnP = 1 @ t = -2 fSN = 2 @ t = -2 Comp = Down @ t = -2
10	L = 0 @ t = -2 ELP = 0 @ t = -2 CL = 0 @ t = -2 SbnP = 1 @ t = -2 fSN = 2 @ t = -2 BFV = Frz @ t = -2

11	L = 0 @ t = -2 ELP = 0 @ t = -2 CL = 0 @ t = -2 fSN = 2 @ t = -2 BFV = Arb @ t = -2

3.5.2.2 Example of Inductive DFM Analysis

Besides the deductive analysis, inductive failure and fault analyses were executed for the example initiating event. These inductive analyses identified the progression of the system states from different combinations of initial component states.

For the failure and fault analysis example, to find out the event sequence as a result of the stuck BFV, the following initial condition was used**:**

At time 0, BFV = F-S and remains in the same state	AND
At time 0, CL = 0	AND
At time 0, CP = 0	AND
At time 0, Comp = OP and remains in the same state	AND
At time 0, ELP = 0	AND
At time 0, LP = 0	AND
At time 0, SbnP = 0	AND
At time 0, fSN = 1	AND
At time 1, fSN = 1	AND
At time 2, fSN = 1	AND
At time 3, fSN = 1	

This initial condition corresponds to the failure of the BFV in the stuck position while there is a mismatch between the steam flow and the feed flow (steam flow > feed flow). The inductive analysis engine was used to trace through the causality of the model. First, the states of the nodes included in the initial condition were used to determine the states of the nodes immediately downstream. After that, the states of these immediately downstream nodes were used to determine the states of the nodes further downstream. When the forward tracing was completed for 1 time step, the nodes were updated and the process was repeated for the next

time step. The complete set of intermediate steps for this inductive analysis was summarized in Table 3.5.19 to Table 3.5.29. In these tables, the columns in normal face represent the inputs to the transfer box/transition box in question, and the column in bold face represents the output for the same box. The first row indicates the time stamp associated with the input and output nodes. A time stamp of 0 indicates the initial time step, and it increases by 1 after a complete traversal of the loop. For example, Table 3.5.19 shows the result of tracing through transfer box Tf3 in Figure 3.5.1. Given the input states (from the initial condition) ELP = 0 and CL = 0, the decision table for Tf3 was consulted to determine that the output state is EL = 0. This newly derived state of EL, together with the states of the nodes Comp, BFV, SbnP and CP (defined in the initial condition) were used to determine the state of Sbn from the decision table associated with the transfer box Tf1. This step is summarized in Table 3.5.20. Next, the forward tracing progresses through transfer box Tf2. After this, the inductive analysis has traced through all the transfer boxes. This completes the forward tracing for time step 0. Table 3.5.22 to Table 3.5.26 shows the results of forward tracing through the transition boxes and update the states of the nodes for the next time step. In particular, Table 3.5.22 shows that Sbn = 0 @ time step 0 is equivalent to SbnP = 0 @ time step 1. The latter state was used in time step 1 to determine the new state of Sbn, as shown in Table 3.5.28. In summary, this inductive analysis showed that the BFV failure in the stuck position, together with an initial steam flow feed flow mismatch (steam flow > feed flow), will cause the steam generator level to drop from the normal state (L = 0), to the lowest state (L = -2) in 2 time steps. From LP = 0 @ time step 0 (equivalent to L = 0 @ time step -1) to L = -2 @ time step 1. The final state of the steam generator level is shown in Table 3.5.29.

Table 3.5.19 Forward Tracing through Transfer Box Tf3

Time	0	0	**0**
Node	ELP	CL	**EL**
State	0	0	**0**

Table 3.5.20 Forward Tracing through Transfer Box Tf1

Time	0	0	0	0	0	**0**
Node	Comp	BFV	SbnP	CP	EL	**Sbn**
State	OP	F-S	0	0	0	**0**

Table 3.5.21 Forward Tracing through Transfer Box Tf2

Time	0	0	0	**0**
Node	Sbn	fSN	LP	**L**

State	0	1	0	**-1**

Table 3.5.22 Forward Tracing through Transition Box Tt6

Time	0	**1**
Node	Sbn	**SbnP**
State	0	**0**

Table 3.5.23 Forward Tracing through Transition Box Tt10

Time	0	**1**
Node	fSN	**CP**
State	1	**1**

Table 3.5.24 Forward Tracing through Transition Box Tt9

Time	0	0	0	0	**1**
Node	L	fSN	Sbn	CL	**CL**
State	-1	1	0	0	**0**

Table 3.5.25 Forward Tracing through Transition Box Tt8

Time	0	**1**
Node	EL	**ELP**
State	0	**0**

Table 3.5.26 Forward Tracing through Transition Box Tt7

Time	0	**1**
Node	L	**LP**
State	-1	**-1**

Table 3.5.27 Forward Tracing through Transfer Box Tf3

Time	1	1	**1**
Node	ELP	CL	**EL**
State	0	0	**0**

Table 3.5.28 Forward Tracing through Transfer Box Tf1

Time	1	1	1	1	1	**1**
Node	Comp	BFV	SbnP	CP	EL	**Sbn**
State	OP	F-S	0	1	0	**0**

Table 3.5.29 Forward Tracing through Transfer Box Tf2

Time	1	1	1	**1**
Node	Sbn	fSN	LP	**L**
State	0	1	-1	**-2**

It is important to note that once a single DFM model is constructed, it can be analyzed for many different initial conditions. For example, the same DFM model can be analyzed for the initial condition:

At time 0, BFV = Frz and remains in this state AND
At time 0, CL = 0 AND
At time 0, CP = 0 AND
At time 0, Comp = OP and remains in this state AND
At time 0, ELP = 0 AND
At time 0, LP = 0 AND
At time 0, SbnP = 2 AND
At time 0, fSN = 1 AND
At time 1, fSN = 1 AND
At time 2, fSN = 1 AND
At time 3, fSN = 1

This initial condition corresponds to the failure of the BFV controller in the frozen state while there is a mismatch between the steam flow and the feed flow (steam flow < feed flow).

As in the previous inductive analysis example, the states of the nodes included in the initial condition were first used to determine the states of the nodes immediately downstream. After that, the states of these immediately downstream nodes were used to determine the states of the nodes further downstream. When the forward tracing was completed for 1 time step, the nodes were updated and the process was repeated for the next time step. The complete set of intermediate steps for this inductive analysis was summarized in Table 3.5.30 to Table 3.5.40. In summary, this inductive analysis showed that the BFV controller failure in the frozen state, together with an initial steam flow feed flow mismatch (feed flow > steam flow), will cause the steam generator level to rise from the normal state (L = 0), to the highest state (L = +2) in 2 time steps, from LP = 0 @ time step 0 (equivalent to L = 0 @ time step -1) to L = +2 @ time step 1. The final state of the steam generator level is shown in Table 3.5.40.

Table 3.5.30 Forward Tracing through Transfer Box Tf3

Time	0	0	**0**
Node	ELP	CL	**EL**
State	0	0	**0**

Table 3.5.31 Forward Tracing through Transfer Box Tf1

Time	0	0	0	0	0	**0**
Node	Comp	BFV	SbnP	CP	EL	**Sbn**
State	OP	Frz	2	0	0	**2**

Table 3.5.32 Forward Tracing through Transfer Box Tf2

Time	0	0	0	**0**
Node	Sbn	fSN	LP	**L**
State	2	1	0	**+1**

Table 3.5.33 Forward Tracing through Transition Box Tt6

Time	0	**1**

Node	Sbn	**SbnP**
State	2	**2**

Table 3.5.34 Forward Tracing through Transition Box Tt10

Time	0	**1**
Node	fSN	**CP**
State	1	**1**

Table 3.5.35 Forward Tracing through Transition Box Tt9

Time	0	0	0	0	**1**
Node	L	fSN	Sbn	CL	**CL**
State	+1	1	2	0	**0**

Table 3.5.36 Forward Tracing through Transition Box Tt8

Time	0	**1**
Node	EL	**ELP**
State	0	**0**

Table 3.5.37 Forward Tracing through Transition Box Tt7

Time	0	**1**
Node	L	**LP**
State	+1	**+1**

Table 3.5.38 Forward Tracing through Transfer Box Tf3

Time	1	1	**1**
Node	ELP	CL	**EL**
State	0	0	**0**

Table 3.5.39 Forward Tracing through Transfer Box Tf1

Time	1	1	1	1	1	**1**
Node	Comp	BFV	SbnP	CP	EL	**Sbn**
State	OP	Frz	2	1	0	**2**

Table 3.5.40 Forward Tracing through Transfer Box Tf2

Time	1	1	1	**1**
Node	Sbn	fSN	LP	**L**
State	2	1	+1	**+2**

4. MARKOV/CCMT METHODOLOGY

Notation

$P(t)$	Power
t	Time
x_n	SG_n Level (see Section 2.1)
V_j	Cells that partition the CVSS ($j=1,...,J$)
J	Total number of V_j
n	Component state combination index
N	Number of components state combinations
n_m	Component state index ($n_m=1,...,N_m$)
N_m	Total number of n_m
M	Number of components
E_{Ln}	SG_n level error (see Section 2.1)
C_{ln}	SG_n compensated level (see Section 2.1)
S_{Bn}	SG_n BFV position (see Section 2.1)
γ	Top Event
Γ	Number of Top Events
Δt	Cell-to-cell mapping time step
$\lambda_{n'm,nm},\mu_{n'm,nm}$	Transition rates from component state n'_m to n_m
$c_m(n_m\|n'_m,j'\rightarrow j)$	Pr{Component m is in state n_m at time $t=(k+1)\Delta t$ |Component m is in state n'_m at time $t=k\,\Delta t$ *and* controlled/monitored variables move from cell j' to cell j during $k\Delta t \le t \le (k+1)\Delta t$ }
$g(j\|j',n',k)$	Pr{Controlled/monitored variables are in cell j at time $t=(k+1)\Delta t$ |Controlled/monitored variables are in cell j' at time $t=k\Delta t$ }
$h(n\|n',j'\rightarrow j)$	Pr{Hardware/software/firmware in state n at time $t=(k+1)\Delta t$ |Hardware/software/firmware in state n' at time $t=k\,\Delta t$ *and* controlled/monitored variables move from cell j' to cell j during $k\Delta t \le t \le (k+1)\Delta t$ }
$p_{n,j}(k)$	Pr{Controlled variables are in cell j and hardware/software/firmware is in state n at time $t=k\Delta t$
$F_\gamma(k)$	Cdf of Top Event γ at time k
$w_{n,\gamma}(k)$	pdf of Top Event γ at time k
$q(n,j\|n',j',k)$	Elements of the transition matrix for the Markov chain

4.1 Example Initiating Event

As stated in Section 2.5, the example initiating event under consideration assumes the following:

1. Turbine trips
2. Reactor is automatically shutdown (tripped)
3. Power *P* is generated from the decay heat
4. Reactor power and steam flow rate reduce to 6.6% of 3000 MW_{th} 10 seconds after the turbine trip/reactor trip
5. Feedwater flow is at nominal level
6. Off-site power is available
7. Main Computer (MC) is failed

In high power mode, the pump demand of SG1 is a function also of the feedwater demand of SG2. Thus, from a modeling view point, the DFWCS of each of the two SG are coupled to each other. However, since the reactor is shutdown (Assumption 2 above), any thermodynamic variation in the secondary circuit (which contains the SGs, turbine, and condenser) and the primary circuit (which contains the reactor vessel, pumps, SGs, and the pressurizer) do not affect the power generated in the core due to decay heat (Assumption 3). Then by Assumption 4, power decays rapidly to 6.6% of the power before the shut down [125]. In particular, the power *P(t)* generated from a reactor core which operated previously at maximum power P_0 (3000 MW_{th}) for a time τ_s (in seconds) will be [125]

$$P(t) = 0.066P_0\left[t^{-0.2} - \left(t + \tau_s\right)^{-0.2}\right]$$

Since the SGs are decoupled from the primary system, the power generated *P(t)* will be equally discharged by the two SGs, and the power $P_n(t)$ transferred from the primary circuit to each SG will be

$$P_n(t) = 0.033P_0\left[t^{-0.2} - \left(t + \tau_s\right)^{-0.2}\right]$$

Following these assumptions, the system under consideration here involves only one SG with its own feedwater control system. Thus, the control system operates under the following conditions:

- FP fixed at minimum flow

- MFV closed
- Feedwater flow controlled by the BFV
- Measured quantities are power, level, and feedwater temperature (Fig.2.1.1).

In the reliability model construction of digital I&C systems using the Markov methodology, the system failure probability (i.e., the probability that Top Events are reached) is evaluated through a series of discrete transitions within the system state or the controlled variable state space (CVSS). These discrete transitions take into account:

1. the dynamic behavior of the system (i.e., mass and energy conservation laws),
2. the control laws, and
3. hardware/firmware/software states and their effect on the controlled/monitored process variables.

Items 1 and 2 above are modeled using the CCMT [137]. Item 3 uses the FMEA charts from Section 2.4.1 to define the hardware/firmware/software states (Section 4.2.3). It should be emphasized at this point that the purpose and implementation of the Markov methodology described in Section 4.2 and then illustrated in Section 4.3 using the example initiating event is quite distinct from the purpose and implementation of the Markov models in Section 2.4. Section 2.4 uses the Markov models to obtain a better understanding of the hardware/software/firmware structure of the system to plan for fault injection experiments for failure data generation. The Markov methodology of Section 4.2 utilizes the data generated by these fault injection experiments, as well as obtained from other databases (e.g., PRISM [118]) and CCMT to quantify the likelihood of Top Event occurrence.

4.2 The Markov Approach Coupled with CCMT: Markov/CCMT Methodology

The CCMT [137] is a systematic procedure to describe the dynamics of both linear and non-linear systems in discrete time and discretized system state space (or the subspace of the controlled variables only). The CCMT first requires a knowledge of the Top Events (Section 4.2.1) for the partitioning of the state space or the CVSS into V_j ($j=1,...,J$) cells (Section 4.2.2). The evolution of the system in discrete time is modeled and described through the probability $p_{n,j}(k)$ that the controlled variables are in a predefined region or cell V_j in the state space at time $t=k\Delta t$ ($k=0, 1,...$) with the system components (such as pumps, valves, or controllers) being in a component state combination $n=1,...,N$ (Section 4.2.5). The state combination represents the system configuration at a given time and contains information regarding the operational (or failure) status of each component (Section 4.2.3). Transitions between cells depend on (Section 4.2.4):

- the dynamic behavior of the system,

- the control laws, and
- the hardware/firmware/software states.

The dynamic behavior of the system is usually described by a set of differential or algebraic equations, as well as the set of control laws (Sections 2.2 and 2.5). The operating/failure states of each component are specified by the user. A system state n ($n=1...,N$) is an array of n_m $(m=1,..,M;\ n_m=1,...,N_m)$ variables each of which defines a state of component m. The direct coupling between operating/failure states of each component can be determined from the FMEA tables (Section 2.3). Both direct and indirect coupling between operating/failure states through the controlled/monitored process can be accounted for in the quantification of component state transitions (see Section 4.2.5).

The methodology is based on the following assumptions [6, 126]:

1. Components of the system (e.g., pumps, valves, or processors) do not change state during the time interval $[k,\ k+\Delta t)$ but possibly at $k+\Delta t$;
2. For a given component state combination n and cell V_j, $p_{n,j}(k)$ is uniformly distributed over V_j;
3. If the modeling is conducted in the CVSS, no two controlled variable trajectories arrive at the same point in state space at the same time and move in different directions for the same component state combinations.

Both Assumptions 1 and 2 lead to an approximation of the probabilistic system dynamics. Assumption 1 also leads to an approximation of the failure characteristics of the components. Assumption 2 reflects epistemic uncertainties in the deterministic system model (e.g., Eqs. (2.2.1) - (2.2.15) or Eqs.(2.5.1) - (2.5.8) for the benchmark system) as well as measurement uncertainties. Violation of Assumption 3 leads to the loss of the Markov property when following probabilistic system trajectories in the CVSS. However, Assumption 3 can be satisfied for most systems by choosing sufficiently refined partitioning of the CVSS (i.e., choosing adequately small V_j). The procedure to determine the Cdf (cumulative distribution function) and the pdf (probability distribution function) of each Top Event follows several steps [126]. These steps are explained in the following sections Section 4.2.1 to 4.2.6.

4.2.1 Definition of the Top Events

As indicated in Section 2.1, the purpose of the feedwater controller is to maintain the water level x_n (n=1,2) inside SGn optimally within ± 2 inches (with respect to some reference point) of the setpoint level (defined at 0 inches). The controller is regarded as failed if water level in SGn rises above +30 inches and falls below -24 inches (Section 2.1). Consequently, there are two Top Events:

1. x_n < -24 in (Low Level)

2. x_n > +30 in (High Level)

The cells that correspond to Top Events are modeled as absorbing cells or sink cells, i.e., the system can not move out of these cells and thus the transition probabilities from these cells to other cells in the state space are equal to 0.

4.2.2 Partitioning of the State Space or the CVSS into Computational Cells

The dynamics of the system are modeled as transitions between cells V_j *(j=1,...J)* that partition the state space or CVSS. For the example initiating event defined in Section 2.5, Eqs. (2.5.17) - (2.5.20) show that the state space is 4-dimensional and is comprised of

- level x_n
- level error E_{Ln}
- compensated level C_{ln}
- BFV position S_{Bn}

The partitioning needs to be performed in such a way that, other than cells V_j being disjoint and covering the whole space (definition of partitioning), values of the controlled variables defining the Top Events (in our case x_n) and the setpoints must fall on the boundary of the cells V_j and not within any cell. If this requirement is not satisfied for some $V_{j'}$, then the system state becomes ambiguous when the state variables are within $V_{j'}$ since the methodology assumes that $p_{n,j}(k)$ is uniformly distributed over $V_{j'}$ (Assumption 2 stated earlier in Section 4.2). Figure 4.2.1 shows the 3-dimensional projection of CVSS based on Eqs (2.5.17) - (2.5.20) in terms of level, level error, and BFV position. It should be also indicated that the choice of the discretization scheme for the CVSS and the time step (i.e., Δt) of the simulation are not independent of each other. For a given partitioning of the CVSS, too low a value of Δt could lead to the system being unable to exit the cell it is in at time t.

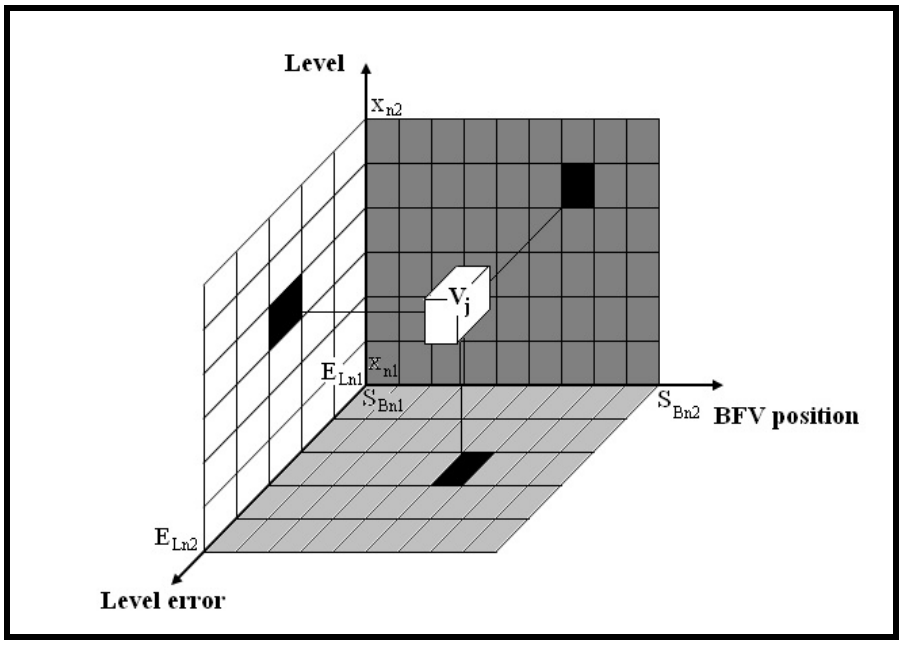

Figure 4.2.1 The CVSS for the Benchmark System based on Eqs (2.5.17) - (2.5.20)

4.2.3 Markov Modeling of Components

The construction of a Markov model for the components assumes that:

- a set of mutually exclusive and exhaustive states n_m *(m=1,...,M;* n_m*=1,...,*N_m*)* has already been defined for component *m*, i.e.,

$$\sum_{i=1}^{M} p(n_m) = 1$$
$$\forall n_m, n_{m'} \rightarrow n_m \cap n_{m'} = 0$$

- probability of transitions between states has been determined.

The choice of the failure states is based on the FMEA tables presented in Section 2.3. Regarding software common cause failure, software is not being treated separately but as embedded in hardware as indicated in Sections 1.2, 8, and throughout Section 2.4. In that respect, conventional common caused failure methods are applicable[138]. For example, platform commonality can be accounted for using the beta factor method if data are available for system failure due to platform based failure modes. Section 2.4 describes how data for the component state transitions can be obtained. Modeling of each benchmark system component (see Section 2.1) is presented in Sections 4.2.3.1 through 4.2.3.6. Section 4.2.3.7 shows how

the grouping of several components into macro-components can be useful to simplify modeling and reduce the computational effort. It should be noted that although the classical ET/FT approach also uses similar FMEA tables and the concept of macro-components, the Markov methodology has the capability to represent statistically dependent failure events while the classical ET/FT approach does not. The repair of components is not considered in this report. Mechanical failures(e.g. inadvertent FP trip) are modeled through the relevant controller failing high or low (see Section 4.2.3.5).

4.2.3.1 MFV and BFV

The behavior of MFV and BFV based upon the FMEA tables in Section 2.3.1 can be described by two states as shown in Fig. 4.2.2:

1. Valve operating correctly (Operating)
2. Valve stuck at the previous position (Stuck)

While it is possible for the valve to fail without reaching its target position after it is actuated, this situation is not considered due to lack of data. State 2 or the Stuck state partially accounts for such a situation.

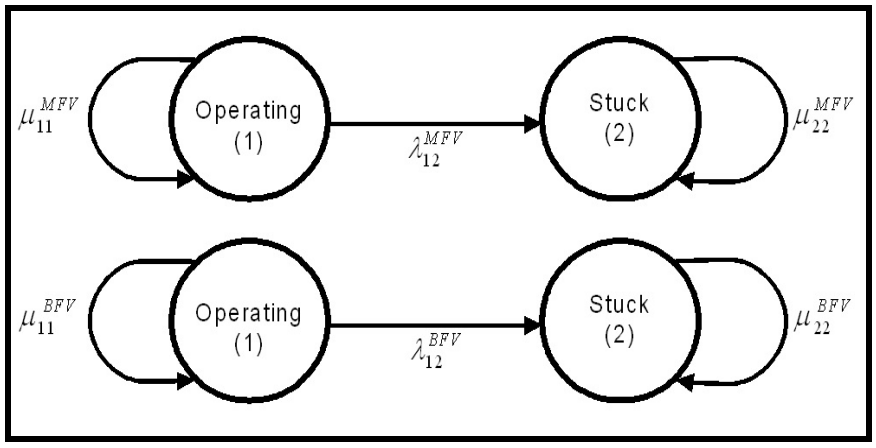

Figure 4.2.2 Failure States for the MFV and BFV

4.2.3.2 FP

From the FMEA table presented in Section 2.3.1, the description of the FP follows a configuration determined by two states as shown in Fig. 4.2.3:

1. FP operating
2. FP stuck at the previous speed

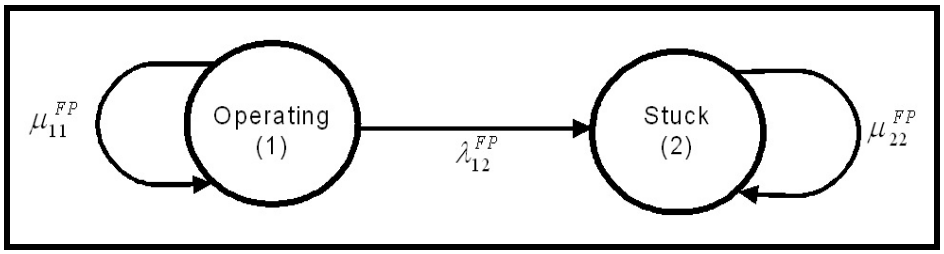

Figure 4.2.3 Failure States for the FP

The FP may fail to reach the determined speed after it is actuated. Again, this situation is not considered due to lack of data. State 2 (Stuck) partially accounts for such a situation.

4.2.3.3 Main (MC) and Backup Computers (BC)

Identification of failure states for these kinds of components strictly depend on their internal structure. In principle, a Markov model should be built for each of the constituent components (e.g., control unit, internal memory, register or buses) in such a way as to consider all the possible internal failure modes.

In this report each computer is considered as a single component and failure states are determined following the FMEA chart presented in Section 2.3.1. The failure states involve three elements (see Tables 2.3.1 and 2.3.2) :

1. Failure to read the data coming from the sensors
1. Failure in the processing of the data and internal errors
2. Failure in the communication between computers and controllers

Since the effects of these failures on the system are different if the computer is the main computer (MC) or the backup computer (BC) (see FMEA tables in Section 2.3.1), transitions between system configurations (i.e., transition from the general configuration *n* to *n'*) affect in different ways the cell-to-cell transitions. Thus, a separate Markov model for each computer will be considered.

The Markov modeling of each computer starts by introducing the two states which define:

- the computer which is operating correctly (Operating state), and
- the computer which is down and not sending any data to the controllers (Down state).

Element 1 involves an error in the communication between each pair of sensors for

each measured variable (e.g., SG level and FW flow) and the computer. The computers can detect the following types of communication failures (see Tables 2.3.1 and 2.3.2):

1. Loss of input from one sensor (i.e., the computer detects 0.0 vdc in the sensor reading)
1. Loss of input from both sensors
2. A value which is out of range for the measured variable or a physically impossible rate of change (intermittent failure)

If a loss of input failure occurs, the MC uses the old value of the measured variable or, in case the input signal does not return, the control system switches to the BC. However, if the loss of input affects the BC, then MFV, BFV, and FP freeze the output value to the actuated device.

In the Markov modeling of each computer the loss of input from one sensor and the intermittent failure of one sensor will be modeled together since the consequences are the same. Thus, two states will be considered as failure states of the Markov model for both the MC and BC:

- *Loss of one input*: Computer detects 0 vdc from one sensor or it recognizes that the output value of a sensor is out of range or physically impossible
- *Loss of both inputs*: Computer detects 0 vdc from both sensors or it recognizes that the output values of both sensors are out of range or physically impossible

From these failure states, it is possible to return to the Operating state if the failure is recovered. Otherwise the computer goes to the Down state.

Element 2 involves errors during the processing of the data received from the sensors. We consider the following errors (see Tables 2.3.1 and 2.3.2):

- Roundoff/truncation/sampling rate errors
- Failure in the response requirements
- Failure of the watchdog timer

As described in Tables 2.3.1 and 2.3.2, the computer takes itself down as a consequence of these failures. In that respect, these failure modes are modeled as direct transitions of the computer to the Down state.

Finally, Element 3 involves errors in the communication between the computers and the MFV/BFV/FP controllers:

1. Failure to send data to the controller (i.e., loss of one output).
2. Controller output different from the computer output

3. Arbitrary value output

Error 2, if detected, is regarded as transition to the computer Down state (see Tables 2.3.1 and 2.3.2). If not detected, it is regarded as Error 3. In this state, the computer is sending random data to the controllers.

The loss of power failure for the computer (see Tables 2.3.1 and 2.3.2) results in the shutting down of the computer itself. Thus, this failure is implemented as a direct transition to the Down state of the computer.

In this report, setpoint drift is not considered since the level setpoint of each steam generator is not affected by any other component of the benchmark system.

Thus, from the FMEA chart for the computers, a five-state chain has been identified for each computer (see Fig. 4.2.4 and Fig. 4.2.5) :

1. Computer operating
2. Loss of one input
3. Loss of both inputs
4. Arbitrary value
5. Computer down

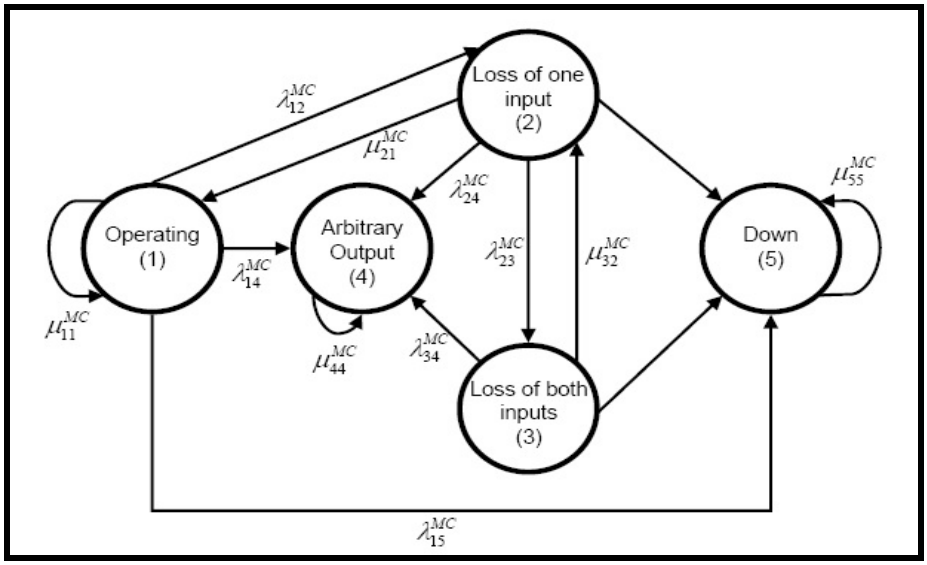

Figure 4.2.4 Failure States for the Main Computer (MC)

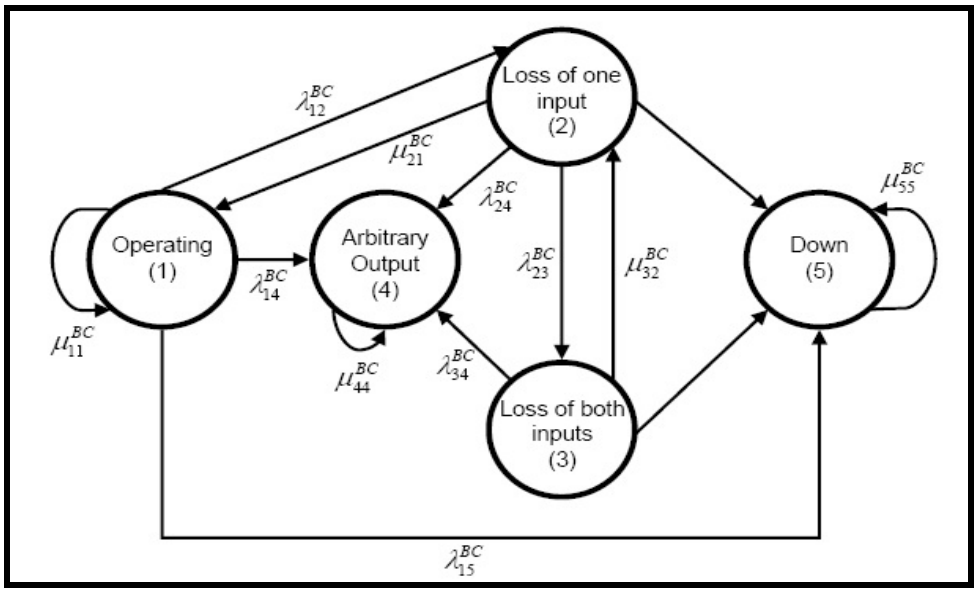

Figure 4.2.5 Failure States for the Backup Computer

The BC model that is used for the example initiating event (see Section 2.5) presents fewer number of states. Following Assumptions 1, 2, and 3 of Section 2.5.1 these states are the following:

- Computer operating
- Loss of both inputs
- Computer down

The Markov model of the computer hardware/software/firmware transitions for the example initiating event is presented in Fig. 4.2.6.

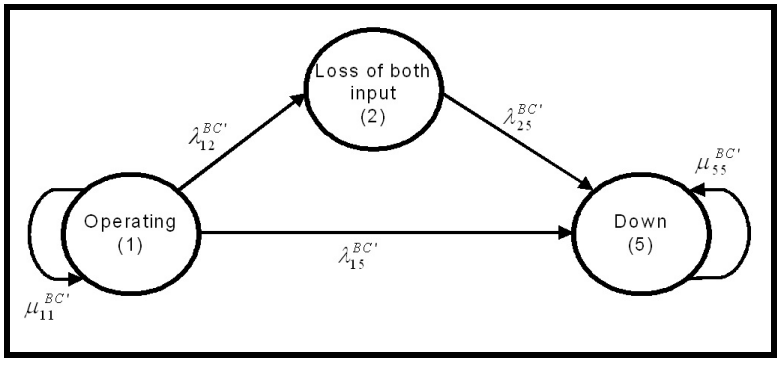

Figure 4.2.6 Failure states for the Example Initiating Event of the BC

4.2.3.4 Sensors

The control system uses several sets of sensors to measure different parameters (i.e., level, feedwater temperature, and power). Each of these parameters is measured by a pair of sensors and the final value of the parameters themselves is obtained averaging the two measured values. The information generated by these sensors is sent directly to both MC and BC. From the FMEA charts presented in Section 2.3.1, two failure modes involve sensors:

1. *Loss of output*: The sensor is not sending any data to the computer
1. *Out of range*: Value received by the computer from the sensor is out of range or it has an impossible rate of change

Since the FMEA charts presented in Section 2.3 regard sensor failure as a failure mode of the components sensor data are being transmitted to, sensor failures are not explicitly modeled, but rather directly implemented in the computer Markov model presented in Section 4.2.3.3 as loss of input or arbitrary input. Figure 4.2.7 shows the state transition diagram for the sensors.

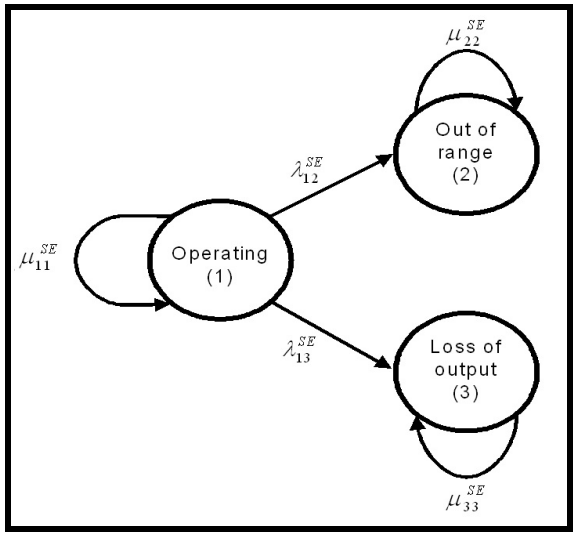

Figure 4.2.7 Failure States for the Sensors

4.2.3.5 FP, MFV and BFV Controllers

Signals generated from the MC and BC travel to the pump and valves through controllers which communicate continuously with the computers. The possible failures are (see Section 2.3.2):

1. Loss of power

2. Erroneous communication between controllers and computer
3. Loss of output to MFV/BFV/FP
4. High value output
5. Low value output
6. Arbitrary value output

The modeling of the controller starts by defining the Operating state in which the controller is receiving data from the computer and is able to send data to the actuated device (i.e., BFV, MFV, or FP). Upon a loss of input from both computers (here it is assumed that a loss of output of the computer is logically equivalent to a loss of input of the controller), the controller maintains as output the old valid value to the actuated device (i.e., the controller “freezes” the output).

Erroneous communication between controllers and the computer that lead to high, low, or arbitrary output are considered equivalent to Failures 4, 5, and 6, respectively. Loss of communications between the controller and the actuated device or loss of power are modeled as 0 voltage on the line which connect these two components.

In summary, five states have been identified for each of the MFV, BFV, and FP controllers (see Fig. 4.2.8 for the BFV controller; the others are similar):

1. Controller is in operating state and is communicating with the actuated device
2. Controller is in operating state and is not communicating with the actuated device
3. Freeze
4. Output high
5. Output low
6. Arbitrary output
7. 0 vdc output

States 4 and 5 are also modeling mechanical failures such as inadvertent device actuation and inadvertent disengagement (e.g. FP inadvertently turns on or trips).

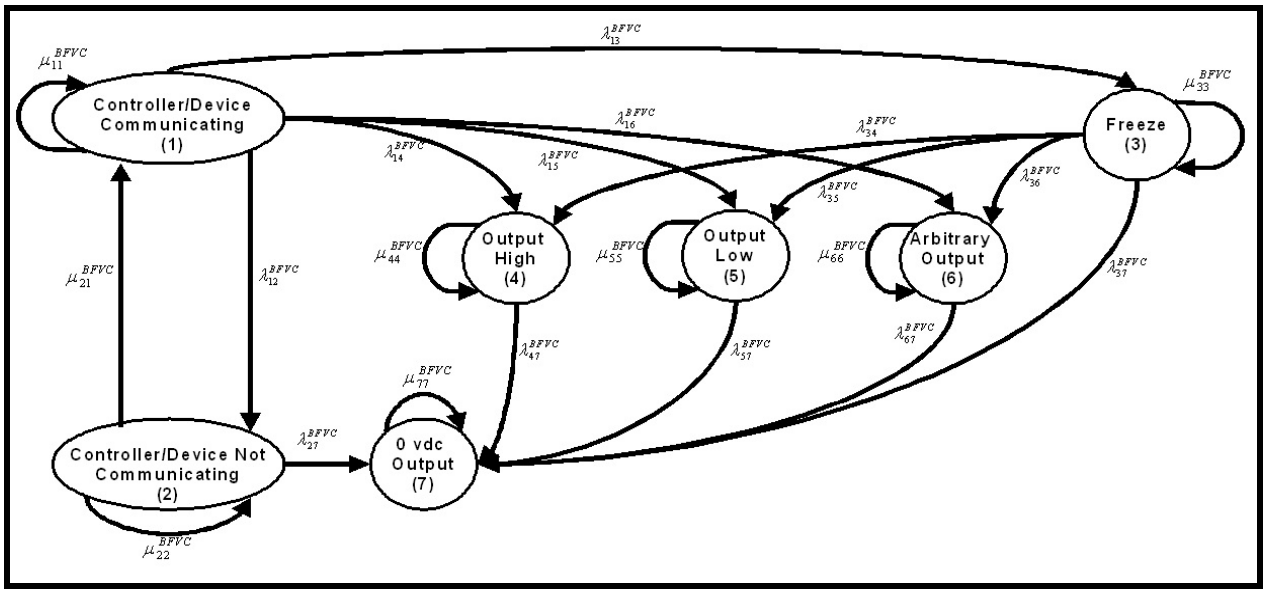

Figure 4.2.8 Failure States for the BFV Controller

4.2.3.6 PDI Controller

The benchmark DFWCS also contains a PDI controller (Section 2.1) which is used as backup to the MFV controller in case a loss of output of the MFV controller occurs. In this case, the PDI controller outputs the old valid value to the MFV. Following the FMEA chart of Section 2.3.2, it is possible to identify the following failure modes for the PDI controller:

1. Loss of inputs
2. Loss of power
3. Loss of outputs
4. Arbitrary failure

The modeling of the PDI controller (see Fig. 4.2.9) starts by introducing the state in which the PDI controller is operating correctly. Secondarily, the loss of inputs and Arbitrary failure define the other two states where the PDI controller send the last valid value or random data to the MFV.

As implemented in Section 4.2.3.5, a loss of communication between the PDI controller and the MFV controller can be recovered. The PDI controller can recover communications between MFV and itself. On the contrary, the Arbitrary failure lead to a sink state where the PDI controller is sending random data to the MFV controller. For the Markov modeling of the PDI controller this situation implies:

- Two-way connection between Operating state and Loss of Inputs state
- One-way connection between Operating state and Arbitrary Failure state

- One-way connection between Loss of inputs state and Arbitrary Failure state

Again, the PDI controller and MFV can recover communications between MFV controller and itself. This requires three additional states (i.e., Operating, Loss of inputs, and Arbitrary failure), which describe the state of the controller when MFV and PDI controller are not communicating (Fig. 4.2.9). Figure 4.2.9 also assumes that two events cannot occur at the same time in the spirit of Markov models. In that respect, connections between the two sets of states exist only among the similar PDI controller states.

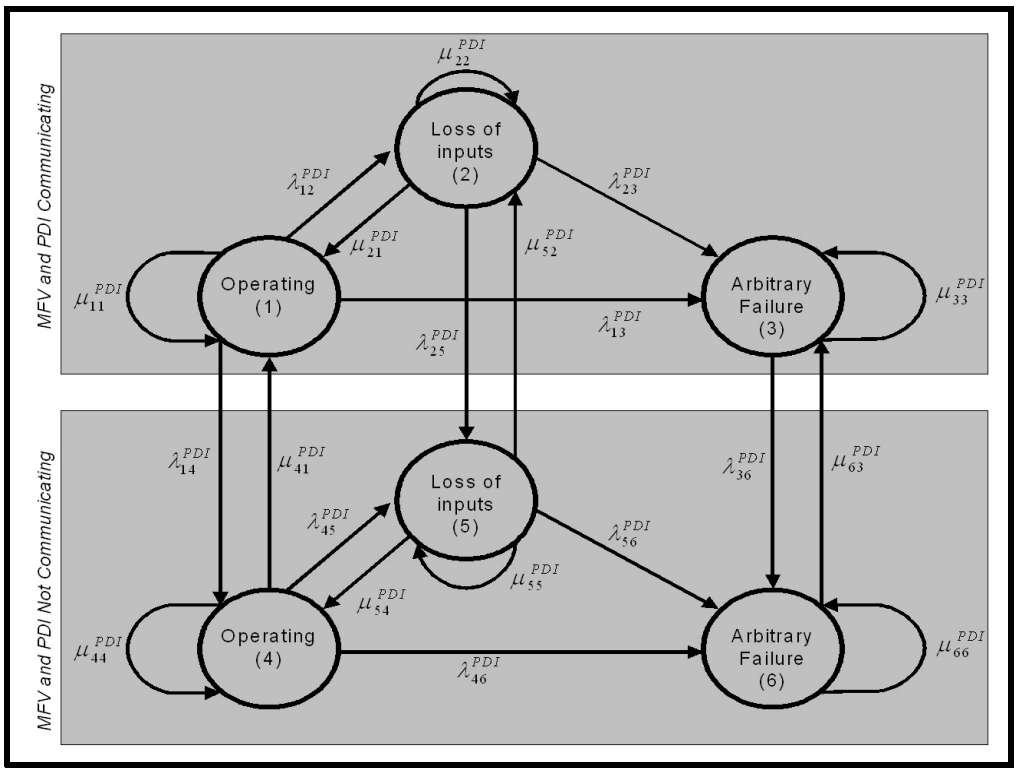

Figure 4.2.9 Failure States for the PDI Controller

4.2.3.7 System State Reduction Through Macro-Components

In summary, the benchmark system consists of the following M =14 components and N = 3^5x2x2^2x7^3x6x5^2= 100,018,800 combination of component states (equal to the number of system states):

- 5 sensors (3 states each)
- 1 pump (2 states)

- 2 valves (2 states each)
- 3 controllers, one for the pump and one for each valve (7 states each)
- 1 PDI controller (6 states)
- 2 computers, BC and MC (5 states each)

Each combination can be represented as an array with *M* = 14 columns where each column corresponds to a specific component listed above. Table 4.2.1 shows some example of these combinations.

Table 4.2.1 Examples of State Combinations (m=1,...,M; n=1,...,N)

FP (Fig. 4.2.2)	*MFV* (Fig. 4.2.3)	*BFV* (Fig. 4.2.3)	*MFV Controller* (Fig. 4.2.8)	*BFV Controller* (Fig. 4.2.8)	*FP Controller* (Fig. 4.2.8)	*MC* (Fig. 4.2.4)	*BC* (Fig. 4.2.5)	*PDI* (Fig. 4.2.9)	*Level sensor* (Fig. 4.2.7)	*Power sensor* (Fig. 4.2.7)	*FW flow sensor* (Fig. 4.2.7)	*Steam Flow sensor* (Fig. 4.2.7)	*FW temp sensor* (Fig. 4.2.7)	*m /n*
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	2	2
1	1	1	1	1	1	1	1	1	1	1	1	1	3	3
1	1	1	1	1	1	1	1	1	1	1	1	2	1	4
1	1	1	1	1	1	1	1	1	1	1	1	2	2	5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

It is important to note that the computational costs are proportional to the number of state combinations. Therefore, a reduction in the number of combinations can have substantial computational benefits. The following considerations can be made towards this objective:

- Sensor failure states can be merged into the computer models since both the MC and BC can recognize physically impossible data from the sensors. In case MC and BC don't recognize these failures of the sensors, they can be regarded as being in the Arbitrary Output state.
- Since there is a direct connection between BFV and FP and their controllers, it is possible to merge the actuated devices with their controllers. The Markov model for this kind of macro-components is represented in Fig. 4.2.10. The macro-component consists of the BFV and the BFV controller and transition rates take into account both the BFV and the BFV controller behavior.

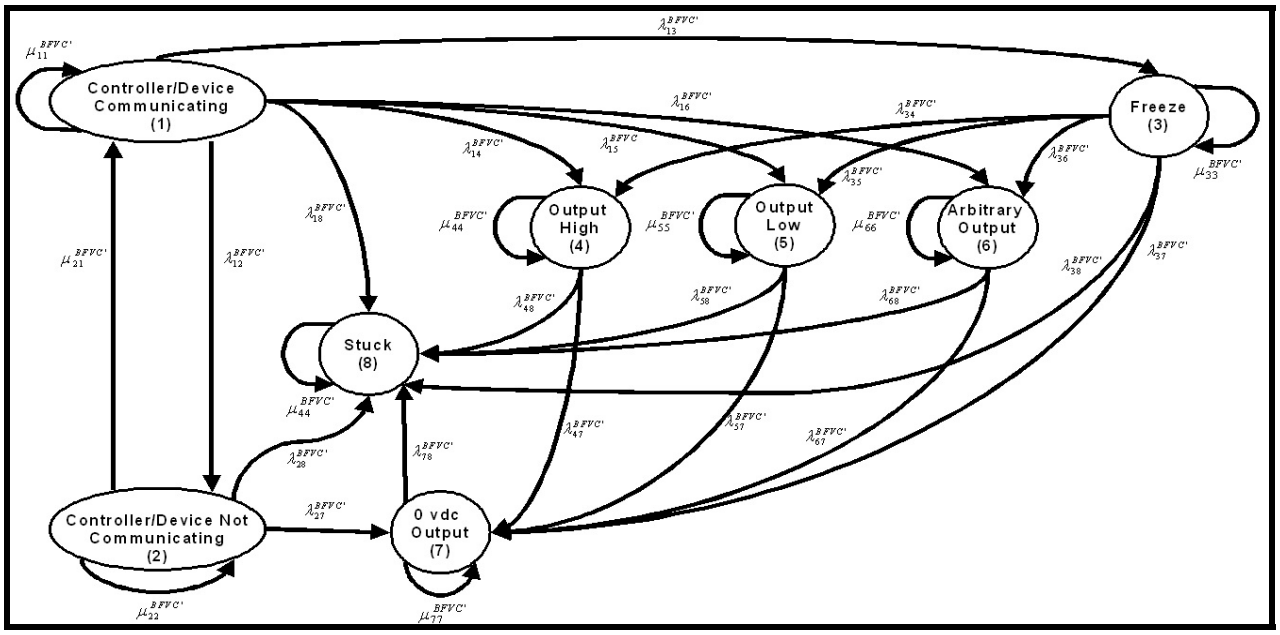

Figure 4.2.10 Failure States for the Combined BFV and BFV Controller

As a result, the number of components can be reduced to *M* = 6 as follows:

- 3 controllers, one for the pump and one for each valve (8 states each)
- 1 PDI controller (6 states)
- 2 computers, MC and BC (5 states each)

Thus, the final number of component state combinations is reduced to *N* = 5x5x6x8x8x8=76,800 states. Table 4.2.2 shows examples of ordering of state combinations after grouping (m=1,...,M; n=1,...,N).

Table 4.2.2 Examples of Ordering of State Combinations After Grouping (m=1,...,M; n=1,...,N)

MFV+ controller (Fig.4.2.10)	*BFV+ controller (Fig.4.2.10)*	*FP+ controller (Fig.4.2.10)*	*PDI (Fig.4.2.9)*	*BC (Fig.4.2.5)*	*MC (Fig.4.2.4)*	*m/n*
1	1	1	1	1	1	1
1	1	1	1	1	2	2
1	1	1	1	1	3	3

1	1	1	1	1	4	4
1	1	1	1	1	5	5
1	1	1	1	2	1	6
...	...	...	...	...	...	...

4.2.4 Determination of the Cell-to-Cell Transition Probabilities

As indicated earlier in Section 4.2, the dynamic evolution of the system depends on:

- the dynamic equations of the system,
- the control laws of the control system, and
- the state of each component.

Consequently, the probability of the system to transit from a cell $V_{j'}$ to cell V_j also depends on these three factors. In the Markov/CCMT methodology, the first two factors are accounted for in the transition probability $g(j|j',n',k)$ while the third one is captured by the transition probability $h(n|n',j' \rightarrow j)$.

The cell-to-cell transition probabilities $g(j|j',n',k)$ are conditional probabilities that the controlled variables are in the cell V_j at time $t = (k+1)\Delta t$ given that:

- the controlled variables are in the cell $V_{j'}$ at time $t = k\Delta t$, and,
- the system components are in component state combination $n(k) = n'$ at time t.

It can be shown that [5] the $g(j|j',n',k)$ can be found from

$$g(j|j',n',k) = \frac{1}{v_{j'}} \int_{V_{j'}} dv' \, e_j\{\tilde{x}_{k+1}(x',n',k)\}$$

$$e_j\{\tilde{x}_k\} = \begin{cases} 1 \rightarrow \tilde{x}_k \in V_j \\ 0 \rightarrow otherwise \end{cases}$$

where:

- $v_{j'}$ is the volume of the cell $V_{j'}$

- $\tilde{x}_k$ is the arrival point in the state space/CVSS at time $t = (k+1)\Delta t$
- x' is the starting point in the cell $V_{j'}$ at time $t = k\Delta t$
- n' is the component state combination at time $t = k\Delta t$.

For the example initiating event in Section 2.5, the integral in Eq. (4.2.2) can be evaluated by the following quadrature scheme:

- Partition a cell j' in Fig. 4.2.1 into N_p subcells.
- Choose the midpoint of each subcell as initial conditions of Eqs. (2.5.17) - (2.5.19), integrate Eqs. (2.5.17) - (2.5.19) over the time interval $k\Delta t \le t \le (k+1)\Delta t$ under the assumption that the component state combination remains n' at all times during $k\Delta t \le t \le (k+1)\Delta t$.
- Observe the number of arrivals in N_{p+1} at time $t = k\Delta t$ (i.e., $\tilde{x}_{k+1}(x', n', k)$).
- Obtain $g(j|j',n',k)$ as $g(j|j',n',k) = N_p / N_{p+1}$.

4.2.5 Determination of the Component State Transition Probabilities

The stochastic behavior of hardware/software/firmware is represented through $h(n|n',j'\to j)$, which is the probability that the component state combination at time $t = (k+1)\Delta t$ is n, given that:

1. $n(k) = n'$ at $t = k\Delta t$, and
2. the controlled variables transit from cell $V_{j'}$ to cell V_j during $k\Delta t \le t < (k+1)\Delta t$.

For components with statistically independent failures, the probabilities $h(n|n',j'\to j)$ are the products of the individual component failure or non-failure probabilities during the mapping time step from $k\Delta t$ to $(k+1)\Delta t$, i.e.,

$$h(n|n', j' \to j) = \prod_{m=1}^{M} c_m(n_m|n'_m, j' \to j)$$

where $c_m(n_m|n'_m,j'\to j)$ is the transition probability for component m from the combination n'_m to n_m within $[k\Delta t, (k+1)\Delta t]$ during the transition from the cell $V_{j'}$ to V_j.

As an example, suppose that m in Eq. (4.2.3) corresponds to the combined BFV and BFV controller (see Fig. 4.2.10) and that the transition from the configuration n'_m to n_m involves the transition from the “Controller/Device Communicating” state to the “Stuck” state. Then from Fig.4.2.10 the value of $c_m(n_m|n'_m,j' \to j)$ is equal to $\lambda^{BFV}_{18}\Delta t$.

The probability $p_{n,j}(k+1)$ that at $t = (k+1)\Delta t$ the controlled variables are in cell V_j and the component state combination is n is the sum of $N \times J$ terms where each of these includes two factors:

- the probability for the system to transit from the cell $V_{j'}$ and component state combination n' to cell V_j and component state combination n (i.e., $q(n,j|n',j',k)$) and
- the probability that the system is in the initial cell $V_{j'}$ and state combination n' (i.e., $p_{n,j}(k)$).

In general:

$$p_{n,j}(k+1) = \sum_{n'=1}^{N} \sum_{j'=1}^{J} q(n, j | n', j', k) p_{n',j'}(k)$$

The elements of the transition matrix $q(n,j|n',j',k)$ are functions of both

- the cell to cell transition probability $g(j|j',n',k)$ (see Section 4.2.4), and
- the component state transition probabilities $h(n|n',j',j)$ (see Section 4.2.5).

Thus

$$q(n, j | n', j', k) = g(j | n', j', k) h(n | n', j' \to j)$$

Since cells V_j cover the whole CVSS and N includes all the possible state combinations:

$$\sum_{n=1}^{N} \sum_{j=1}^{J} p_{n,j}(k) = 1$$

$$\sum_{n=1}^{N} \sum_{j=1}^{J} q(n, j | n', j', k) = 1$$

The grouping of several components into macro-components (as seen in Section 4.2.3) can be useful to decrease the number of possible state combinations, which can be very large for systems that involve a large number of components. Note that for autonomous processes the transition matrix $q(n,j|n',j',k)$ has to be constructed only once and not at each step throughout the duration of the mission of the system.

4.2.6 Determination of the pdf and Cdf for the Top Events

After computing $p_{n,j}(k)$ at the end of each time step k, the Cdf $F_\gamma(k)$ and pdf $w_{n,\gamma}(k)$ for the Top Event γ ($\gamma = 1,..., \Gamma$) can be calculated as presented in Eq. (4.2.7):

$$F_\gamma(k) = \sum_{n=1}^{N} p_{n,\gamma}(k)$$

$$w_{n,\gamma}(k) = \frac{1}{\Delta t}\left[p_{n,\gamma}(k+1) - p_{n,\gamma}(k)\right]$$

Each Top Event γ is a set of cells in the CVSS (see Section 4.2.1). Statistical importance of hardware/software/firmware configuration n to failure event γ at time $t = k\Delta t$ can be determined from [6]:

$$(\mathrm{Im})_{n,\gamma}(k) = \frac{\sum_{n \in S_n} w_{n,\gamma}(k)}{\sum_{n=1}^{N} w_{n,\gamma}(k)}$$

where S_n is the set of system states containing the hardware/software/firmware configuration n.

4.3 Implementation with the Example Initiating Event

In order to illustrate the methodology presented in Section 4.2, the example initiating event presented in Section 2.5 will be used. As indicated in Section 4.2.2, the CVSS of the system consists of level, level error change, compensated level, and BFV position, and the benchmark system is reduced essentially to only two components (M = 2), i.e., the BC and the BFV valve.

4.3.1 Definition of the Top Events

The two Top Events are:

1. x_n < -2.0 feet (Low Level)
2. x_n > +2.5 feet (High Level)

4.3.2 Partitioning of the CVSS

The CVSS consists of 4 variables:

- level x
- level error E_L
- compensated level C_L
- BFV position S_B

For illustration purposes, each of these variables has been partitioned in three intervals. Then the CVSS consists of 3^4=81 cells plus the two cells which represent the Top Events. The partition of the four variables are presented in Table 4.3.1. The partition follows the guidelines given in Section 4.2.2.

Table 4.3.1 Partitioning Scheme of the 4 Variables of the CVSS

Interval for x	*Value*
0	-2.0 <= x < -0.17
1	-0.17 <= x < 0.17
2	0.17 <= x <= 2.5
Low	x < -2.0
High	x > 2.5

Interval for E_L	*Value*
0	-1000 <= E_L < -1.587
1	-1.587 <= E_L < 4.203
2	4.203 <= E_L <= 1000

Interval for C_L	*Value*
0	-500 <= C_L < -100
1	-100 <= C_L < 100
2	100 <= C_L <= 5000

Interval for S_B	*Value*
0	0 <= S_B < 30
1	30 <= S_B < 70
2	70 <= S_B <= 100

4.3.3 Markov Modeling of the Components and the Determination of the Elements *h(n|n',j'->j)*

The Markov models for the BC and the combined BFV-BFV controller are presented in Section 4.2. As presented in Section 2.5, several simplifications occur for both components for the example initiating event. For this event, the BC can be represented by 3 states instead of 5 (see Section 4.2.2.3) which are:

- BC Operating
- Loss of both inputs
- BC Down

For the combined BFV-BFV the 8 states presented in Fig. 4.2.10 have been reduced to 5 (see Section 4.2.3.7):

- Controller operating and communicating correctly with the BFV
- Freeze
- 0 vdc output
- Arbitrary output
- Stuck

These states are shown in Fig. 4.3.1.

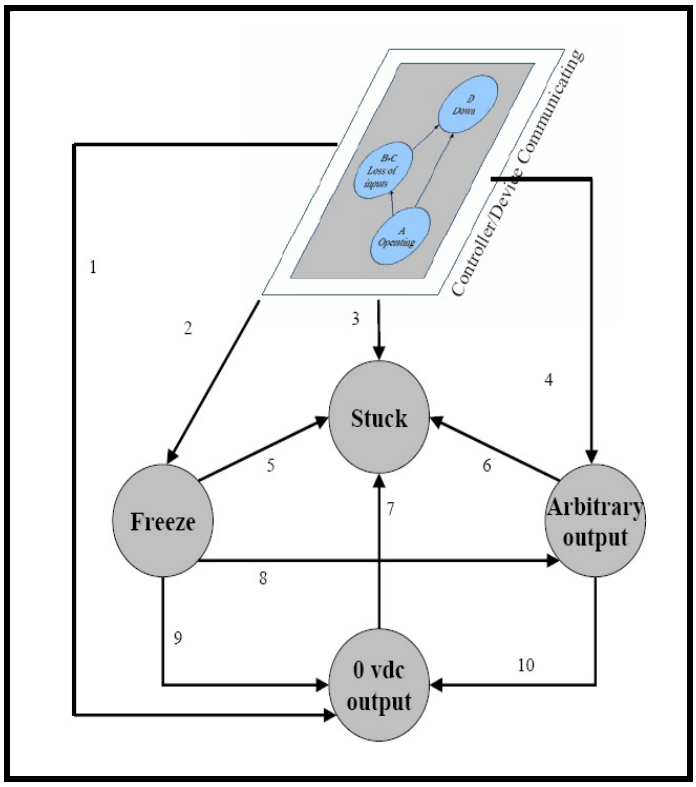

Figure 4.3.1 Markov Modeling of the Example Initiating Event

Thus, the total number *N* of component state combinations is 7 and each state combination is determined by an array of 2 elements (one element for each component) as presented in Table 4.3.2.

Table 4.3.2 Component State Combinations for the Example Initiating Event

BFV Controller	***BC***	***n***
Communicating	Operating	1
Communicating.	Loss of input	2
Communicating	Down	3
Freeze	-	4
Arbitrary output	-	5
0 vdc output	-	6
Stuck	-	7

Transitions between state combinations are shown in Table 4.3.3, which contains the elements *h(n|n',j'->j)*.

Table 4.3.3 Allowed Component States Combination Transitions

n'\n	1	2	3	4	5	6	7
1	$\mu_{11}\Delta t$	$\lambda_{12}\Delta t$	$\lambda_{13}\Delta t$	$\lambda_{14}\Delta t$	$\lambda_{15}\Delta t$	$\lambda_{16}\Delta t$	$\lambda_{17}\Delta t$
2	0	0	$\lambda_{23}\Delta t$	$\lambda_{24}\Delta t$	$\lambda_{25}\Delta t$	$\lambda_{26}\Delta t$	$\lambda_{27}\Delta t$
3	0	0	0	$\lambda_{34}\Delta t$	$\lambda_{35}\Delta t$	$\lambda_{36}\Delta t$	$\lambda_{37}\Delta t$
4	0	0	0	$\mu_{44}\Delta t$	$\lambda_{45}\Delta t$	$\lambda_{46}\Delta t$	$\lambda_{47}\Delta t$
5	0	0	0	0	$\mu_{55}\Delta t$	$\lambda_{56}\Delta t$	$\lambda_{57}\Delta t$
6	0	0	0	0	0	$\mu_{66}\Delta t$	$\lambda_{65}\Delta t$
7	0	0	0	0	0	0	$\mu_{77}\Delta t$

4.3.4 Determination of the Cell-to-Cell Transition Probabilities

The determination of the cell-to-cell transition probabilities *g(j|n',j',k)* takes into account the dynamic laws presented in Section 2.5, which describe the process and the state of each component at time $t = k\Delta t$ or $t = (k-1)\Delta t$. As presented in the FMEA analysis in Section 2, in case a loss of input or a loss of output occurs, the last value of the BFV position must be taken into account. In this respect, Table 4.3.4 shows which value of the BFV position must to be taken into account for each of the 7 component state combinations (see also Table 2.5.3). The possible values are:

- the current value determined by the set of equations presented in Section 2 (CV),
- the old value determined in the previous time step (OV), or,
- a value that is not possible to predict (i.e., any value) due to a particular component state combination.

Table 4.3.4 BFV Position for Different Component State Combinations

n	***BFV***	***BC***	***BFV Position***
1	Communicating	Operating	CV
2	Communicating	Loss of input	OV
3	Communicating	Down	OV
4	Freeze	-	OV
5	Arbitrary output	-	Any

6	0 vdc output	-	Closed
7	Stuck	-	OV

Note: OV - BFV position determined in the previous time step

CV - BFV position determined from the set of equation presented in Section 2

Any - BFV position can be any value (in the 0-100 range).

It is important to note that, for the example initiating event, the transition between cells depends on both the starting cell and the component state combination. However, a coupling in the opposite direction (i.e., a dependence of the component state combinations on the CVSS cells) is not relevant for this example initiating event since the failure rates are not assumed to be affected by controlled process variables and there are no on-off controllers whose actuation depends on setpoint crossings. The setpoint dependence of the control process is described by Eq. (2.5.2) or Eq. (2.5.10).

Figure 4.3.2 shows a small portion of the overall matrix which contains the elements *g(j|n',j',k)*. The first column defines the components state combination *n'* while the second one defines the cell $V_{j'}$ the system is in at time *t*. The first row of the rest of the columns indicates the cell V_j the system is in at time $t+\Delta t$. Each cell V_j and $V_{j'}$ is represented as an array of four elements (i.e., one for each variable of the CVSS), where each element contains an integer number corresponding to the intervals of the continuous process variables as defined in Table 4.3.1.

System Config.	From/To	0-0-0-0	1-0-0-0	2-0-0-0	0-1-0-0	1-1-0-0	2-1-0-0	0-2-0-0	1-2-0-0	2-2-0-0	0-0-1-0	1-0-1-0	2-0-1-0	0-1-1-0
OK-OK	0-0-0-0	0.67	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	1-0-0-0	1	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	2-0-0-0	0.33	0	0.67	0	0	0	0	0	0	0	0	0	0
OK-OK	0-1-0-0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	1-1-0-0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	2-1-0-0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	0-2-0-0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	1-2-0-0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	2-2-0-0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	0-0-1-0	0.22	0	0	0	0	0	0	0	0	0.44	0	0	0
OK-OK	1-0-1-0	0.33	0	0	0	0	0	0	0	0	0.67	0	0	0
OK-OK	2-0-1-0	0.11	0	0.22	0	0	0	0	0	0	0.22	0	0.44	0
OK-OK	0-1-1-0	0	0	0	0	0	0	0	0	0	0.07	0	0	0.15
OK-OK	1-1-1-0	0	0	0	0	0	0	0	0	0	0.11	0	0	0.22
OK-OK	2-1-1-0	0	0	0	0	0	0	0	0	0	0.04	0	0.07	0.04
OK-OK	0-2-1-0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	1-2-1-0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	2-2-1-0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	0-0-2-0	0	0	0	0	0	0	0	0	0	0.22	0	0	0
OK-OK	1-0-2-0	0	0	0	0	0	0	0	0	0	0.33	0	0	0
OK-OK	2-0-2-0	0	0	0	0	0	0	0	0	0	0.11	0	0.22	0
OK-OK	0-1-2-0	0	0	0	0	0	0	0	0	0	0.22	0	0	0
OK-OK	1-1-2-0	0	0	0	0	0	0	0	0	0	0.33	0	0	0
OK-OK	2-1-2-0	0	0	0	0	0	0	0	0	0	0.07	0.04	0.22	0
OK-OK	0-2-2-0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	1-2-2-0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	2-2-2-0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4.3.2 Small Portion of the Matrix which Contains the Elements *g(j|n',j',k)*

4.3.5 Determination of the Elements q(n,j|n',j',k)

As presented in Section 4.2.5, the product of the terms *g(j|n',j',k)* and *h(n|n',j'->j)* gives the transition probabilities *q(n,j|n',j',k)*. Figure 4.3.3 shows a small portion of the matrix containing the elements *q(n,j|n',j',k).* Since we have:

- 7 components state combinations
- 81 cells of the CVSS
- 2 cells which represent the Top Events

the overall matrix has 7*81+2= 569X569 elements.

	System Config.	OK-OK	OK-OK	OK-OK	OK-OK	OK-OK	OK-OK	OK-OK	OK-OK	OK-OK	OK-OK	OK-OK	OK-OK	OK-OK	OK-OK	OK-OK
System Config.	From/To	0-0-0-0	1-0-0-0	2-0-0-0	0-1-0-0	1-1-0-0	2-1-0-0	0-2-0-0	1-2-0-0	2-2-0-0	0-0-1-0	1-0-1-0	2-0-1-0	0-1-1-0	1-1-1-0	2-1-1-0
OK-OK	0-0-0-0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	1-0-0-0	0.14	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	2-0-0-0	0.05	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	0-1-0-0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	1-1-0-0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	2-1-0-0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	0-2-0-0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	1-2-0-0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	2-2-0-0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	0-0-1-0	0.03	0	0	0	0	0	0	0	0	0.06	0	0	0	0	0
OK-OK	1-0-1-0	0.05	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0
OK-OK	2-0-1-0	0.02	0	0.03	0	0	0	0	0	0	0.03	0	0.06	0	0	0
OK-OK	0-1-1-0	0	0	0	0	0	0	0	0	0	0.01	0	0	0.02	0	0
OK-OK	1-1-1-0	0	0	0	0	0	0	0	0	0	0.02	0	0	0.03	0	0
OK-OK	2-1-1-0	0	0	0	0	0	0	0	0	0	0.01	0	0.01	0.01	0.01	0.02
OK-OK	0-2-1-0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	1-2-1-0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	2-2-1-0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	0-0-2-0	0	0	0	0	0	0	0	0	0	0.03	0	0	0	0	0
OK-OK	1-0-2-0	0	0	0	0	0	0	0	0	0	0.05	0	0	0	0	0
OK-OK	2-0-2-0	0	0	0	0	0	0	0	0	0	0.02	0	0.03	0	0	0
OK-OK	0-1-2-0	0	0	0	0	0	0	0	0	0	0.03	0	0	0	0	0
OK-OK	1-1-2-0	0	0	0	0	0	0	0	0	0	0.05	0	0	0	0	0
OK-OK	2-1-2-0	0	0	0	0	0	0	0	0	0	0.01	0.01	0.03	0	0	0
OK-OK	0-2-2-0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	1-2-2-0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	2-2-2-0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	0-0-0-1	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	1-0-0-1	0.14	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	2-0-0-1	0.05	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	0-1-0-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OK-OK	1-1-0-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4.3.3 Small Portion of the Matrix which Contains the Elements *q(n,j|n'j',k)*

It is important to note that the matrix presented in Fig. 4.3.3 is time dependent due to the time-dependence of decay heat power (see Eq. (2.5.8)) and thus has to be calculated for each time step. Once the matrix is generated, the determination of the state probabilities $p_{n,j}(k)$ is made recursively through Eq. (4.2.4) for each time step. The determination of the state probabilities for each Top Event is made through Eq. (4.2.7).

5. INCORPORATION OF THE DFM AND MARKOV/CCMT MODELS INTO THE EXAMPLE PLANT PRA

Notation

$P(t)$	Power
t	Time
x_n	SG_n level (see Section 2.1)
Δt	Simulation time step
S, S'	System states; they include both the state of all the process variables and the configuration of all system components
P, P'	Probabilities
$N = (S, P)$	Node in Algorithm 1
$Prob[S,S']$	Probability of transition from state S to state S' in time Δt
$N = <(S1,P1),...,(Sk,Pk)>$	Node in Algorithm 2
$Conf(S)$	Configuration of system components in state S
E_{Ln}	SG_n Level error (see Section 2.1)
C_{Ln}	SG_n Compensated level (see Section 2.1)
S_{Bn}	SG_n BFV position (see Section 2.1)

5.1 Introduction

This chapter describes how the models generated according to the DFM and Markov/CCMT methodologies discussed in Chapters 3 and 4, respectively, can be integrated into the PRA for an existing, operational nuclear power plant as exemplified by a NUREG-1150 [66] plant. Then the risk significance of each failure mode listed in Tables 2.3.1 through 2.3.6 or their combinations can be determined using different measures (e.g. Fussel-Vesely, Birnbaum) through the standard features of the PRA quantification tools.

5.2 Description of Example Plant PRA

The model PRA to be used represents a simplified model of the example two-unit nuclear power plant (Fig. 5.2.1). Both units are PWRs, each with a three loop design. Both units are rated at 2441 MW(th), or 788 MW(e). The Unit 1 reactor first started commercial operation in 1972. The PRA to be used was modeled using the SAPHIRE PRA code, which uses the ET/FT methodology and is further explained in Section 6.

The example plant PRA models include a loss of offsite power (LOSP), loss of coolant accidents (LOCA), and fire and seismic events. Some of these initiating events are listed below in Table

5.2.1. Each initiating event leads to an ET modeling how various plant systems attempt to respond to the initiating event. A sample ET from the example plant PRA is also given below, as Fig. 5.2.2 and continued in Fig. 5.2.3, which models the plant's response to a turbine trip. A turbine trip could occur for several reasons, such as a loss of vacuum in the main condenser or if the turbine experiences overspeed. As can be seen by the ET, the plant's first response would be to scram the reactor through the reactor protection system (RPS). Failure of the RPS to scram the reactor will lead to an anticipated transient without scram (ATWS), and is modeled in a separate ET. After a successful reactor scram, the primary and secondary system safety relief valves (SRVs) must close (failure to do so leads to another ET modeling further plant actions in this scenario). With both the reactor scrammed and the SRVs closed, the Auxilliary Feedwater System (AFW) must then provide water to the SGs, maintaining a heat sink for the reactor. If the AFW system is unable to provided adequate water to the SGs, then the Main Feedwater (MFW) system is brought back online to provide cooling water to the SGs. Failure of both of these systems (Fig 5.2.3) will require High Pressure Injection (HPI), and opening of the relief valves for feed and bleed, and could possibly lead to core damage. Successful operation of the auxiliary or MFW system will result in a safe condition for the plant. The turbine trip ET represents an ideal model to incorporate a model for a digital feedwater controller, as the SG water level is critical to the safety of the plant and water is added from either the AFW or the MFW system.

As previously stated, the benchmark system to be incorporated into the example plant PRA models a DFWCS. Feedwater control monitors the water level in the plant SGs, and thus is tied to the MFW and AFW systems, which supply water to the SG. By ensuring an appropriate water level in the SGs, the DFWCS will maintain a steady heat sink for the reactor and allow for safe operation of the plant. The DFWCS monitors parameters such as steam flow out of the SG and feed flow into the SG, as well as the water level itself in the SG. Other parameters are also monitored, including system temperature and pressure, reactor power, FP speeds, and valve positions. The DFWCS will attempt to maintain water level in the SGs between certain setpoints, as well as maintain a balanced steam flow out and feedwater flow into the SG. This is accomplished primarily by controlling the speed of FPs and the position of flow control valves to adjust the rate of feedwater flow into the SG. This control was traditionally managed using analog systems; however, many plants are now upgrading their systems to incorporate DFWCSs.

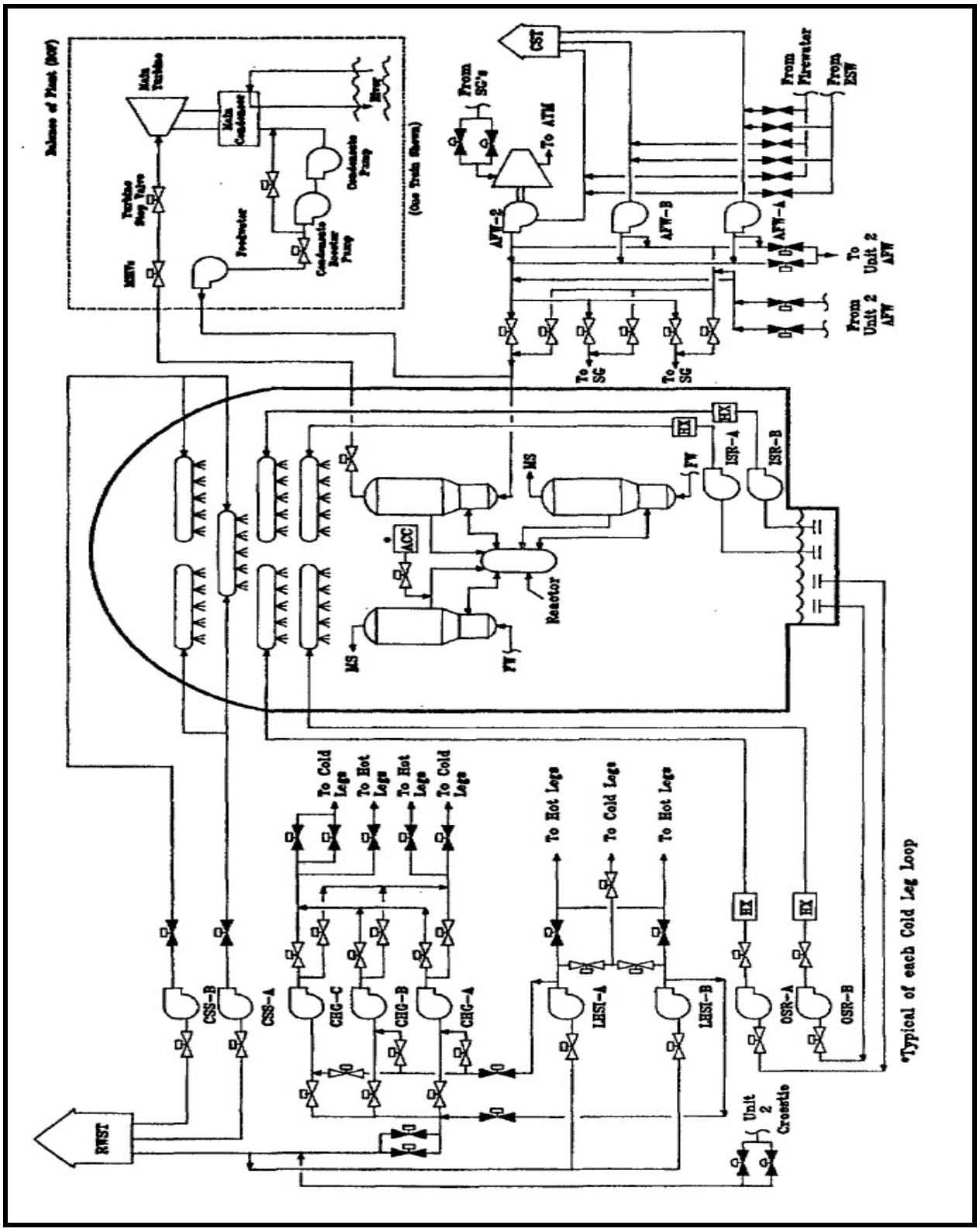

Figure 5.2.1 Schematic of Example Plant Unit 1 [124]

Table 5.2.1 Selected Example Plant PRA Initiating Events

SAPHIRE Name	Initiating Event Description
EQ	SEISMIC EVENT TREE WITH EARTHQUAKE INITIATING EVENT
FA	LARGE LOCA EVENT TREE
FAUX	FIRE EVENT TREE FOR AUXILIARY BUILDING
FCR	FIRE EVENT TREE FOR CONTROL ROOM
FCSR	FIRE EVENT TREE FOR CABLE VAULT/TUNNEL
FPR	FIRE EVENT TREE FOR CHARGING PUMP SERVICE WATER ROOM
FS1	MEDIUM LOCA EVENT TREE
FS2	SMALL LOCA EVENT TREE
FS3	VERY SMALL LOCA
FSWGR	FIRE EVENT TREE FOR EMERGENCY SWITCHGEAR ROOM
FT1	LOSS OF OFFSITE POWER
FT1S	STATION BLACKOUT AT UNIT 1
FT1SB	STATION BLACKOUT AT BOTH UNITS
FT2	LOSS OF MAIN FEEDWATER
FT3	TURBINE TRIP WITH MFW
FT5A	LOSS OF DC BUS A
FT5B	LOSS OF DC BUS B
FT7	SG TUBE RUPTURE
FTKT	ANTICIPATED TRANSIENT WITHOUT SCRAM (IE-T)
SE2	SEISMIC INDUCED LARGE LOSS OF COOLANT ACCIDENT (S-ALOCA)
SE3	SEISMIC INDUCED MEDIUM LOSS OF COOLANT ACCIDENT (S-MLOCA)
SE4	SEISMIC INDUCED SMALL LOSS OF COOLANT ACCIDENT (S-LOCA)
SE5	SEISMIC INDUCED SMALL LOSS OF OFFSITE POWER (S-LOSP)
SE6	SEISMIC INDUCED GENERAL TRANSIENT (S-GT)

In order to incorporate the DFWCS into the example plant PRA, it must be appropriately connected to the existing model. Since the reactor is shutdown following the turbine trip, the power is less than 15% and the DFWCS is in the low power mode, with BFV regulating the water inflow into the SG. With both the reactor scrammed and the SRVs closed, the AFW will then provide water to the SGs. Note that the available PRA model for the example plant does not contain any information on the MFW system. For the purposes of this proof-of-concept study, we will assume that the DFWCS is controlling the AFW system, and map the DFWCS equipment to the corresponding equipment in the AFW system (e.g. valves) for the ET in Fig. 5.2.2.

The AFW system for the example plant, shown in Fig. 5.2.4, consists of three trains that draw water from a single tank, the plant Condensate Storage Tank (CST). Each train can supply water to any of the three SGs. Water for two of these trains is pumped from the CST by Motor Driven Pumps (MDP), while the third train uses a Turbine Driven Pump (TDP), using steam produced from the SGs. The flow rate of the water sent to the SGs is controlled by a Motor-Operated Valve

(MOV). Each water line to the SGs has two such MOVs for redundancy. Two Air-Operated Valves (AOV) control the steam flow from the SGs to a turbine, which drives the AFW's Turbine Driven Pump. Therefore, the components controlled by the new DFWCS include the three FPs, six MOVs, and three AOVs, as all these components affect the water flowrate to the SGs. For modeling purposes, it is assumed that the DFWCS is being used for steam generator A, and that MOV-151-E and MOV-151-F, shown in Fig. 5.2.4, are the same MFV and BFV components contained in the DFWCS model. For demonstration purposes, the DFWCS is only connected to one steam generator, and the other two remain under analog control.

It should also be noted that this AFW system is cross-tied to the AFW system for the plant's Unit 2 reactor. Two additional MOVs, under the control of Unit 1, are used to control the flow rate from Unit 1 to Unit 2. Similarly, two MOVs controlled by Unit 2 can direct water from the Unit 2 AFW to Unit 1.

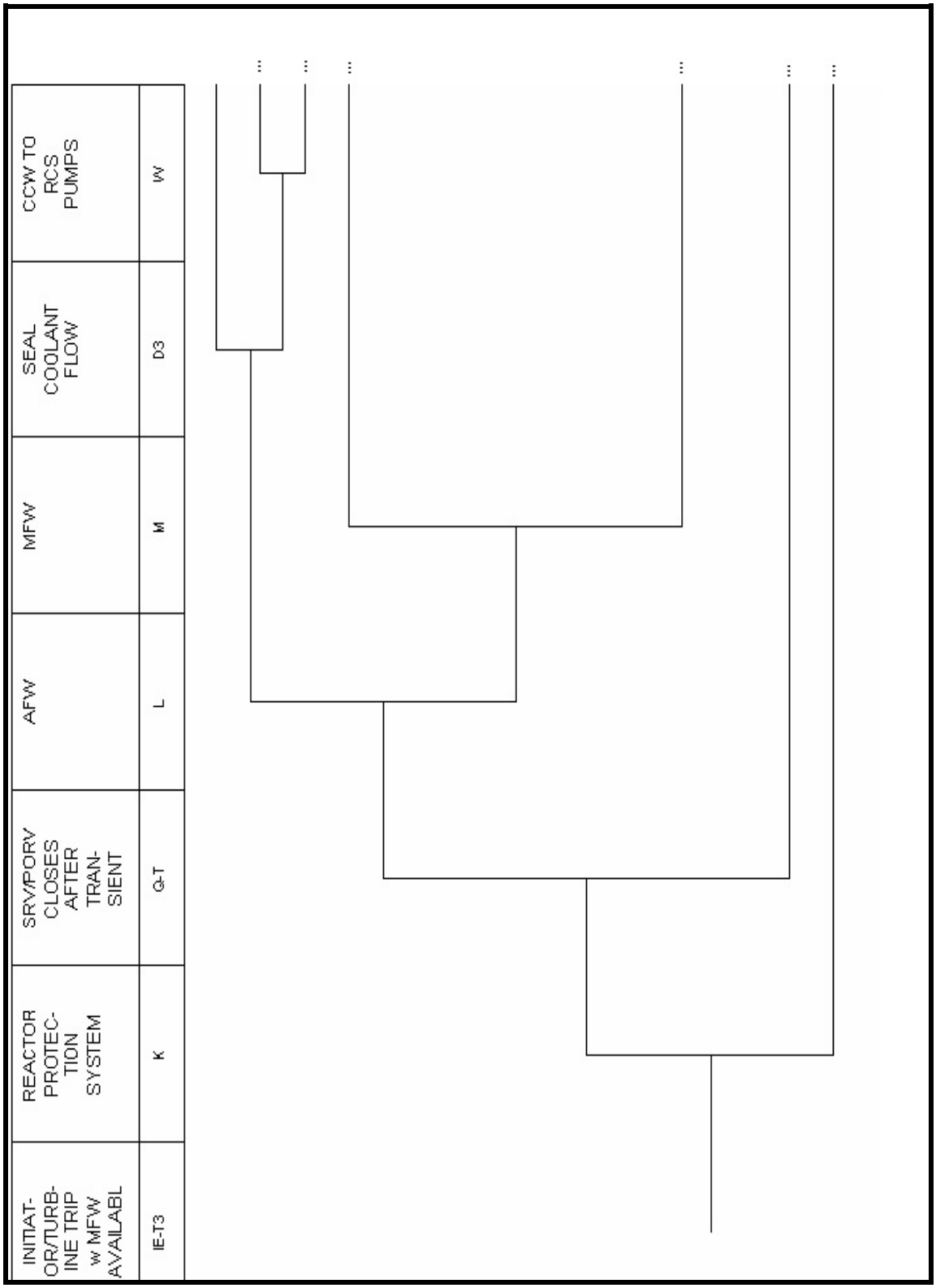

Figure 5.2.2 Example Plant Event Tree for Turbine Trip

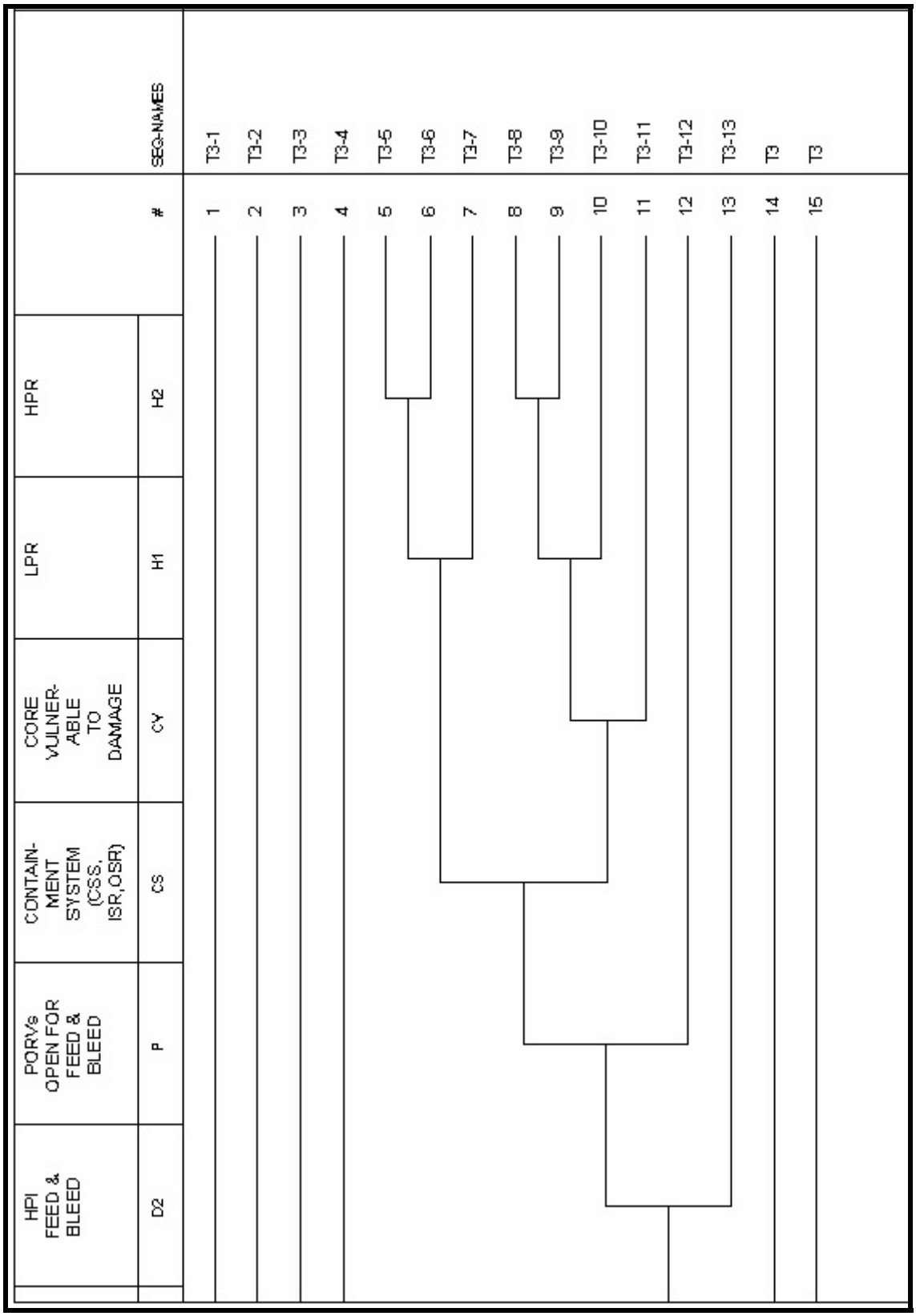

Figure 5.2.3 Example Plant Event Tree for Turbine Trip (continued)

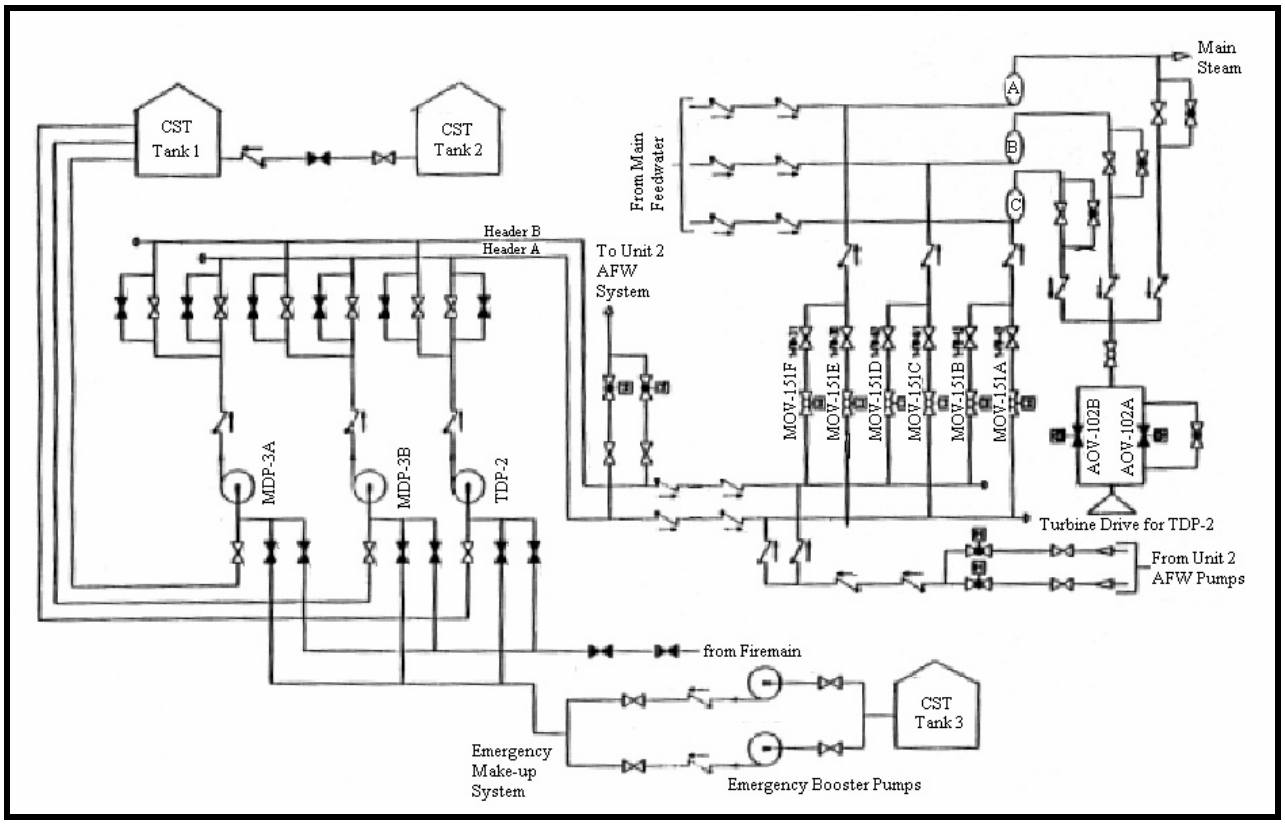

Figure 5.2.4 P&ID for Example Plant Simplified AFW System

Figures 5.2.5 to 5.2.20 below detail the FT for the simplified example plant AFW system. Due to the large number of valves and other components in the system, the model is quite large and has been broken up using transfer gates. The transfer gates (triangular elements) in the figures transfer to the FT with the name indicated. The top event, AFW-1 in Fig.5.2.5, occurs if there is insufficient flow to at least one of example plant's three SGs. Insufficient flow to any one of the three SGs will occur if either a check valve fails to open, or if there is insufficient flow due to upstream failures in the piping. Figure 5.2.5 shows that such insufficient flow may result from a stuck motor operated valve, or again by upstream failures.

The possible failures upstream of AFW-1 are modeled in FTs AFW13 (Fig. 5.2.6) and AFW14 (Fig. 5.2.7). By following the transfer gate to FT AFW13, insufficient flow could result from failure of one of two check valves or by further upstream failures. Insufficient flow may also occur if some of the water has been diverted to the Unit 2 system. The logic for AFW14 is modeled similarly. Upstream failures from AFW13 and AFW14 are both modeled in FT AFW15 (Fig. 5.2.8). Insufficient flow at AFW15 could occur due to undetected leakage in the system, insufficient water available in the condensate storage tank, no actuation signal sent to the system, or if backflow occurs due to either the motor driven FP 3A fails to start and backflow occurs in the nearby piping, or if the turbine driven FP fails to start and backflow occurs in nearby piping. Failure could also

result from the loss of electrical power, further modeled in FTs E1B (Fig. 5.2.14) and 4KV1J. In addition to these events modeled in AFW15, further events modeled in AFW17 (Fig. 5.2.9) and AFW18 (Fig. 5.2.10) must occur in order for AFW15 to indicate insufficient flow. The logic for AFW17 is similar to that in AFW15, but accounts for motor driven FP 3B failing to start, along with backflow, and draws power from electrical systems E1A (Fig. 5.2.13) and 4KV1H. Again, no actuation signal received, undetected leakage in the system piping, or insufficient water available in the condensate storage tank will also lead to insufficient water flow. FT AFW18 includes both motor driven FPs A and B, along with any associated backflow, but also accounts for insufficient steam flow to the turbine driven FP. Insufficient steam flow occurs if both events detailed in FT AFW21 (Fig. 5.2.11) and if air operated valve (AOV) A fails to open or is otherwise plugged, no actuation signal is received to the AOV, or if events in FT AFW22 (Fig. 5.2.12) occur.

Figure 5.2.11, which gives FT AFW21, shows that insufficient steam flow can result if air operated valve B fails to open, is plugged, or if no actuation signal is received. AFW21 will also show insufficient steam flow if the events in AFW22 occur. AFW22, given in Fig. 5.2.12, shows that insufficient flow will result if flow from all three pipe segments is blocked, due to either check valves failing to open or manual valves plugged.

Figure 5.2.13 and Figure 5.2.14 model the loss of electrical power to 125 V DC buses 1A and 1B, respectively. Recall that these FTs are entered from AFW17 or AFW15 and can account for insufficient flow to the SGs. For bus 1A, electrical power can be lost if the 480 V AC MCC 1H1-1 (modeled in Fig. 5.2.15) fails or its respective circuit breaker is open, and if the 480 V AC MCC 1H1-2 (modeled in Fig. 5.2.16) fails or its respective circuit breaker is open, and a battery failure occurs. Similarly, for bus 1B, electrical power can be lost if the 480 V AC MCC 1J1-1 (modeled in Fig. 5.2.17) fails or its respective circuit breaker is open, and if the 480 V AC MCC 1J1-2 (modeled in Fig. 5.2.18) fails or its respective circuit breaker is open, and a battery failure occurs. Electrical power to either system will also be lost if the bus itself has failed. Figure 5.2.15 shows that electrical power to the MCC 1H1-1 is lost if either buswork failure occurs, the circuit breaker is open, or if a failure occurs to the 4kV AC electrical bus 1H (modeled in Fig. 5.2.19). MCC 1H-1 will also lose power due to buswork failure, an open circuit breaker, or loss of bus 1H. Similarly, MCC 1J-1 and 1J-2 lose power due to buswork failure, an open circuit breaker, or a failure due to the 4kV AC electrical bus 1J (modeled in Fig. 5.2.20). These two buses will fail to provide power if either buswork failure occurs, or if offsite power is lost and backup power provided by emergency diesel generators is unavailable. Diesel generator #1 provides power to the 4kV bus 1H, and the #3 diesel generator provides power to bus 1J. Either diesel generator will fail to provide power if it fails to start or run for at least an hour, if it is out for testing and maintenance, or if a respective circuit breaker has failed.

To summarize, the FT models show that the AFW system will fail to provide sufficient water to the three SGs if the appropriate valves fail to open, FPs fail to start or insufficient steam is sent to power the turbine driven FP, pipe leakage occurs, or there is insufficient power provided to motorized components. Electrical power can be lost if offsite power is lost and emergency diesel generators fail to run, or by a buswork failure in the plant's electrical system.

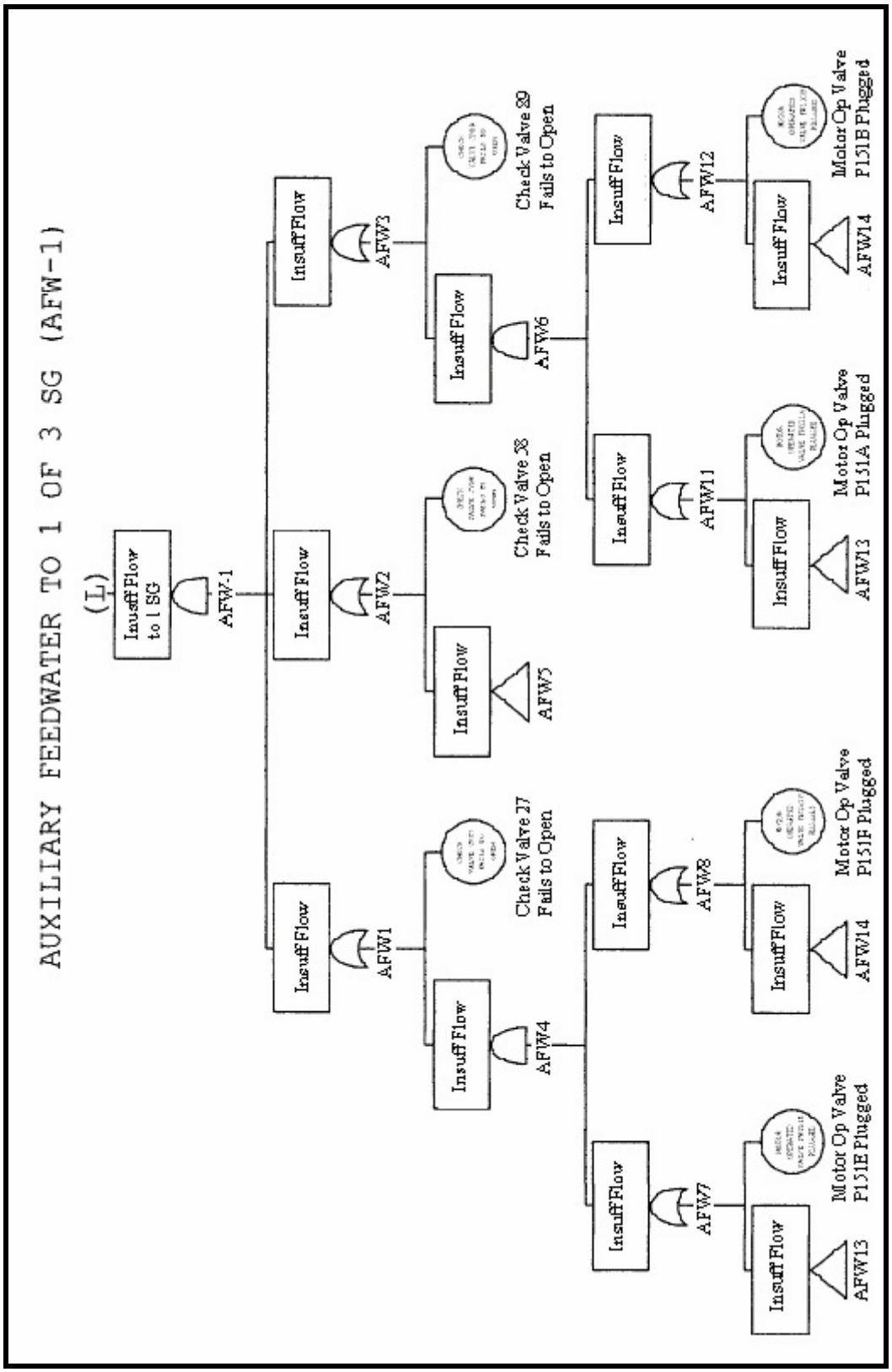

Figure 5.2.5 Example Plant AFW Top Event - Insufficient Water Flow to SGs

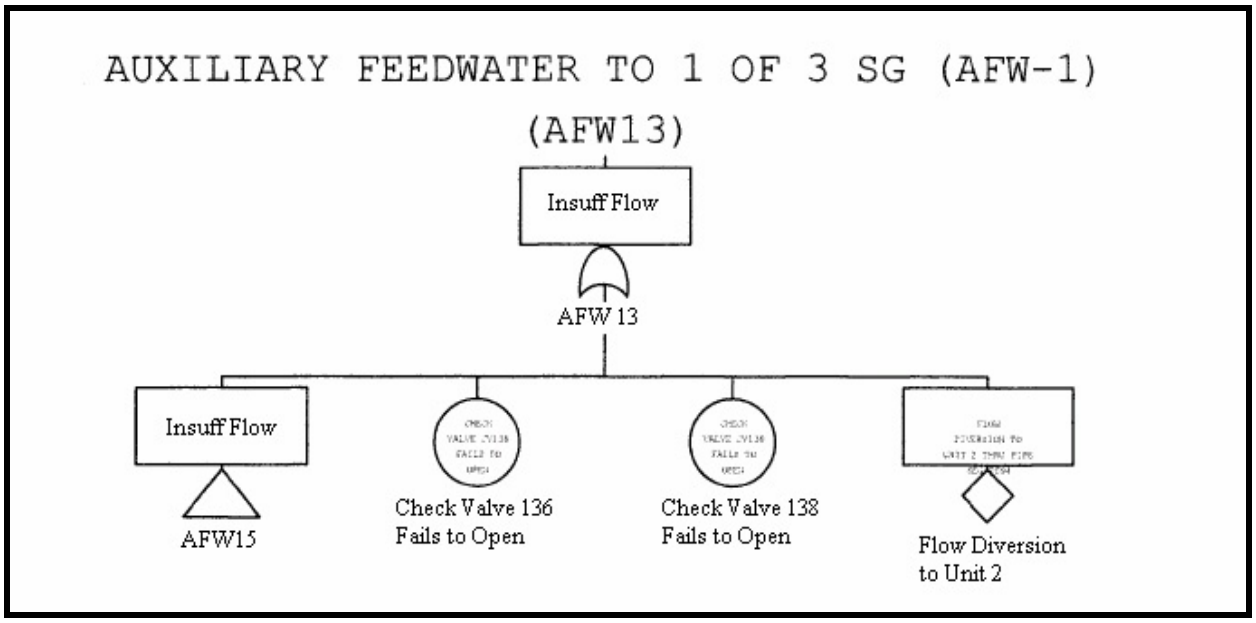

Figure 5.2.6 AFW13 - Subtree for Example Plant AFW System

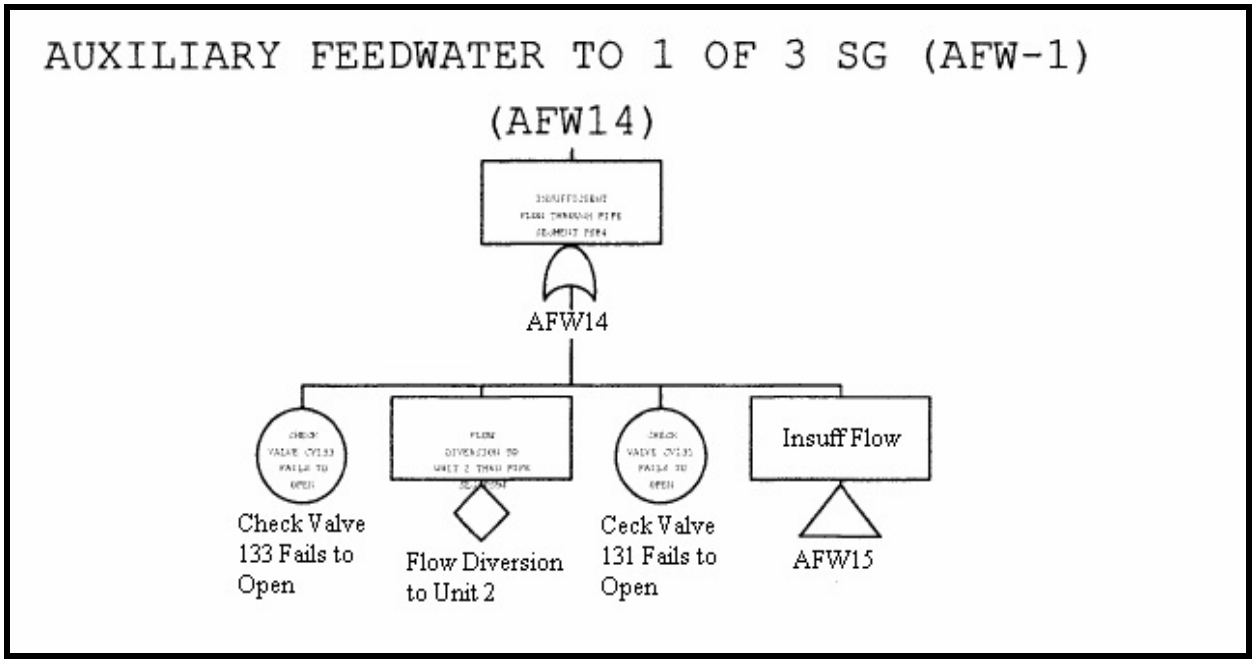

Figure 5.2.7 AFW14 - Subtree for Example Plant AFW System

Figure 5.2.8 AFW15 - Subtree for Example Plant AFW System

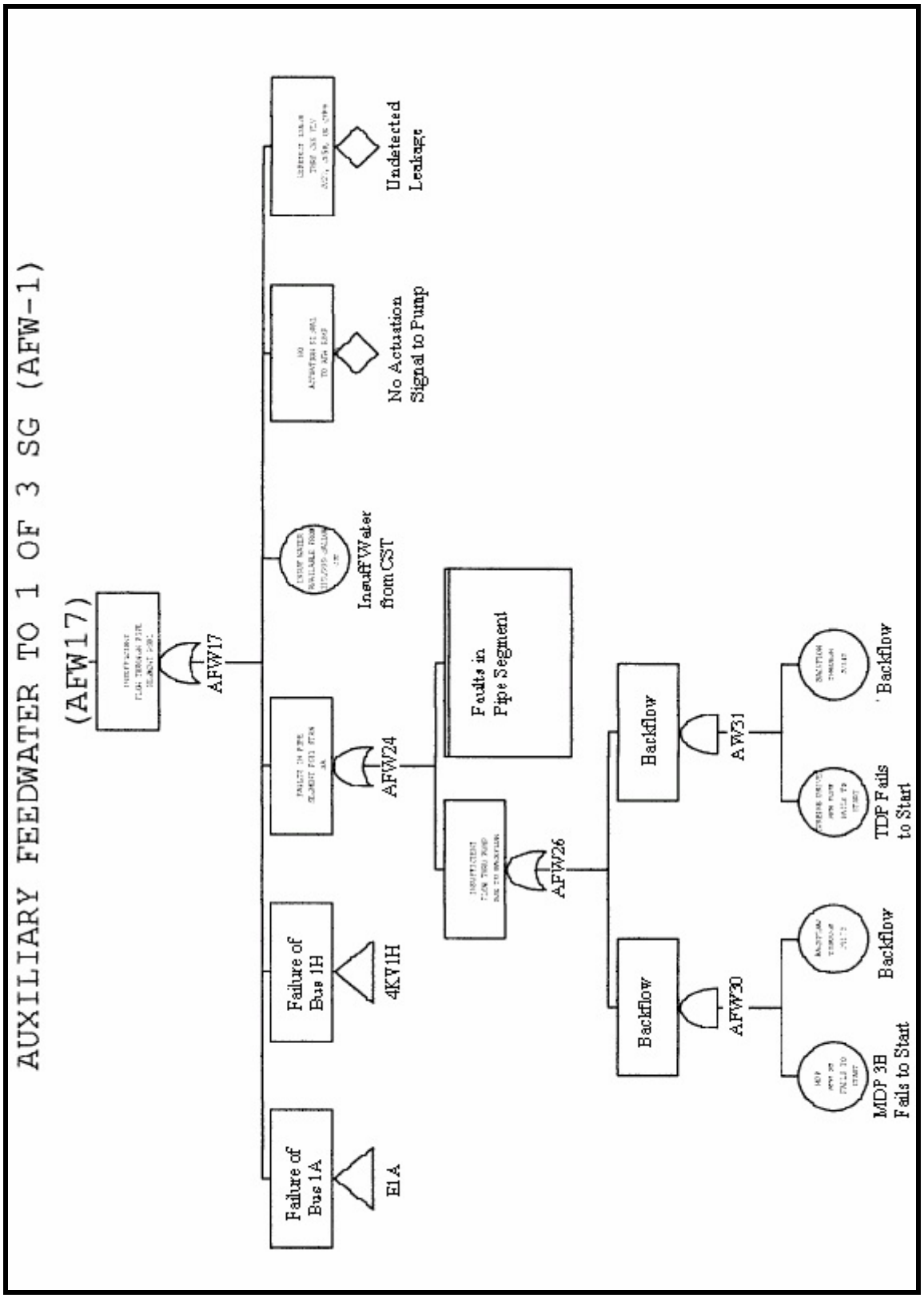

Figure 5.2.9 AFW17 - Subtree for Example Plant AFW System

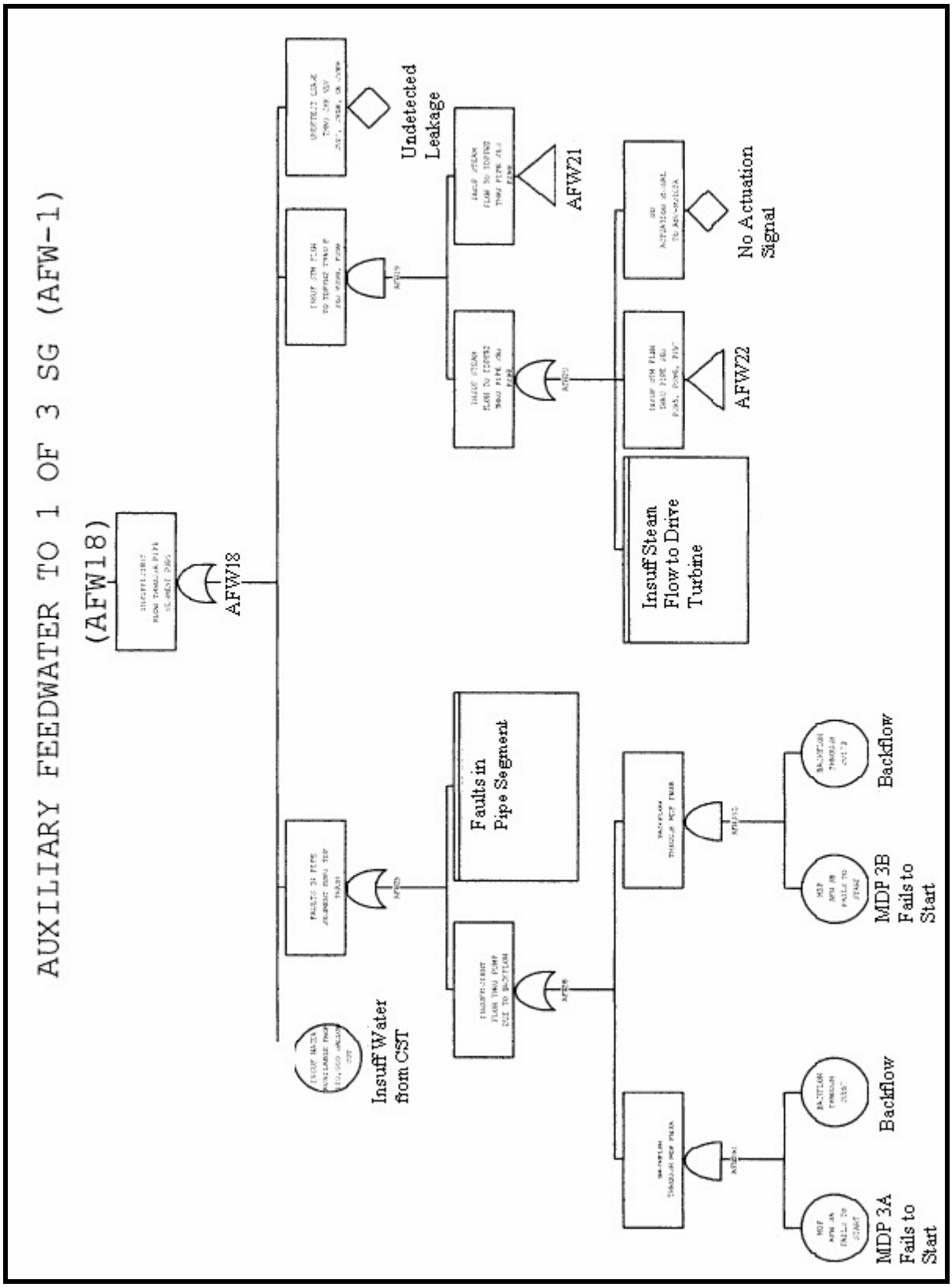

Figure 5.2.10 AFW18 - Subtree for Example Plant AFW System

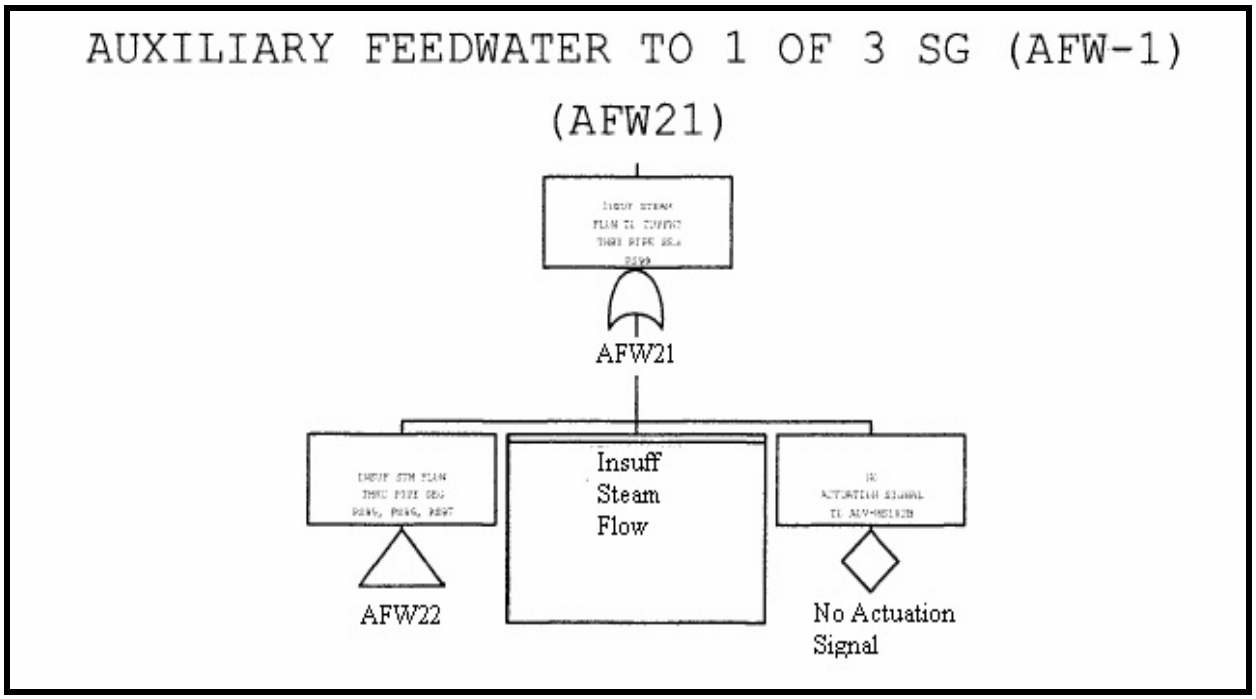

Figure 5.2.11 AFW21 - Subtree for Example Plant AFW System

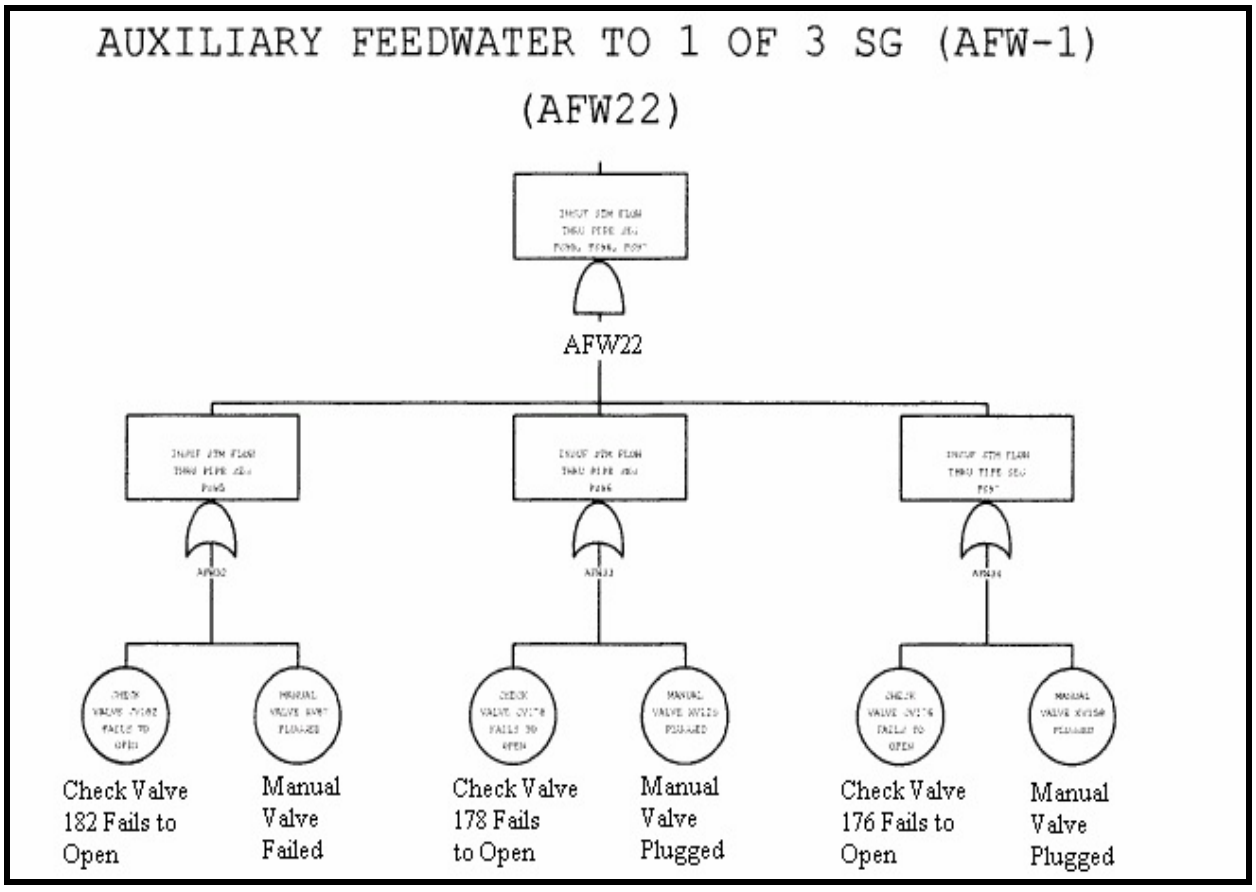

Figure 5.2.12 AFW22 - Subtree for Example Plant AFW System

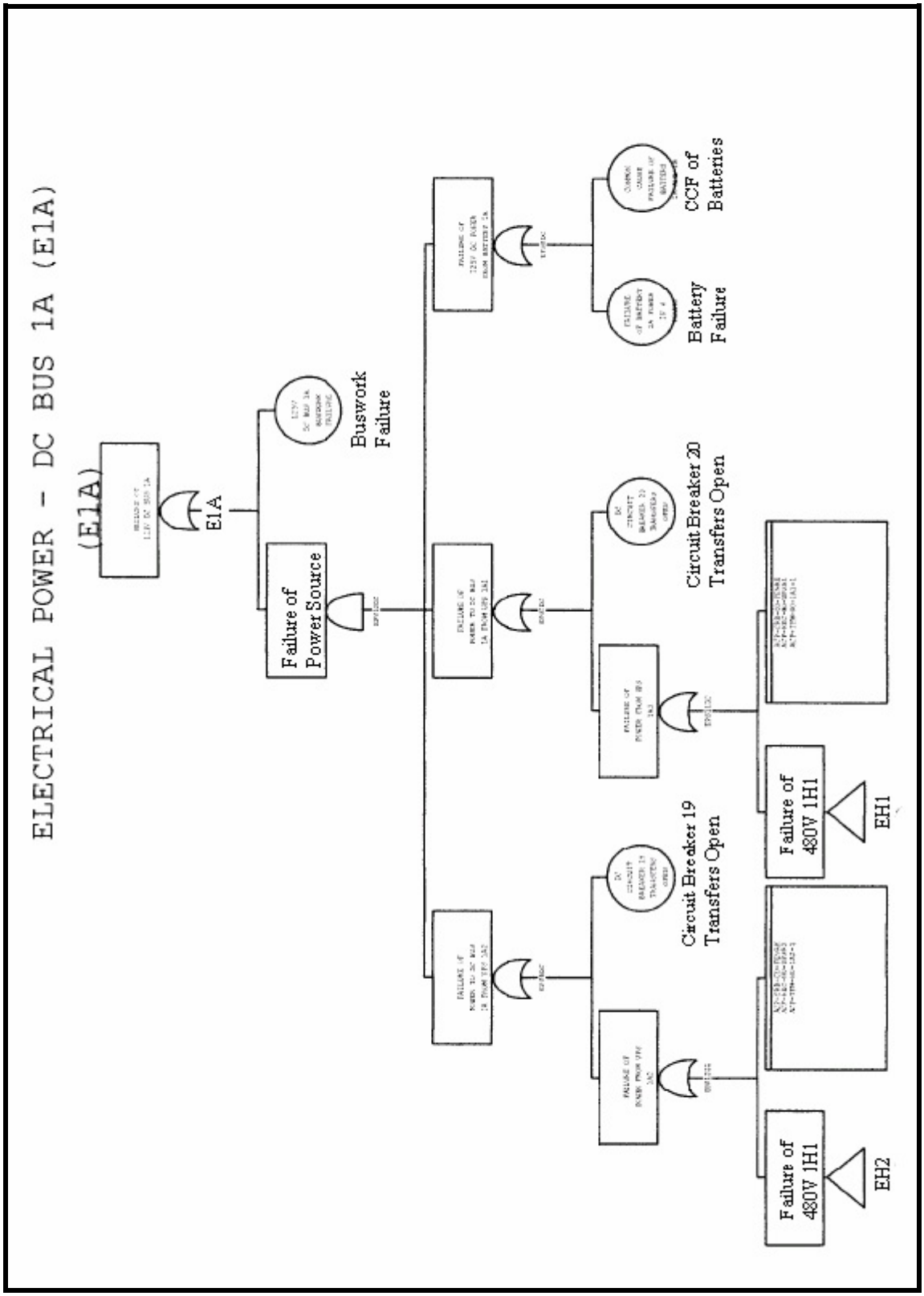

Figure 5.2.13 E1A - Failure of 125V DC Bus 1A

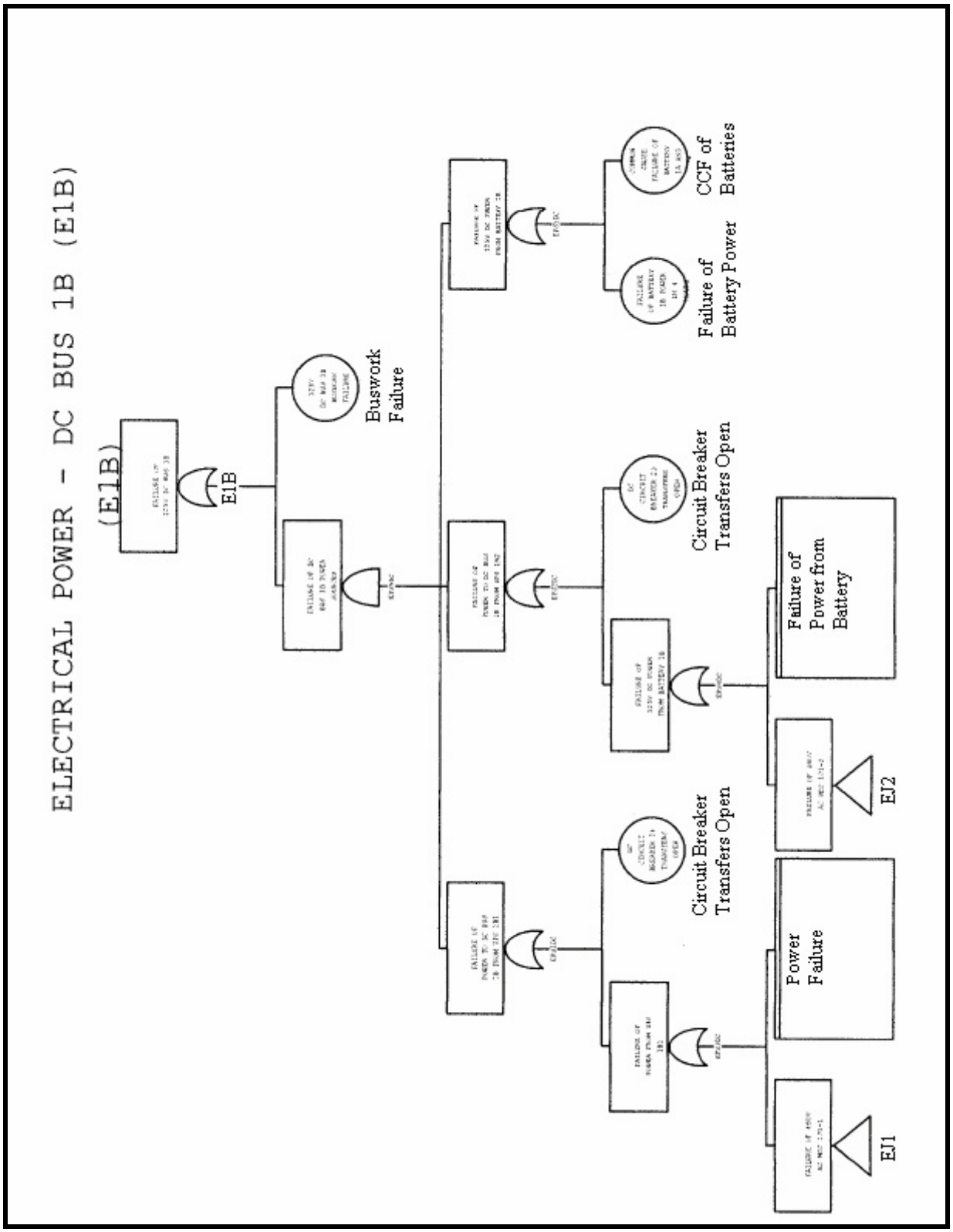

Figure 5.2.14 E1B - Failure of 125V DC Bus 1B

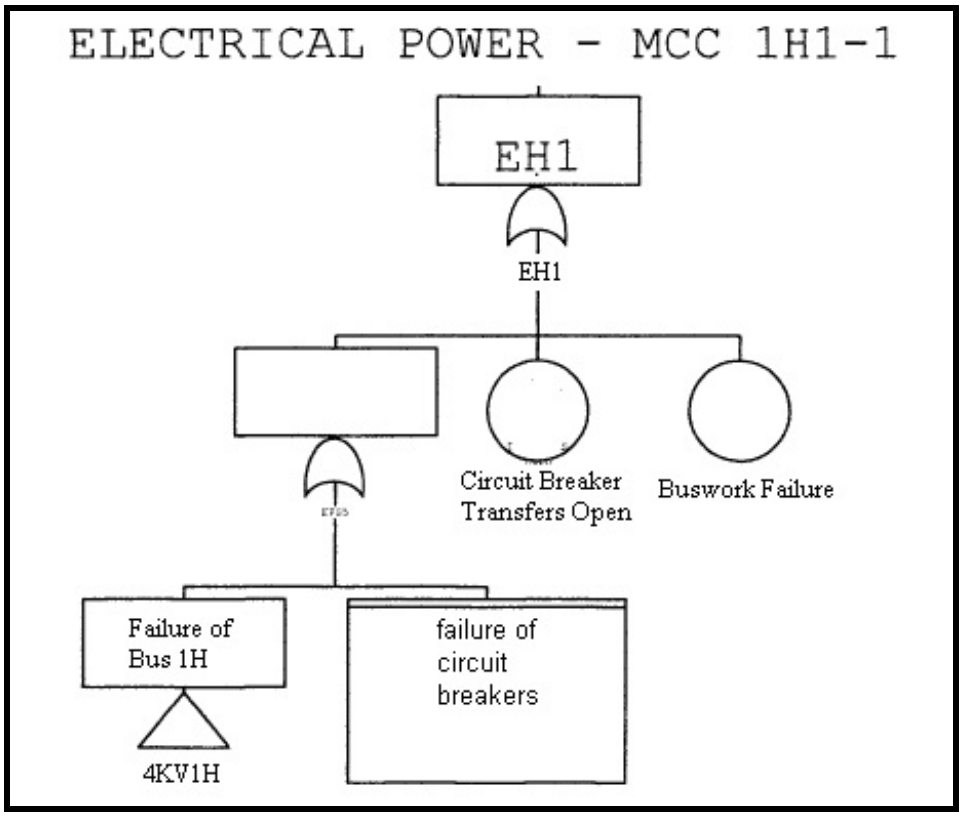

Figure 5.2.15 EH1 - Failure of 480 V AC MCC

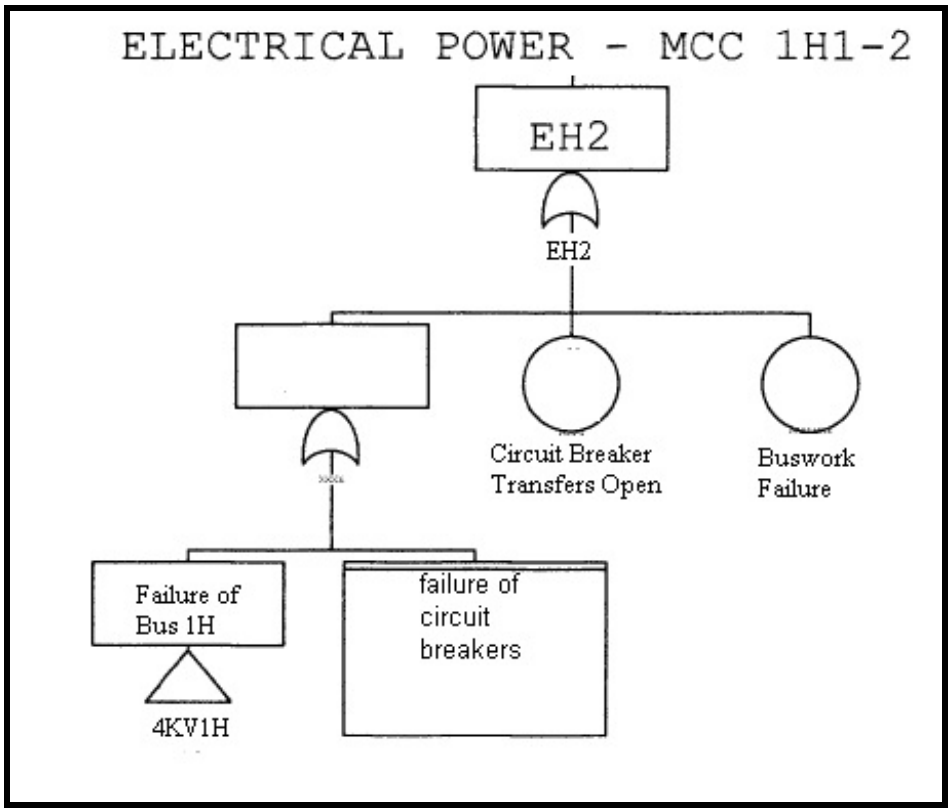

Figure 5.2.16 EH2 - Failure of 480 V AC MCC

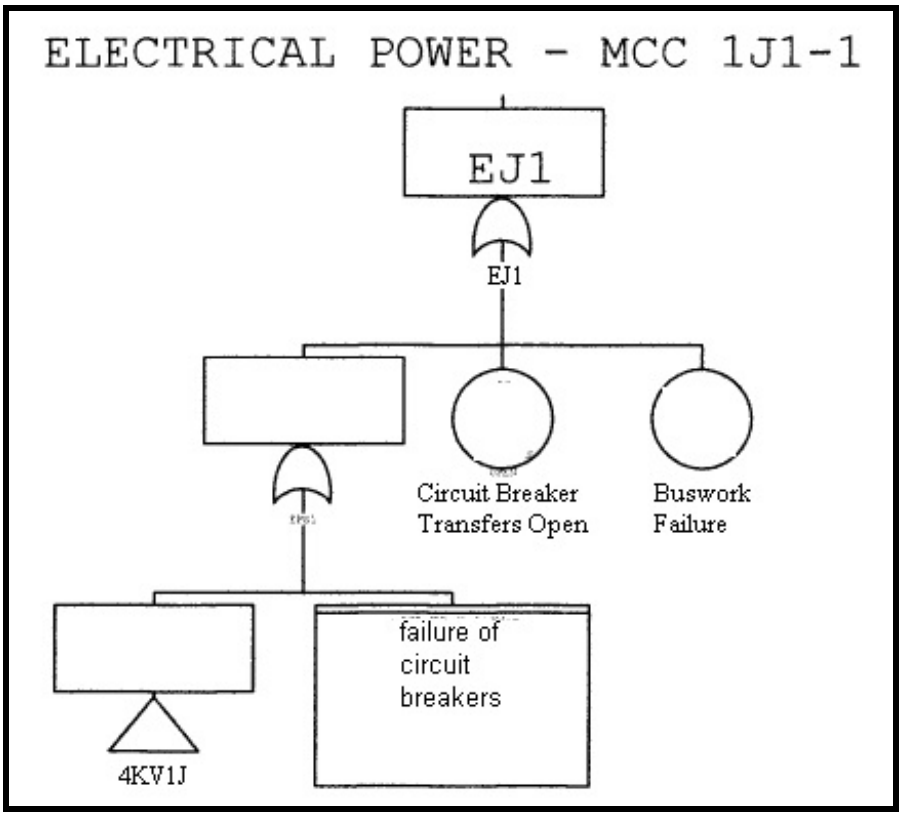

Figure 5.2.17 EJ1 - Failure of 480 V AC MCC

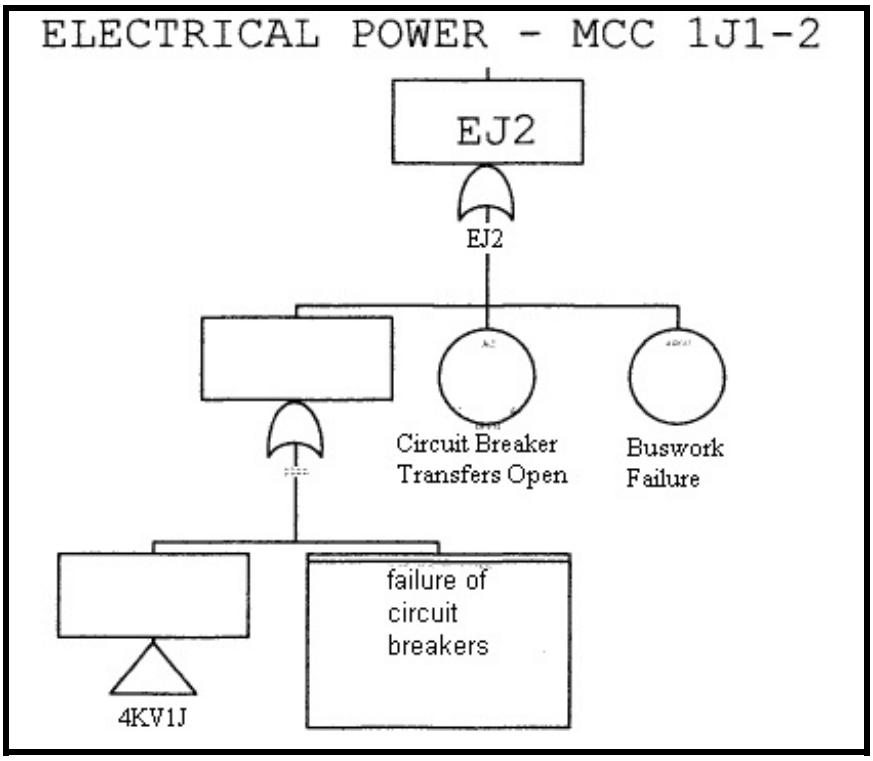

Figure 5.2.18 EJ2 - Failure of 480 V AC MCC

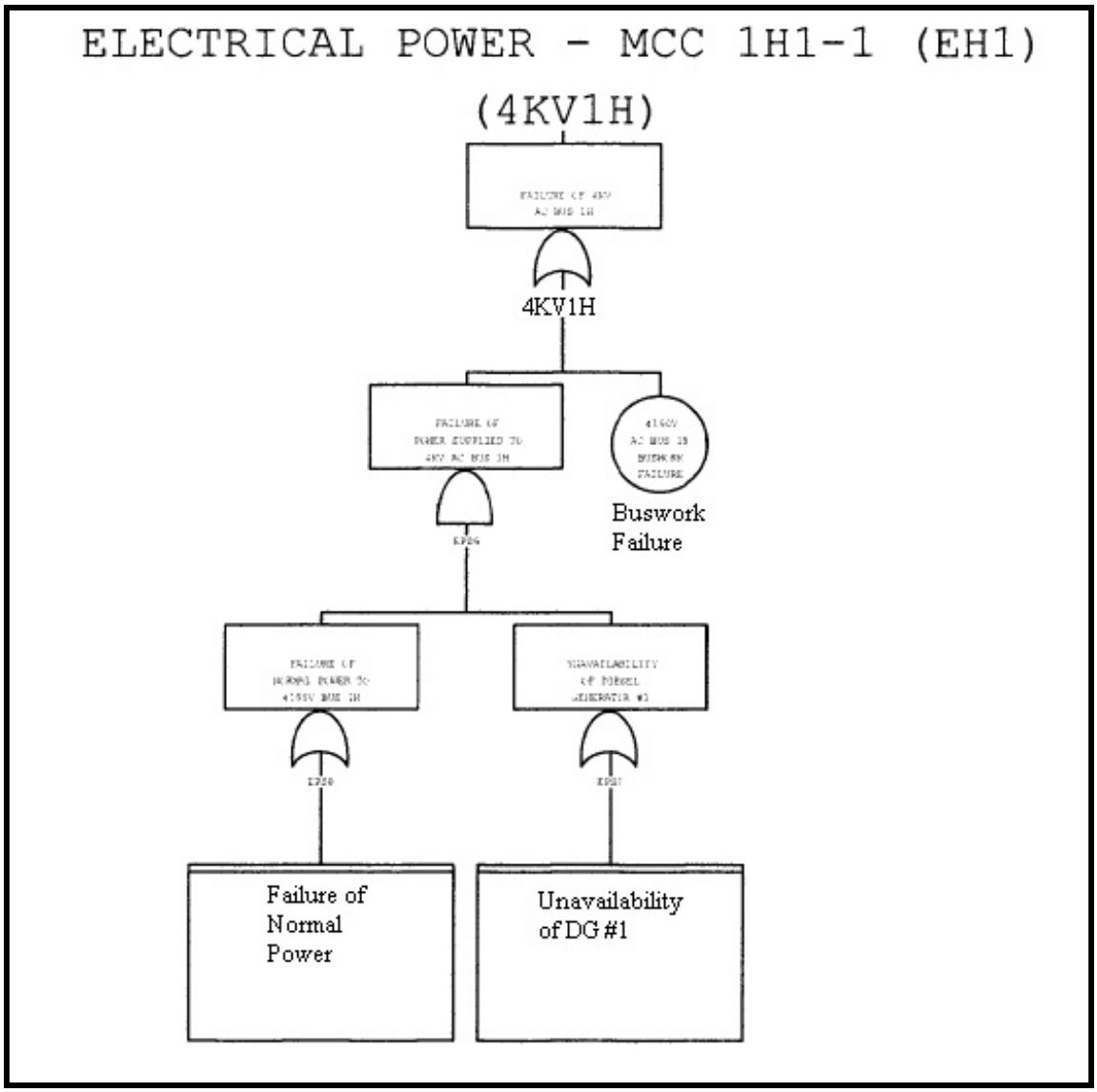

Figure 5.2.19 4KV1H - Failure of 4kV AC Bus 1H

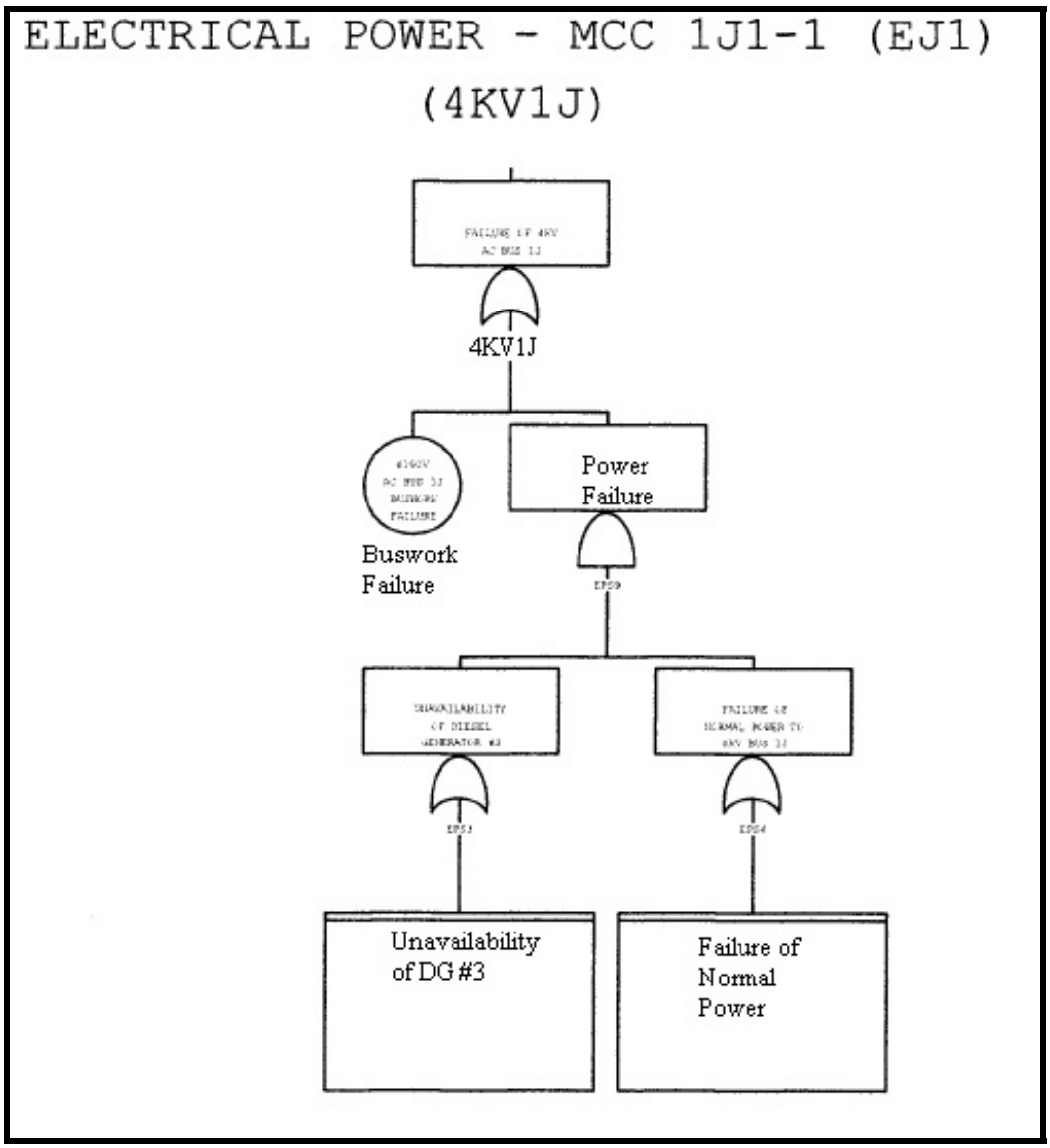

Figure 5.2.20 4KV1J - Failure of 4kV AC Bus 1J

Incorporation of the DFWCS model into the example plant PRA can be performed using the plant models present in the SAPHIRE database and SAPHIRE's MAR-D feature. The MAR-D feature allows FT logical information, ETs, basic event failure data, and other information to be imported to or exported from the SAPHIRE database. By composing the model for the DFWCS into an

appropriately formatted text file, the model can be loaded into the SAPHIRE database and connected to the rest of the plant PRA.

In order for SAPHIRE to recognize an imported file, the file must use the proper format. While the file may have any name desired, the file extension to be used will vary based on the type of information to be entered into SAPHIRE. To enter an ET, the extension .ETL (event tree logic) or .ETG (event tree graphics) must be used. To enter a FT, the extension .FTL (fault tree logic) or .FTG (fault tree graphics) must be used. The format for the ET files, as well as the FT graphics file, all contain information pertaining to the graphics as displayed in the SAPHIRE editors. Therefore, the format for these files is complex. However, the format for the FT logic file, .FTL, contains only the logic information of the FT, and is therefore very simple. If desired, the FT logic may be converted into a graphical format once it has been entered into the SAPHIRE database. To illustrate the format to be used for a .FTL file, a sample FT is given below. The text gives the logic information from the small AFW22 FT, given above in Fig. 5.2.12, for insufficient flow.

Example plant-NUREG-1150, AFW22 =

```
AFW22                   AND  AFW32 AFW33 AFW34
AFW32                   OR   AFW-CKV-FT-CV182 AFW-XVM-PG-XV87
AFW33                   OR   AFW-CKV-FT-CV178 AFW-XVM-PG-XV120
AFW34                   OR   AFW-CKV-FT-CV176 AFW-XVM-PG-XV158
```

In the top line, example plant-NUREG-1150 represents the project database this file was exported from. When importing a new file, any name may be used here (it need not match the project name the file is to be imported into in SAPHIRE). AFW22 is the name of the FT being imported. This name should match the first name given on the second line, which represents the top event of the FT. Twenty four spaces are allotted for the top event name, followed by one more space before the description of each daughter of the top event. A total of six spaces is used to describe what type of gate is used for the gate (OR, AND, or TRAN for transfer gates). The name of each daughter follows the gate type. Up to 24 spaces can be used for each daughter, and there must be a space between each daughter. Note that it is not necessary to state whether daughters are gates or basic events. Every line after the second describes a gate, using the same format as for the top event on the second line. At the end of the tree, any daughters that have not been described as gates will be assumed to be basic events. If the name of any gate or basic event does not appear in the SAPHIRE project, SAPHIRE will create that gate or event.

If the name of a gate or event does appear, SAPHIRE will assume them to be the same and use any descriptions or basic event data already present. If a FT of the same name as the one being imported already exists, SAPHIRE will replace the old FT with the new one being imported. Thus, any FT generated may be incorporated into the database of an existing PRA modeled in SAPHIRE, allowing for the new model to be added to existing plant data.

5.3 Incorporation of DFM Output into the Example Plant PRA

Event tree/fault tree models, implemented with automated PRA tools such as SAPHIRE[7], CAFTA[8] and RISKMAN[9] will likely remain the tool of choice for generating nuclear power plant PRAs. It is known that this well-accepted tool has some shortcomings in handling dynamic systems and non-coherent logic structures. The DFM, being a multi-valued logic dynamic analysis tool with quantification capability, is a natural extension of ET/FT analysis to augment the capability to analyze dynamic systems and non-coherent logic structures. In that respect, it is feasible to integrate the DFM analysis of the benchmark feedwater control system into the master structure of the example plant PRA developed with SAPHIRE. Section 5.3.1 outlines the conceptual steps for using DFM to augment an ET/FT PRA structure and integrating the DFM outputs back into the master PRA. Section 5.3.2 presents an example from the benchmark system analysis to illustrate the procedures. Section 5.3.3 identifies some of the technical issues and potential resolutions to bring about a practical integration solution.

5.3.1 Augmentation of the ET/FT Structure with DFM

The example plant PRA contains a large set of ETs for modeling accident progression as a result of many different initiating events. Some of these ETs contain pivotal events that are tied to the failure of the feedwater control system. Instead of expanding these feedwater control system related pivotal events with FT models, the DFM model of the benchmark system will be solved. The feedwater control system related pivotal events in the ETs will be defined as the top events in a set of deductive analyses. The sets of prime implicants identified in these deductive DFM analyses are logically equivalent to the cut sets for the pivotal events and can be used to quantify them.

In the simplest scenario, when the feedwater control system model is not coupled with the other systems modeled in the PRA, the prime implicants of the feedwater control system will not contain basic events that correlate to the rest of the PRA model. This allows the prime implicant sets to be quantified independently from the master PRA model. The point estimates outputs from DFM quantification can be used as point estimates for the pivotal events in the ET model to calculate the base case results.

For a coupled scenario, when the feedwater control system prime implicants are correlated with the PRA model cut sets, such as common basic events in the ET/FT model and the DFM model, quantification of the final results will be complicated. One possible solution will be to convert the prime implicants identified by the DFM analyses to cut sets in the ET/FT model and then use SAPHIRE to quantify the results. However, this has the potential of introducing errors into the calculation. The reason is that the cut sets converted from the DFM prime implicants are very likely to contain dynamic information and a non-coherent logic structure, and that SAPHIRE is known to produce erroneous results when the cut sets have these time dependent and non-coherent characteristics. As a simple example, assume that the DFM analysis yields a single prime implicant:

A = 1 @ time = -1 and B = 1 @ time = 0,

with A being a process parameter that is discretized into 3 states, 1, 2 and 3, and B being a process parameter that is discretized into 2 states, 1 and 2.

This prime implicant means that the top event can be brought about by process variable A entering state 1 followed by process variable B entering state 1. This prime implicant contains both dynamic information and a non-coherent logic structure. In particular, event A=1 has to occur prior to event B=1, and A is not a binary variable.

To convert this prime implicant into a cut set recognized by SAPHIRE, the multi-state basic event A=1 needs to be decomposed into 3 binary basic events A1 (A in state 1) = true AND A2 = false AND A3 = false, with the additional dependency between A1, A2, and A3 (no more than one of them can be true). The cut set then becomes:

A1 ^ ¬A2 ^ ¬A3 ^ B

Also, note that the dynamic information regarding A happening before B is no longer available. However, converting the DFM prime implicants into cut sets will allow SAPHIRE to solve the ET/FT model with the dependency accounted for.

For beyond base case point estimate analyses, such as sensitivity analysis, importance measures and uncertainties, the conceptual framework would apply. However, some automation capabilities need to be developed to allow SAPHIRE and DFM to exchange information regarding basic events in cut sets/prime implicants. Some technical issues in this regard are identified in Section 5.3.3.

5.3.2 Example of Integrating DFM Results into the Master PRA

For example, using the Turbine Trip initiating ET (Fig. 5.2.2) as an example, the pivotal event MFW, corresponding to the failure of the Main Feedwater System to maintain steam generator level, is analyzed with DFM. The analyses for steam generator level high and steam generator level low are carried out as described in Section 3.5.2. Depending on the ease of integration with SAPHIRE, the prime implicants or the mutually exclusive implicants identified by DFM can be exported back to the SAPHIRE ET model, as shown in Fig. 5.3.1.

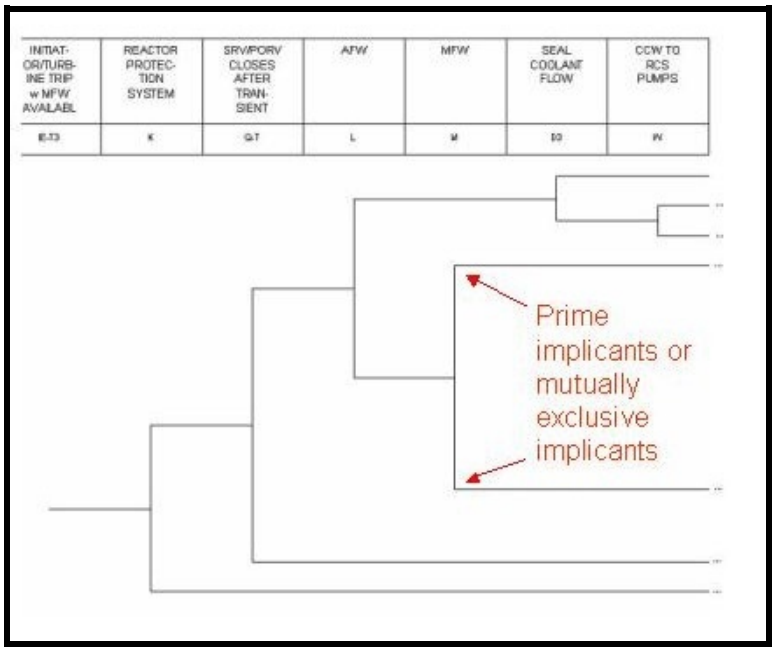

Figure 5.3.1 Integration of DFM Results into SAPHIRE

For the analysis of this simplified benchmark problem, the prime implicants do not show any time dependency among the conditions in the prime implicants. However, with the DFM model of full benchmark system, it is expected that dynamic behavior will be associated with the prime implicants, i.e., prime implicants will contain conditions across different time steps. Under such conditions, the prime implicants and the probability estimated cannot be exported as is to SAPHIRE. Some post-processing is required. These are further discussed in Section 6.

5.3.3 Technical Issues and Potential Resolution for Integrating DFM into the Master PRA

It is recognized that augmenting the SAPHIRE ET/FT analysis capability with DFM will be beneficial in terms of accounting for dynamic and non-coherent logic behavior. As discussed previously in Section 5.3.1, the concept is feasible. However, to make this practically applicable, several technical issues need to be addressed. These technical issues are associated with quantification and integration of results beyond point-estimate outputs.

- Sensitivity Analysis: Is there anything not straightforward in the quantification of the alternate cases, when a parameter value in the DFM portion of the models is the object of the sensitivity question? If not, then the answer would be that those prime implicants for the alternate case that contain the variable of interest be identified and tagged in the post-processing, so that the SAPHIRE main module can be used to produce the desirable sensitivity analysis results.

- Importance Measures: How should the importance measures be calculated? Is this simply a variation of the "sensitivity analysis" question or is there some hidden issue that we need to investigate better? If not, the computational process can follow what is outlined above for the sensitivity analyses. To address the question, a better understanding of how the importance measure results are derived in the current SAPHIRE version is required.
- Uncertainty Analysis: How should a full uncertainty analysis be carried out? One general question concerns the control mechanism of the Monte Carlo (or Latin hypercube) sampling scheme when parameters on both sides of the SAPHIRE/DFM interface are sampled. An important corollary of the question is how to accurately estimate the uncertainty if some basic events are present in both the coherent (i.e., SAPHIRE) portion of the model and the non-coherent (i.e., DFM) portion of the model. In such a case the SAPHIRE "control engine" needs to "know" that the event probability has to be sampled only once and the same input value needs to be used on both the SAPHIRE and DFM sides to quantify the logic structure. If the event were to be sampled separately on either side, this would be equivalent to erroneously consider each instance as a separate, statistically independent event. A possible approach is to identify and tag those prime implicants that contain variables common to the coherent and non-coherent portions in the post-processing, so that the SAPHIRE main module can be used to produce the desirable Monte Carlo simulation results.

5.4 Incorporation of Markov/CCMT Methodology Output into the Example Plant PRA

The integration of any model of digital I&C systems into existing PRAs currently requires the ability to link the output of these models to the standard event tree/fault tree models used by PRA tools such as SAPHIRE. Very few attempts have been encountered in the reliability literature to generate ETs or FTs from Markov models [139]. As described in Chapter 1, dynamic event trees (DETs) have been used in methodologies capable of accounting for at least Type I interactions in the reliability modeling of digital I&C systems. The DETs are similar to conventional event trees except that the branching times are determined from the system simulator through user specified branching rules [37, 38, 40, 41]. This section describes an approach based on dynamic event trees to integrate the output of a Markov model of a system into an existing PRA. Section 5.4.1 describes two algorithms for the generation of dynamic event trees from a Markov model of the system [140]. Section 5.4.2 presents a sample analysis of one failure scenario for the benchmark system using the DETs generated from a Markov model. Section 5.4.3 shows how the DETs can be incorporated into an existing PRA developed in SAPHIRE. Section 5.4.4 discusses some outstanding issues.

5.4.1 DET Generation from Markov Model [140]

This section describes an approach to generate the DET for a given system once a Markov model has been built using the CCMT described in Chapter 4. The basic idea of this approach is to use the transition matrix of the Markov model of the system as a graph representation of a finite state machine (a discrete process model of the stochastic dynamic behavior of the system). With this representation and standard search algorithms [141] it is possible to explore all possible paths to

failure (scenarios) with associated probabilities and to construct dynamic event trees of arbitrary depth.

The DET generated by this approach has a somewhat unusual structure because instead of simply describing some failure scenario, it captures all possible failure scenarios for the system under consideration. Assuming the system starts in an operational state where all process variables are in the nominal range and all system components are operational, the algorithm starts branching through discrete time steps such that each level of branching in the tree represents all the possible states in which the system may be after a given time interval. Branching stops any time a branch reaches a "sink" state (i.e., a state from which the system cannot move out) or the probability associated with the branch is below a chosen threshold. It is also possible to stop after a certain amount of system time has elapsed or, equivalently, once the branching has reached a chosen depth in the tree.

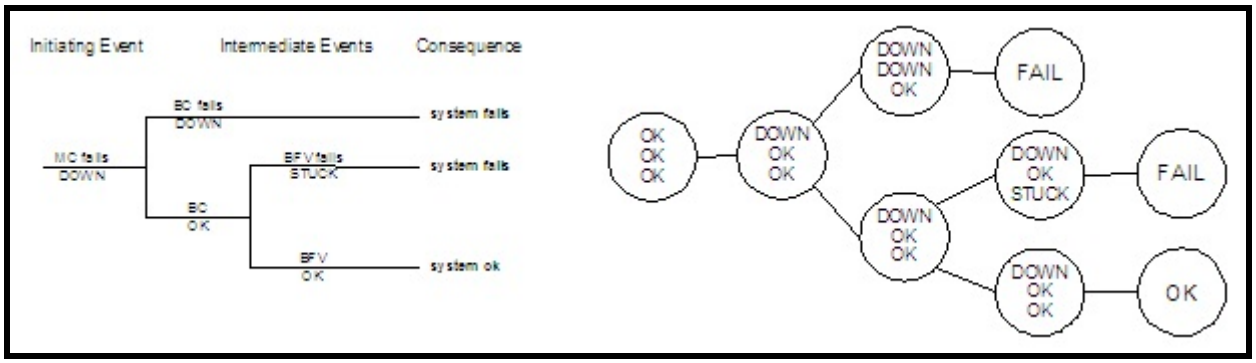

Figure 5.4.1 Event Tree vs. Tree Data Structure

The DET is represented by a tree data structure. A tree data structure is composed of "nodes" (where information is stored) and "links" that connect the nodes. The nodes in the tree data structure correspond to the branching points in the DET and the links represent the branches. Figure 5.4.1 shows a simple event tree for a simplified view of the level controller and a tree data structure that might be used to represent it. In the example, the tree nodes hold the information about the system component states: Main Computer (MC), Backup Computer (BC), and Backup Flow Valve (BFV). The event corresponding to a specific branch in the tree can be deduced by comparing the configurations at the beginning and at the end of the branch.

Sections 5.4.1.1 and 5.4.1.2 present two algorithms for the generation of DETs from the transition matrix of the Markov model of a system.

5.4.1.1 Algorithm 1

The nodes in the tree data structure used in the first DET generation algorithm (see Fig. 5.4.2) store the state *S* of the system and the probability *P* that the system will have followed the specific path in the DET from the initial state to state *S*. In the current discussion, a state of the system is

always meant to include both the state of all the process variables and the configuration of all system components.

```
initialize DET root node to initial state and probability 1
add DET root node to queue Q of nodes to process
while Q is not empty
    remove next node N = (S,P) from Q
    if S is not a sink state
        for each possible state S'
            if Prob[S,S'] > 0
                compute probability P' for this branch as Prob[S,S'] * P
                if P' > epsilon
                    create new node N' = (S',P')
                    add N' to the list of children of N in the DET
                    add N' to queue Q of nodes to process
                end if
            end if
        end for each
    end if
end while
```

Figure 5.4.2 Dynamic Event Tree Generation–Algorithm 1

The algorithm uses a queue (first-in, first-out behavior) to hold the nodes in the tree before they are processed by the algorithm. It starts by creating the top node in the tree (the root) containing the initial state of the system (usually the state in which the process variables are all in the nominal range and all the system components are operational) and a probability equal to 1. This node is added to the initially empty queue. Then the algorithm repeatedly goes through the steps of removing and processing the first node in the queue. The algorithm terminates when the queue becomes empty, or it could easily be modified to terminate when a certain number of levels in the tree have been generated. Each node $N = (S, P)$ extracted from the queue is processed as follows. If S is a sink state, there is nothing more to be done. Otherwise, by consulting the transition matrix generated by the Markov model, we find all the states S' in which the system can evolve from state S in time Δt. For each state S' we compute the probability P' of entering that state, at the given time, and having followed the given path through the tree, by multiplying the probability P of being in node N by the probability of going from S to S' (i.e., $Prob[S, S']$). If P' is below a chosen threshold, we do not need to expand that path any further. Otherwise, we create a new node $N' = (S', P')$, add it to the list of children of N in the tree, and append it to the queue to be processed at a later time.

The basic DET generated by this algorithm can be quite large due to the many branches emanating from most nodes. However, note that the problem complexity is not NP because the DETs are terminated if: a) scenario time exceeds system mission time, b) controlled/monitored variables fall outside allowed ranges (i.e. system fails), c) scenario probability falls below a user specified number, and, d) if the system is restored to its nominal state. Also, at the cost of a little extra complexity in the algorithm, we can group the DET paths by configuration changes. The states at each level in the tree are grouped by common configuration of system components. Each branch in the DET corresponds to failure of 0 or more of the system components.

5.4.1.2 Algorithm 2

The algorithm in Fig. 5.4.3 shows how we can construct this new DET. In this case, each node in the event tree representation corresponds to a specific configuration of the system components at a specific time. Each node contains a list of all the states in which the system could be at that time and in the particular scenario (path) in the tree and, for each state, the probability of being in that state at that time. All the states in a node share the same configuration of the system components. Two nodes in the tree are linked if it is possible for the system under consideration to go from one of the states in the first node to one of the states in the second node in the time Δt.

```
initialize DET root node to initial state(s) and probability 1
add DET root node to queue Q of nodes to process
while Q is not empty
    remove node N = <(S1,P1),…,(Sk,Pk)> from Q
    initialize A: array [1..number of configurations] of nodes
    for each pair (S,P) in the list of pairs in N
        if S is not a sink state
            for each possible state S'
                if Prob[S,S'] > 0
                    compute probability P' for this branch as Prob[S,S'] * P
                    if P' > epsilon
                        if S' is not in the list of states in node A[Conf(S')]
                            add (S',P') to the list of states in node A[Conf(S')]
                        else
                            add P' to the current probability value associated with S'
                              in the list of states in node A[Conf(S')]
                        end if
                    end if
                end if
            end for each
        end if
    end for each
    add all the nodes in A that contain at least one pair
      to the list of children of N in the DET and to queue Q
end while
```

Figure 5.4.3 Dynamic Event Tree Generation–Algorithm 2

Just like Algorithm 1, Algorithm 2 employs a queue to hold the nodes in the tree before they are processed. In addition, Algorithm 2 uses an array of nodes to keep track, at each iteration, of which configurations have already been generated and which states in each configuration can be reached at that point in time. It starts by creating the root node of the DET containing one or more initial states where all the process variables are in the nominal range and all system components are operational. If there is more than one initial state, each state is assigned a probability so that the sum of the probabilities is 1. The root node is inserted in the initially empty queue. Just like the first, Algorithm 2 repeatedly goes through the steps of removing and processing the first node in the queue, and terminates when the queue becomes empty. The processing of each node, however, requires some extra complexity. It starts by initializing an array of nodes containing one node for each distinct configuration of system components. Then for each state, probability pair (S, P) in the current node for which S is not a sink state, it finds all the states S' in which the system can evolve in the time increment Δt. For each state S', it computes the probability P' of entering that state, at the given time, and having followed the given path through the tree, by multiplying the probability P of being in state S by the probability of going from S to S' (i.e., *Prob*[S,

S′]). If *P′* is below a chosen threshold, that path is ignored. Otherwise, the algorithm adds the pair (*S′*, *P′*) to the node in array A corresponding to the configuration of system components in state *S′* (A[*Conf*(*S′*)]). If a pair with state *S′* is already present, it simply adds *P′* to the current probability associated with *S′*. When all the pairs in the current node have been processed, the algorithm adds all nodes in array A that contain at least one state, probability pair to the list of children of *N* in the tree, and adds the same nodes also at the end of the queue to be processed at a later time.

5.4.2 DET Analysis of a Failure Scenario for the Benchmark System

This section describes the DET analysis of the failure scenario detailed in Section 2.5 and Section 4.3 and presents some results. Here is a summary of the assumptions made on the scenario under consideration:

- Turbine trips
- Reactor is shutdown
- Power *P*(*t*) is generated from the decay heat
- Reactor power and steam flow rate reduce to 6.6% of 3000 MW 10 seconds after the turbine trip
- Feedwater flow is at nominal level
- Off-site power is available
- Main Computer is failed and Backup Computer is in control
- FP fixed at minimum flow and does not fail
- MFV closed and feedwater flow is controlled by the BFV
- There are two Top Events: Low Level and High Level

There are three process variables: level, level error, and compensated level. The two system components controlling the process are the Backup Computer (BC) and the combined BFV-BFV controller. The BC can be in one of three distinct states (see Fig. 4.2.6):

- Operating (OK)
- Loss of inputs (LOSS/IN)
- Down (DOWN)

The combined BFV and BFV controller can be in one of five distinct states (see Fig. 4.3.1):

- Operating (OK)
- Freeze: when it recognizes that BC is down (FREEZE)
- Arbitrary output: an failure occurs inside the controller (ARB/OUT)

- 0 vdc output: the signal from controller to valve is 0.0 (ZVDC/OUT)
- Stuck: a mechanical failure of the valve occurs (STUCK)

In addition, it is necessary to include the BFV position in the model to keep track of the position at which the BFV may be when/if it becomes stuck.

For the purpose of this analysis, the following parameters were used:

- The water level x_n is partitioned into 5 intervals (all measures are expressed in feet):

  -2: $x_n < -2.0$ (Low Level)
  -1: $-2.0 \leq x_n < -0.17$
  0: $-0.17 \leq x_n < 0.17$
  +1: $0.17 \leq x_n \leq 2.5$
  +2: $x_n > 2.5$ (High Level)

- The level error E_{Ln} is partitioned into 3 intervals (all measures are expressed in feet):

  -1: $-1000.0 \leq E_{Ln} < -1.587$
  0: $-1.587 \leq E_{Ln} < 4.203$
  +1: $4.203 \leq E_{Ln} \leq 1000.0$

- The compensated level C_{Ln} is partitioned into 3 intervals (all measures are expressed in feet):

  -1: $-500.0 \leq C_{Ln} < -100.0$
  0: $-100.0 \leq C_{Ln} < 100.0$
  +1: $100.0 \leq C_{Ln} \leq 500.0$

- The BVF position S_{Bn} is discretized into 3 intervals (percentage open):

  0: $0.0 \leq S_{Bn} < 30.0$
  +1: $30.0 \leq S_{Bn} < 70.0$
  +2: $70.0 \leq S_{Bn} \leq 100.0$

- The time increment Δt used is $\Delta t = 1$ second

The number and size of the intervals to partition each process variable and the choice of the time increment Δt are bound by constraints described in Section 4.2.2. Essentially, a finer partition (with a larger number of smaller intervals) can yield a better approximation of the system at a cost of extra computational resources. Furthermore, the time increment is dependent on the size of the cells: too small a time increment may result in the CCMT (Section 4.2.4) not producing useful results if most of the sample points and trajectories fail to leave the starting cell; too large an increment may cause some CCMT trajectories to cross multiple setpoint boundaries. Therefore, it is necessary to determine the partitioning scheme and the time interval by analyzing the actual system.

The partitioning chosen for the level variable is based on the following observations:

- the LOW and HIGH points and intervals were identified in Section 4.2.1
- Section 4.2.1 also points out that it is desirable to keep the level between ± 2 inches of the setpoint, i.e., ± 0.17 feet
- the other intervals for the level variable were added to provide a finer description of the behavior of the variable of primary interest

The partitioning chosen for the BFV position is based on NUREG/CR-6465 [4]. The range of this variable is naturally 0%-100%. The range for level error and compensated level were determined experimentally through simulation of the system. The middle interval of the level error captures the entire range of values of the BFV position variable (which is computed as a function of level error). Finally the partitioning for the compensated level was chosen to minimize the number of intervals while still modeling nominal, low, and high levels for this variable.

Given the partitioning of the process variables, Δt = 1 second was chosen experimentally as a reasonable time increment relative to the size of the process variables intervals.

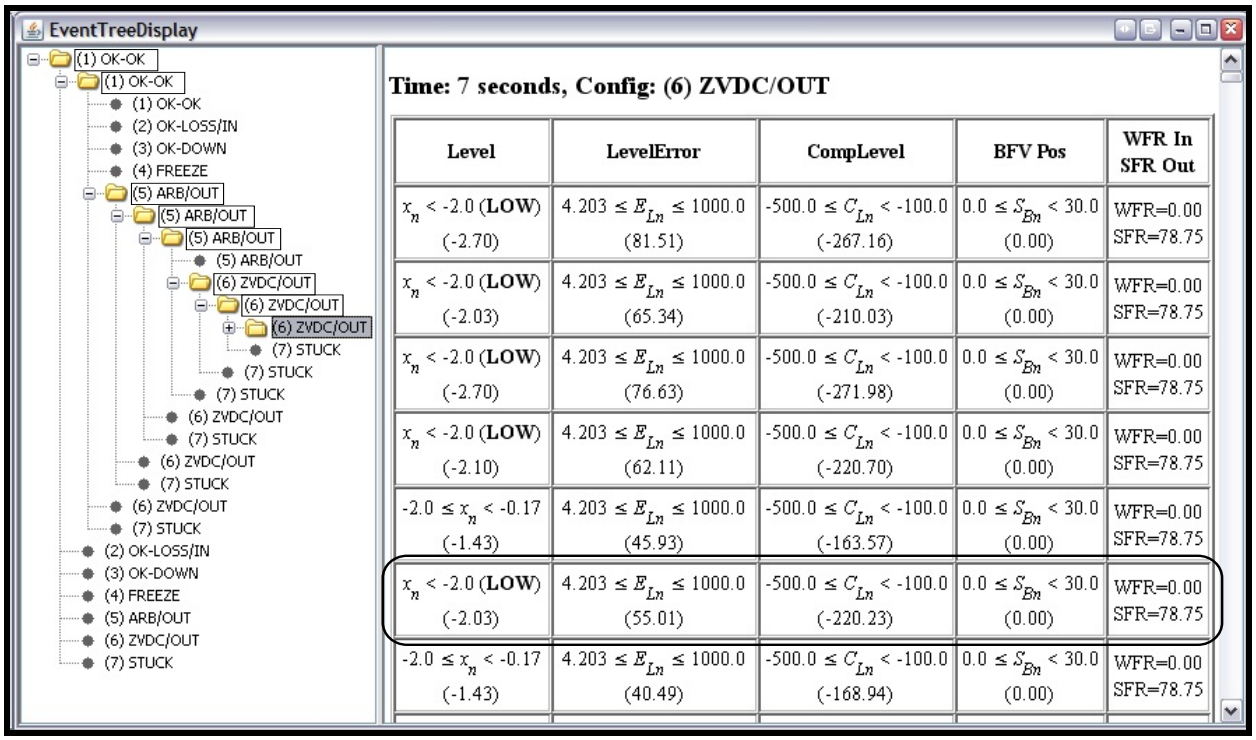

Figure 5.4.4 Display of Part of the Dynamic Event Tree

Figure 5.4.4 shows part of a DET generated for the system. The tool used to generate and display DETs starts from a normal state in which all the system components are operational and the process variables are within their nominal range. It then generates (employing a variant of Algorithm 2 from Section 5.4.1) all possible configurations at the next time step (in this case 1 second), keeping track of all the possible states the process variables may be in at that point in time and in that configuration of the system components.

Figure 5.4.4 shows the tool window: the left pane shows a primitive representation of the event tree and the right pane shows the possible process states for the configuration and time step currently selected in the left pane. Instead of showing the events between branching points (as it is usually done when displaying event trees), the representation of the event tree in the left pane shows the configuration of the control units at each branching point. The event(s) corresponding to a specific branch in the tree can be deduced by comparing the configurations to the left and to the right of the branch. For instance, if in the configuration at the left of a branch both the BFV and the BC are operational (OK-OK) and in the configuration to the right the BC is down (OK-DOWN), the event that has occurred along that branch must be that the BC had a problem and took itself down. At each branching point (or node) in the event tree, the node label shows the state of both the BFV and the BC.

The event tree in the left pane is generated on demand. The top of the tree (displayed in the top-left corner of the left pane in Fig. 5.4.4) represents the normal configuration where all the system components are operational (OK-OK, i.e., the ensemble BFV and BFV controller and the BC are both operating correctly). Whenever the user clicks one of the displayed nodes (branching

points), the program generates all the possible configurations in which the system may evolve in the given time step. For example, there are 7 such possible distinct configurations after the first time step because from the OK-OK state the system can evolve into any of the 7 states (see Fig. 2.5.13). By repeatedly clicking and expanding the tree nodes, the user can explore any possible scenario in the tree. For instance, the (partial) event tree shown in Fig. 5.4.4 corresponds to one possible path (or failure scenario) leading to the level going below the LOW setpoint (dryout).

Boxes in the left pane of Fig. 5.4.4 highlight a possible failure scenario presented in detail in Table 5.4.1. The rounded box in the right pane shows the final failed state for this scenario: the value of each process variable (i.e., level, error level, compensated level, and BFV position) and the value of the steam flow rate (SFR) and feedwater flow rate (WFR). Note that in going from t=2 to t=3 s, a minimum BFV aperture change of 40%/s (difference in the high BFV position of S_{Bn}=30% at t=2 s vs. low BFV position of S_{Bn}=30% at t=3 s in Table 5.4.1) occurs. While this rate of change may be high, no inertia in the valve response was directly taken into account due to the lack of reliable data, and also because it does not impact the ultimate outcome of the scenario. The valve inertia in somewhat indirectly accounted for in the rate of level change which from Table 2.5.1 and Fig. 2.5.1 is 140/109.0 = 1.28 ft/s when the BFV is fully open.

Table 5.4.1 Example Failure Scenario

Time (s)	System Configuration	Process State	Explanation
t = 0	BFV: OK BC: OK	$-0.17 \le x_n < 0.17$ $-1.587 \le E_{Ln} < 4.203$ $-100.0 \le C_{Ln} < 100.0$ $0.0 \le S_{Bn} < 30.00$	Both BFV and BC are in their operational state, and all process variables are in their nominal range
t = 1	BFV: OK BC: OK	$-0.17 \le x_n < 0.17$ $4.203 \le E_{Ln} \le 1000.0$ $-100.0 \le C_{Ln} < 100.0$ $70.0 \le S_{Bn} \le 100.0$	Level error is high, so BFV opens more
t = 2	BFV: ARB/OUT BC: OK	$0.17 \le x_n < 2.5$ $4.203 \le E_{Ln} \le 1000.0$ $-100.0 \le C_{Ln} < 100.0$ $0.0 \le S_{Bn} < 30.0$	BFV controller fails and starts generating arbitrary outputs to the valve, in this case a low value. The level is higher than the nominal level interval.
t = 3	BFV: ARB/OUT BC: OK	$-2.0 \le x_n < -0.17$ $4.203 \le E_{Ln} \le 1000.0$ $-100.0 \le C_{Ln} < 100.0$ $70.0 \le S_{Bn} \le 100.0$	BFV controller is still generating arbitrary outputs, in this case a high value. The level is lower than the nominal level interval..

t = 4	BFV: ARB/OUT BC: OK	$0.17 \leq x_n < 2.5$ $4.203 \leq E_{Ln} \leq 1000.0$ $-100.0 \leq C_{Ln} < 100.0$ $0.0 \leq S_{Bn} < 30.0$	BFV controller is still generating arbitrary outputs, in this case a low value. The level is higher than the nominal level interval.
t = 5	BFV: ZVDC/OUT BC: OK	$-2.0 \leq x_n < -0.17$ $4.203 \leq E_{Ln} \leq 1000.0$ $-100.0 \leq C_{Ln} < 100.0$ $0.0 \leq S_{Bn} < 30.0$	Communication between the BFV controller and the valve is lost; this effectively tells the valve to close completely, and the level is already lower than the nominal level interval
t = 6	BFV: ZVDC/OUT BC: OK	$-2.0 \leq x_n < -0.17$ $4.203 \leq E_{Ln} \leq 1000.0$ $-500.0 \leq C_{Ln} < -100.0$ $0.0 \leq S_{Bn} < 30.0$	The valve remains closed and the level keeps decreasing
t = 7	BFV: ZVDC/OUT BC: OK	$x_n < -2.00$ (**LOW**) $4.203 \leq E_{Ln} \leq 1000.0$ $-500.0 \leq C_{Ln} < -100.0$ $0.0 \leq S_{Bn} < 30.0$	The level falls below the LOW setpoint and the system fails

The event trees constructed for this analysis use a modified version of Algorithm 2. The CCMT employed to determine the possible behaviors of the system starts with a set of sample points (27 in this specific case) located on a regular grid inside the process state space cell representing all the variables being in their nominal range, i.e., $-0.17 \leq x_n < 0.17$, $-1.587 \leq E_{Ln} < 4.203$, and $-100.0 \leq C_{Ln} < 100.0$. The algorithm then follows the evolution of the system through time and through changing configurations of system components (BFV and BC) by always starting the next cell-to-cell mapping from the locations within a cell where it landed at the previous time step. This allows the event tree display tool shown in Fig. 5.4.4 to display not only the intervals in the state space within which each variable is contained at any point in time for a given scenario, but also information about the exact values of the variables. This information is displayed in the left pane below each variable interval (the number in parentheses in Fig. 5.4.4). Note that the tool displays also the current values of the feedwater inflow rate and of the steam outflow rate (the last column in the left pane in Fig. 5.4.4). This allows the user to know at once whether the level is going up (when WFR > SFR) or down (when WFR < SFR).

In addition to generating DETs with the tools described, it is also possible to use the same variant of Algorithm 2 to compute complete DETs to a given depth. Table 5.4.2 summarizes the number of failure scenarios exhibited by the system as a function of the depth of the tree, i.e., the length of time for which the system is analyzed. The percentage of the total number of scenarios for a given depth that lead the system to fail LOW, fail HIGH, and not fail is included in parentheses. 27 sample points on a regular grid within the process state space cell where each variable is in its

nominal range were used in this analysis. The number of possible scenarios grows with the number of sample points employed.

Table 5.4.2 Number of Failure/Non-Failure Scenarios

Time (in seconds) (Depth of DET)	Number of LOW failure scenarios	Number of HIGH failure scenarios	Number of scenarios without failure
1	0 (0.0%)	0 (0.0%)	243 (100.0%)
2	0 (0.0%)	0 (0.0%)	1,242 (100.0%)
3	530 (10.8%)	0 (0.0%)	4,384 (89.2%)
4	1,480 (9.3%)	0 (0.0%)	14,439 (90.7%)
5	4,999 (10.2%)	186 (0.4%)	43,727 (89.4%)
6	14,811 (10.2%)	2,518 (1.7%)	127,292 (88.0%)
7	47,881 (11.5%)	6,531 (1.6%)	362,153 (86.9%)
8	140,644 (11.9%)	18,559 (1.6%)	1,022,695 (86.5%)
9	411,240 (12.3%)	50,259 (1.5%)	2,871,468 (86.2%)
10	1,126,498 (12.0%)	143,922 (1.5%)	8,091,530 (86.4%)

As can be seen from Table 5.4.2, there are a large number of possible scenarios. The majority of scenarios for each DET depth fail to lead the system to failure within the chosen time limit. However, there is still a substantial number of scenarios leading to failure. This is due in part to the presence in the model of a state (ARB/OUT) of the system where the BFV can receive an arbitrary signal from the controller. This is modeled by exploring scenarios for 3 different values of the BFV position, one for each of the 3 intervals in which the BFV position has been partitioned. Another observation is that the number of LOW failure scenarios is always much larger than the number of HIGH failure scenarios. This is due to the existence in the model of a state (ZVDC/OUT) in which the BFV is closed. Whenever the system enters this state, the valve is forced to close and never reopens. Thus the system is bound to fail LOW. Finally, Table 5.4.2 shows that, given the stated initial conditions, the minimum time necessary for the system to fail LOW is 3 seconds and the minimum time for the system to fail HIGH is 5 seconds.

Given the large number of failure scenarios, it is unrealistic to examine them directly. Also, the user may or may not choose to use all these scenarios depending on how the system under consideration is connected to the other plant systems. For example, if the level information for the example DFWCS is not being used by other plant systems, then only the hardware/software/firmware states are relevant. Then it is possible to remove exact timing information and detailed information about the evolution of the process variables to reduce the large number of failure scenarios to a more manageable set of sequences of component failure

events leading to a failure of the system using Algorithm 2 in Section 5.4.1.2. For the model being considered, there are only 64 distinct sequences of component failures that are possible. Table 5.4.3 shows for each sequence and for different lengths of failure scenarios up to 10 seconds the number of failure scenarios that follow the given sequence of component failures.

Table 5.4.3 Classification of Failure Paths

Scenario	Low 3	Low 4	Low 5	Low 6	Low 7	Low 8	Low 9	Low 10	High 5	High 6	High 7	High 8	High 9	High 10
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1-2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1-2-3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1-2-3-4	8	9	9	9	13	26	34	39	2	21	23	24	36	54
1-2-3-4-5	0	3	3	3	51	321	1,116	3,573	9	138	243	462	1,047	2,577
1-2-3-4-5-6	0	0	0	0	32	223	903	3,127	1	25	58	130	313	805
1-2-3-4-5-6-7	0	0	0	0	26	198	889	3,296	0	0	0	0	0	0
1-2-3-4-5-7	0	0	0	0	31	179	667	2,236	1	33	76	204	581	1,658
1-2-3-4-6	0	1	1	1	15	50	101	165	2	21	23	24	36	54
1-2-3-4-6-7	0	0	0	0	29	86	192	388	0	0	0	0	0	0
1-2-3-4-7	0	1	1	1	4	21	41	57	3	46	52	56	98	164
1-2-3-5	24	30	36	87	306	1,044	3,441	10,119	6	63	189	552	1,434	3,957
1-2-3-5-6	0	2	5	46	229	886	3,009	9,190	2	21	63	184	478	1,319
1-2-3-5-6-7	0	0	1	33	206	908	3,158	10,092	0	0	0	0	0	0
1-2-3-5-7	0	2	4	42	174	652	2,156	6,440	3	46	124	405	1,076	3,111
1-2-3-6	8	9	10	23	35	69	101	130	0	0	0	0	0	0
1-2-3-6-7	0	1	2	36	65	164	260	387	0	0	0	0	0	0
1-2-3-7	8	9	9	9	13	26	34	39	2	21	23	24	36	54
1-2-4	8	9	9	9	13	26	34	39	2	21	23	24	36	54
1-2-4-5	24	33	39	90	357	1,365	4,557	13,692	15	201	432	1,014	2,481	6,534
1-2-4-5-6	0	2	5	46	261	1,109	3,912	12,317	3	46	121	314	791	2,124
1-2-4-5-6-7	0	0	1	33	232	1,106	4,047	13,388	0	0	0	0	0	0
1-2-4-5-7	0	2	4	42	205	831	2,823	8,676	4	79	200	609	1,657	4,769
1-2-4-6	8	10	11	24	50	119	202	295	2	21	23	24	36	54
1-2-4-6-7	0	1	2	36	94	250	452	775	0	0	0	0	0	0
1-2-4-7	8	10	10	10	17	47	75	96	5	67	75	80	134	218
1-2-5	24	63	225	624	1,962	5,466	15,750	42,654	6	63	273	939	2,670	7,908
1-2-5-6	8	31	151	476	1,602	4,657	13,614	37,724	2	21	91	313	890	2,636
1-2-5-6-7	0	10	96	381	1,469	4,557	13,693	39,041	0	0	0	0	0	0
1-2-5-7	8	23	118	355	1,158	3,263	9,455	25,920	5	67	249	817	2,296	6,859
1-2-6	8	10	27	38	65	100	137	151	0	0	0	0	0	0
1-2-6-7	8	11	54	95	180	292	422	470	0	0	0	0	0	0
1-2-7	8	9	9	9	13	26	34	39	2	21	23	24	36	54
1-3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1-3-4	8	9	9	9	13	26	34	39	2	21	23	24	36	54
1-3-4-5	24	33	39	90	357	1,365	4,557	13,692	15	201	432	1,014	2,481	6,534
1-3-4-5-6	0	2	5	46	261	1,109	3,912	12,317	3	46	121	314	791	2,124
1-3-4-5-6-7	0	0	1	33	232	1,106	4,047	13,388	0	0	0	0	0	0
1-3-4-5-7	0	2	4	42	205	831	2,823	8,676	4	79	200	609	1,657	4,769
1-3-4-6	8	10	11	24	50	119	202	295	2	21	23	24	36	54
1-3-4-6-7	0	1	2	36	94	250	452	775	0	0	0	0	0	0

1-3-4-7	8	10	10	10	17	47	75	96	5	67	75	80	134	218
1-3-5	24	63	225	624	1,962	5,466	15,750	42,654	6	63	273	939	2,670	7,908
1-3-5-6	8	31	151	476	1,602	4,657	13,614	37,724	2	21	91	313	890	2,636
1-3-5-6-7	0	10	96	381	1,469	4,557	13,693	39,041	0	0	0	0	0	0
1-3-5-7	8	23	118	355	1,158	3,263	9,455	25,920	5	67	249	817	2,296	6,859
1-3-6	8	10	27	38	65	100	137	151	0	0	0	0	0	0
1-3-6-7	8	11	54	95	180	292	422	470	0	0	0	0	0	0
1-3-7	8	9	9	9	13	26	34	39	2	21	23	24	36	54
1-4	8	9	9	9	13	26	34	39	2	21	23	24	36	54
1-4-5	48	96	264	714	2,319	6,831	20,307	56,346	21	264	705	1,953	5,151	14,44 2
1-4-5-6	8	33	156	522	1,863	5,766	17,526	50,041	5	67	212	627	1,681	4,760
1-4-5-6-7	0	10	97	414	1,701	5,663	17,740	52,429	0	0	0	0	0	0
1-4-5-7	8	25	122	397	1,363	4,094	12,278	34,596	9	146	449	1,426	3,953	11,62 8
1-4-6	16	20	38	62	115	219	339	446	2	21	23	24	36	54
1-4-6-7	8	12	56	131	274	542	874	1,245	0	0	0	0	0	0
1-4-7	16	19	19	19	30	73	109	135	7	88	98	104	170	272
1-5	42	222	783	2,310	7,158	20,103	57,354	148,269	6	90	456	1,716	5,241	16,04 7
1-5-6	24	162	637	1,949	6,055	17,100	48,680	128,251	2	30	152	572	1,747	5,349
1-5-6-7	10	114	551	1,821	5,798	16,558	47,290	126,897	0	0	0	0	0	0
1-5-7	24	129	467	1,380	4,237	11,829	33,505	87,206	7	121	496	1,708	5,019	15,08 9
1-6	14	32	54	67	93	134	163	180	0	0	0	0	0	0
1-6-7	24	73	134	181	264	399	491	547	0	0	0	0	0	0
1-7	8	9	9	9	13	26	34	39	2	21	23	24	36	54
Total	**530**	**1,480**	**4,999**	**14,811**	**47,881**	**140,644**	**411,240**	**1,126,498**	**186**	**2,518**	**6,531**	**18,559**	**50,259**	**143,922**

For instance, let us consider LOW failure scenarios of length 10 seconds. In Table 5.4.2, it was stated that there are 1,126,498 such scenarios. The column labeled “Low 10” of Table 5.4.3 shows how many of these scenarios follow each possible sequence of component failures. In particular, the most common sequence is “1-5” (148,269 different scenarios!) . This means that there are 148,269 LOW failure scenarios that result simply from the system going through configurations 1 and 5. Configuration 1 refers to the state in which both the BFV and the BC are operating normally, and it is the initial configuration in all scenarios analyzed. Configuration 5 is the one where the BFV controller has failed and it is essentially sending arbitrary signals to the valve.

As can be seen from Table 5.4.3, there are 64 possible distinct sequences of system configurations that can occur. Of these, 4 sequences (“1”, “1-2”, “1-2-3”, and “1-3”, identified in **bold** in Table 5.4.3) do not result in the system failing (either HIGH or LOW) within the 10 second time interval considered in the analysis. The fact that the system does not fail when everything remains operational (degenerate sequence “1” in Table 5.4.3) is consistent with our expectations: as long as the controller is functional, the system behaves properly. The fact that the system does not fail within 10 seconds when events follow the other three sequences is due to the fact that, in the model (see Fig. 2.5.13), States 2 and 3 do not have self-loops, i.e., they are transitory: whenever the system enters either of these states, it is guaranteed to abandon them at the next time step. Therefore, for sequences that terminate in State 2 or 3, the system must have been in State 1 (and operational) up until the last time step. It is conceivable that at some point in time, the system could fail (HIGH or LOW) within one time step after having transitioned from State 1 to

States 2 or 3. But this seems unlikely, and, as noted, it does not happen within the time interval considered here.

Of the remaining 60 sequences of system configurations that can occur, 40 can result in the system failing HIGH or LOW depending on the exact timing of the events, and 20 can only result in the system failing LOW. The 20 sequences that cannot result in a HIGH failure (identified in *italics* in Table 5.4.3) have all one thing in common: they represent scenarios in which State 6 (ZVDC/OUT) is reached when the level is still below the high setpoint. If that happens, the system cannot fail high because the BFV valve is closed entirely and the level immediately starts to go down. So all the sequences containing State 6 cannot lead to a HIGH failure, except for sequences that end with State 6 by reaching that configuration once the level has already risen above the high setpoint. On the other hand, any sequence where State 6 (ZVDC/OUT) is reached while the level is below the high setpoint is bound to result in the system failing LOW exactly because the BFV valve is closed entirely.

The 40 sequences of system configurations that do not include State 6 (except possibly as the last configuration, reached once the system has already failed) can result in the system failing HIGH or LOW depending on the exact timing and BFV position at the time of failure. If the controller fails in any way, the valve ends up either being stuck at its old value (states OK-LOSS/IN, OK-DOWN, FREEZE, and STUCK) or it can take on arbitrary values (state ARB/OUT). In either case, if the resulting water inflow is greater than the steam outflow, the system will fail HIGH, and if the water inflow is lower than the steam outflow the system will fail LOW (except for the case where the steam outflow manages to go below the water inflow *before* the system fails LOW–in that case, again, the system will end up failing HIGH). In principle, it could be possible for the system to hit State 5 (ARB/OUT) and go on without failure for an arbitrary amount of time. But this is unlikely and would have to rely on the arbitrary output produced by the controller actually working to control the level successfully, a highly unlikely event. In any case, the analysis summarized in Table 5.4.3 clearly shows that both HIGH and LOW failures can occur whenever the system fails in State 5 (ARB/OUT).

The sample analysis presented in this section shows that it is possible to construct DETs from a Markov model of the system. The analysis can also produce qualitative information such as failure paths with exact timing information or ordered sequences of failure events where time information has been removed. Given appropriate failure data for the system components, the analysis of the system based on a Markov model can generate quantitative information such as probabilities for the various scenarios.

5.4.3 DET Incorporation into an Existing PRA

The DET generated from the Markov model can be inserted into a PRA event tree by replacing the appropriate branch in the existing tree.

5.4.4 Outstanding Issues

The main issue is that DETs are generated with exact timing information attached to all paths and nodes, but existing PRA tools such SAPHIRE do not support dynamic methodologies and are not equipped to deal with timing information beyond simple ordering of events. One possible approach to deal with this problem is to time-tag the events in the DET. The DET can then be incorporated into the existing PRA using standard tools (see Chapter 6), and the necessary analysis can be run. Because the PRA tool is unaware of timing and or time-tagging, the prime implicants resulting from the analysis may need to be post-processed to eliminate outputs that violate the timing constraints.

5.5 Comparison of DFM and Markov/CCMT Methodology Results to be Incorporated into the Example Plant PRA

In this section we compare the results of the analyses of the example initiating event (Section 2.5) using the DFM (Section 3.5) and the Markov/CCMT methodology (Section 5.4) for incorporation into SAPHIRE.

5.5.1 Example Initiating Event

For convenience, here is a summary of the assumptions made on the scenario analyzed in Sections 3.5 and 5.4.

- Turbine trips
- Reactor is shutdown
- Power $P(t)$ is generated from the decay heat
- Reactor power and steam flow rate reduce to 6.6% of 3000 MW 10 seconds after the turbine trip
- Feedwater flow is at nominal level
- Off-site power is available
- Main Computer is failed and Backup Computer is in control
- FP fixed at minimum flow and does not fail
- MFV closed and feedwater flow is controlled by the BFV
- There are two Top Events: Low Level and High Level

There are three process variables: level, level error, and compensated level. The two system components controlling the process are the Backup Computer (BC) and the combined BFV-BFV controller. The BC can be in one of three distinct states (see Fig. 2.5.13):

- Operating
- Loss of inputs
- Down

The combined BFV and BFV controller can be in one of five distinct states (see Fig. 2.5.13):

- Operating
- Freeze: when it recognizes that BC is down
- Arbitrary output: an failure occurs inside the controller
- 0 vdc output: the signal from controller to valve is 0.0
- Stuck: a mechanical failure of the valve occurs

In addition, it is necessary to include the BFV position in the model to keep track of the position at which the BFV may be when/if it becomes stuck.

5.5.2 DFM Analysis Results

The DFM methodology was used for two separate analyses: a deductive (or backward) analysis and an inductive (or forward) analysis.

The deductive analysis resulted in two sets of prime implicants that could cause one of the two Top Events (level fails high or low). Examination of the prime implicants showed that for the system and scenario under consideration, any failure of the backup computer or of the combined BFV-BFV controller could result in failure of the system. In particular, any system failure except 0 vdc output of the combined BFV-BFV controller could result in a high level failure if the failure occurs when the feed flow is greater than the steam flow, and any system failure could result in a low level failure if the failure occurs when the feed flow is smaller than the steam flow.

The inductive analysis showed how the DFM model can be used to investigate the behavior of the system once a certain combination of initial component states has been defined. The two sample sequences generated simply confirmed the results predicted by the deductive analysis, i.e, that if the system starts in a state where BFV is stuck, the system could fail high if the position of the BFV is such that the feed flow is lower than the steam flow, and could fail low if the position of the BFV is such that the feed flow is greater than the steam flow.

5.5.3 Markov/CCMT Analysis Results

The Markov/CCMT methodology was used to perform a forward analysis of the example initiating event. A Markov/CCMT model of the system was employed to generate all possible failure scenarios within 10 time steps (10 second) from the initiating event. This analysis revealed the large number of ways in which the system can evolve leading to failure (level high or low). It showed all the sequences of component failure events that can lead to each kind of failure. It also showed that any failure of a system component (backup computer or combined BFV-BFV controller) can result in failure of the whole system and that the exact timing of the component failure, in addition to the kind of failure, is what determines whether the overall system will fail high or low.

5.5.4 Comparison

Although the two methodologies currently present the results of their respective analyses in different forms so that a direct comparison cannot be performed, they clearly agree on the high level, summary assessment of the system failure modes. From both analyses it follows that the example benchmark system can fail as a result of any system component failure. The DFM results emphasize the relative magnitude of feed flow vs. steam flow at the time of the system component failure as the discriminant to decide what kind of failure will occur (high or low); the Markov/CCMT analysis emphasizes that exact timing of the system component failure events will determine the kind of system failure.

These two characterizations of the results coincide with each other in all cases except for two scenarios:

1. A system component can fail when the feed flow is lower than the steam flow, but before the system can fail low, the steam flow (which decreases with time) falls below the now constant feed flow resulting in the system actually failing high.

2. The BFV controller fails in the arbitrary output state when the feed flow is lower than the steam flow, but because of the potentially erratic nature of the BFV controller signal, the feed flow becomes greater than the steam flow before the system fails low and the system ends up failing high.

Neither of these two scenarios is expressed explicitly by the prime implicants resulting from the DFM analysis. However, the Markov/CCMT model can generate failure scenarios that capture these behaviors of the system. The reason why these two scenarios were not identified explicitly by DFM is that the deductive analysis looks for the shortest path, in terms of time steps, that leads to the Top Event. It should be pointed out that the two scenarios of interest would eventually evolve into the conditions expressed in the prime implicants.

It is worthwhile pointing out that there are some differences in the modeling of the initiating event employed by the analyses in the two approaches.

- The DFM model included the steam flow as a modeled, independent variable. This allowed for “time compression”, i.e., a time increment in the DFM analysis can represent an arbitrary large time interval that is determined by the time needed for the system to transition from one level interval to the next. The Markov/CCMT model, instead, used “real time” and considered steam flow a dependent variable determined by Eq. (2.5.8) as a function of time. Time compression allows DFM to analyze the system for a potentially longer time interval, while the number of possible scenarios limits the depth of the DET generated by the Markov/CCMT approach and therefore the length of the time interval that can be explored with this model. However, time compression also eliminates the details of the many scenarios that are possible and thus may remove potentially useful information.

- The DFM model assumed that all failure states of the BC and combined BFV-BFV controller are sink states for these components, while Markov/CCMT used the state transition diagram in Fig. 2.5.13. This caused some discrepancies in the results. For example, DFM generated prime implicants state that the BC experiencing loss of input or going down can result in failure. Markov/CCMT, however, only generated failure paths that must include at least one more configuration change (failure) after the BC experiences loss of input or goes down. That is because in the model described by Fig. 2.5.13, the BC states for loss of input and down are transitory states with no self-loop and the model forces the system to transition to some other state at the very next time step. This explains why failure paths such as [BC and BFV both OK]–>[BC down] are not included in the Markov/CCMT analysis results, but are captured by the DFM analysis prime implicants.

In conclusion, the DFM backward analysis produces a more concise description of the high-level failure behavior of the system. For certain systems, such description may be entirely satisfactory and more manageable than the much more detailed results produced by the Markov/CCMT approach forward analysis. In those cases, the DFM may be the best choice. For other systems or for particular initiating events it may be necessary to obtain detailed information about all possible failure paths and exact timing of the events. In such cases, one may need to appeal to the full power of the Markov/CCMT approach.

6. INTERFACING WITH SAPHIRE

As mentioned in Section 1.2.3, one requirement for a methodology for digital I&C system reliability model construction is that "The methodology must be able to model the digital I&C system portions of accident scenarios to such a level of detail and completeness that non-digital I&C system portions of the scenario can be properly analyzed and practical decisions can be formulated and analyzed" (Requirement 10). From a practical viewpoint, this requirement implies that it should be possible to incorporate the reliability model into an existing PRA.

This chapter illustrates the mechanics of incorporating initiating events of interest for the digitial I&C system under consideration to the existing PRA model. If there are dependencies between the elements of these events and the rest of the plant PRA model, such dependencies will be automatically resolved though the use of a similar naming convention for basic events (see Section 6.3) and Boolean algebra rules by the PRA quantification tool used. If the dynamic control system has no input from other parts of the PRA and has not output to other parts of the PRA, then the control system can be treated as a standalone component and there is no need to insert the dynamic reliability model into the PRA.

The plant model chosen in this report to illustrate how the Markov and DFM model results can be incorporated into an existing plant PRA is a PRA model for a PWR from NUREG-1150 [124]. This PRA has been modeled using the SAPHIRE code [10]. The SAPHIRE code (or the Systems Analysis Programs for Hands-on Integrated Reliability Evaluations), has been developed by the U.S. NRC and the Idaho National Laboratory (INL). SAPHIRE utilizes both graphical and textual interfaces to model PRAs using the ET/ FT method. The code was first developed by INL in the 1980's in order to create a software PRA code for personal computers. The first version was known as the Integrated Risk and Reliability Analysis System, or IRRAS code [142]. Later, several modules were written to complement IRRAS. These modules include *Models And Results Database* (MAR-D), *System Analysis and Risk Assessment* (SARA), and *Fault Tree, Event Tree, and Piping and Instrumentation Diagram* (FEP). These modules were later all integrated into a single package, forming the SAPHIRE code. Several revisions have been performed on SAPHIRE, with the latest release being Version 7. The work detailed in this section has been performed using SAPHIRE, Version 7.26S [7]. While the examples given in this chapter are specific to SAPHIRE, other PRA quantification tools (e.g. such as CAFTA [8] and RISKMAN [9]) have similar graphical and textual interfaces which would allow input in an analogous fashion.

6.1 Description of SAPHIRE

The SAPHIRE code possesses several capabilities. The code allows a user to enter initiating events, such as a loss of coolant accident, and then model the plant's (or other facility or system's) response to these events, calculating a core damage frequency using the ET/FT method. SAPHIRE uses both graphical and logic editors to construct and modify fault trees. After creating a new fault tree with either the graphical or logical method, SAPHIRE will construct this tree using the other method. Thus a tree may be viewed and edited in either graphical or logical format,

irrespective of the method used to model the tree initially. Fault trees may be solved individually, yielding the minimal cut sets for the specified fault tree. The minimum cut sets represent the minimum number of failures that must occur in order for a fault tree top event to occur. In order to quantify an event tree, SAPHIRE must perform a linkage operation in order to connect the event tree to each specified fault tree. When performing the linkage operation, SAPHIRE will internally construct a large fault tree based on the event trees and fault trees used. This fault tree will then be solved to generate the minimum cut sets for each end state in the event tree.

In order to quantify the minimum cut sets, the user must supply failure data for each basic event in the model. Basic events represent events such as motors failing to start or valves failing to open or close as needed. SAPHIRE allows for multiple calculation types to be entered, based on available data (simple probability, mean failure rate, repair rate, etc.). Regardless of what information is entered, SAPHIRE will then calculate a simple probability for the event, and use this number for all other calculations (cut set and end state frequencies). Using the entered failure data, SAPHIRE can then generate and quantify minimum cut sets for fault trees and event tree end states. Quantification will give a failure frequency for a fault tree top event or event tree end state.

Once minimum cut sets have been quantified, additional analyses may be performed using SAPHIRE. These analyses include uncertainty, importance, and sensitivity analysis. An uncertainty analysis allows Monte Carlo and Latin Hypercube methods to be performed, giving a cumulative distribution or probability density curve for the cut set frequency. This analysis is dependent on failure distributions entered by the user for each basic event. Importance analysis uses methodologies including Fussell-Vessley[8], Risk Increase Ratio[9], Risk Reduction Interval[10], and Birnbaum Method[11]. Importance analysis gives the contribution from each event to the overall frequency of the cut set. Finally, a sensitivity analysis may be performed, showing how sensitive a cut set frequency is to changes in a specific basic event [7].

One final feature of SAPHIRE is the ability to import and export virtually any piece of information into or from SAPHIRE using the MAR-D (Models And Results Database) feature. Importable/ exportable information includes fault tree and event tree logic and graphics, basic event information, and cut sets. This feature will enable event trees generated from other sources, such as the DFM and Markov Model methods described previously in Sections 5.3 and 5.4, respectively, to be imported into an existing PRA in SAPHIRE. This feature can also be used to extract cut set information for other uses. The MARD-D window is shown below as Fig. 6.1.1.

---

[8] Fussell-Vessely Importance gives the contribution of a basic event to the minimum cut set frequency.

[9] Risk Increase Ratio gives an indication of how much the minium cut set frequency would increase if the basic event were assumed to occur (probability = 1.0).

[10] Risk Reduction Interval gives an indication of how much the minium cut set frequency would decrease if the basic event were assumed to not occur (probability = 0.0).

[11] The Birnbaum Method gives an indication of the sensitivity of the mimimum cut set frequency to changes in event probability.

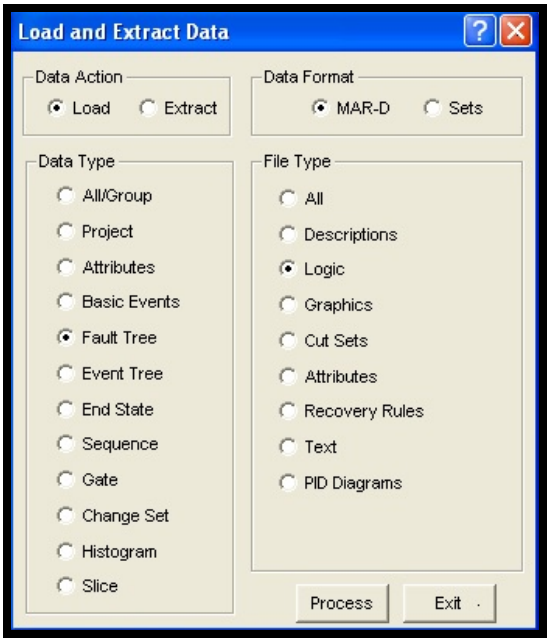

Figure 6.1.1 SAPHIRE MAR-D window

6.2 Model Input Format

As discussed in Section 5.2, output from other models can be entered into SAPHIRE using the MAR-D feature. To reiterate, a file must be created as a text file using a specific format. The file extension will vary based on the type of information to be entered into SAPHIRE. Event tree logic or graphical information is entered using the extension .ETL or .ETG, respectively. To enter fault tree logic information, the extension .FTL is used. The .FTG extension is used to enter fault tree graphical information.

Although the methodologies described in Sections 5.3 and 5.4 result in dynamic event trees, these trees represent a series of AND events, which may be modeled as fault trees. Furthermore, the format for importing fault tree logical information is quite simple, and fault trees may also be easily connected to the existing model through appropriate placement of the model top event. By using the sample fault tree logic format given in Section 5.2 and applying it to an example failure sequence for the digital feedwater controller (recall the sequence presented in Table 5.4.1), a text file (using a .FTL extension) can be created to import the model directly into SAPHIRE:

```
BENCHMARK-DEMO, EXAMPLE-D0 =
EXAMPLE-D0        AND  /BFV-FAILED-T0 /BC-LOSS-OUT-T0
                       CONT /BFV-FAILED-T1 /BC-LOSS-OUT-T1
```

CONT BFV-FAILED-T2 /BC-LOSS-OUT-T2
CONT BFV-FAILED-T3 /BC-LOSS-OUT-T3
CONT BFV-FAILED-T4 /BC-LOSS-OUT-T4
CONT BFV-STUCKCLOSED-T5 /BC-LOSS-OUT-T5
CONT BFV-STUCKCLOSED-T6 /BC-LOSS-OUT-T6
CONT BFV-STUCKCLOSED-T7 /BC-LOSS-OUT-T7

As stated in Section 5.2, BENCHMARK-DEMO is the project database name, which is arbitrary when importing a new file. The model name, DET-D0 in this case, should match the first name given on the second line, which represents the top event of the fault tree. In this example, 'CONT' is used to break up a long sequence of daughter events. Furthermore, a slash ('/') is used to identify complement events, i.e., basic events which must not occur in order for the given sequence to occur. Also note that in this example, the basic events are all time-tagged.

Special recovery rules will need to be written in order to relate these basic events with any non-time-tagged events of the same components already present in the PRA. By using the input format described above, the example failure sequence was successfully entered into SAPHIRE, which then created a graphical representation of the tree, as shown below in Fig. 6.2.1.

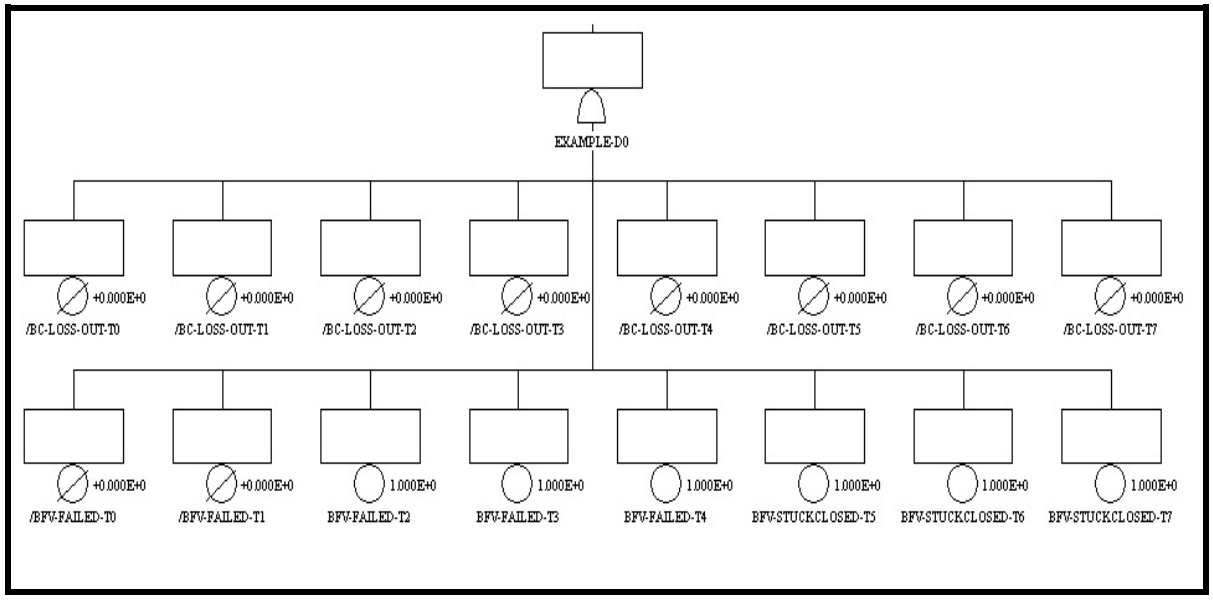

Figure 6.2.1 A Sample Fault Tree Imported into SAPHIRE

The example described above may be expanded to incorporate multiple failure pathways (see Fig. 5.4.4 for some additional failure pathways). To do this, the top event should be replaced as

an OR gate, since any one pathway is enough to lead to a failure of the overall system. Each failure pathway may then be added, following the format described in Section 5.2. While this format works well for small trees, special care must be taken with larger trees, such as those generated through the Markov model approach (discussed in Section 5.4). Such files may contain hundreds of daughters per gate, and will exceed the limits of SAPHIRE's MAR-D format. If the top event (or any other gate) is to have over 50 daughters, it is recommended to break them up using transfer gates. For example, consider the fault tree with 100 daughters, described below:

```
MODEL1, EXAMPLE1 =
EXAMPLE1          OR   EVENT1 EVENT2 EVENT3 .... EVENT99 EVENT100
```

SAPHIRE would be unable to read in all of the daughters, resulting in an incorrect fault tree being imported into the database. By utilizing transfer gates, this fault tree may instead be broken down and imported into SAPHIRE by using multiple files:

```
MODEL1, EXAMPLE1 =
EXAMPLE1          OR EX-TRAN-A EX-TRAN-B
EX-TRAN-A         TRAN
EX-TRAN-B         TRAN

MODEL1, EX-TRAN-A =
EX-TRAN-A         OR    EVENT1 EVENT2 EVENT3 .... EVENT49 EVENT50

MODEL1, EX-TRAN-B =
EX-TRAN-B         OR    EVENT51 EVENT52 EVENT53 .... EVENT99 EVENT100
```

Importing these three files will result in the correct logic appearing in the SAPHIRE database. Fault trees with over 9000 gates and events per file have been successfully loaded into SAPHIRE projects using the MAR-D tool.

6.3 Integrating the Model to the Plant PRA

Once the fault tree (representative of the event trees generated in Section 5.3) has been imported into the PRA, it must be "connected" to the rest of the model. This may be as simple as inserting the top event of the model into the appropriate place in the PRA. It is up to the user to have sufficient understanding of the system to know where the new model fits into the overall system. Care must also be taken to ensure that the newly imported model does not accidentally include names of existing basic events, unless this is explicitly desired. Improperly named events could cause incorrect cut sets to be generated. On the other hand, some events in the new

model may reflect equipment or other events in the plant PRA that are also used by the new model. In this case, it is necessary to ensure that the basic event names used in the new model are exactly the same as those in the old PRA to ensure that the model is appropriately integrated with the PRA.

Figure 6.3.1, below, shows the newly imported DFWCS model appended to an existing fault tree in the model PRA. This figure is identical to Fig. 5.2.5 with the exception of a transfer gate linking to the benchmark DFWCS. As has been mentioned in Section 5.2, two motor-operated valves (MOV-151-E and MOV-151-F) have been assumed to be the same as the MFV and the BFV present in the DFWCS model for the purpose of this proof-of-concept study. In an actual PWR, the AFW control system would be modeled. In Fig. 5.2.5, the DFWCS is actively in control of steam generator A, but SGs B and C are left under analog control. Again, this is done for demonstration purposes due to available data for both the DFWCS (a one SG system) and available PRA models (a 3 SG system). Through the careful naming convention, all components assumed to be identical in both the existing PRA model and the imported DFWCS will be recognized as such by the new model.

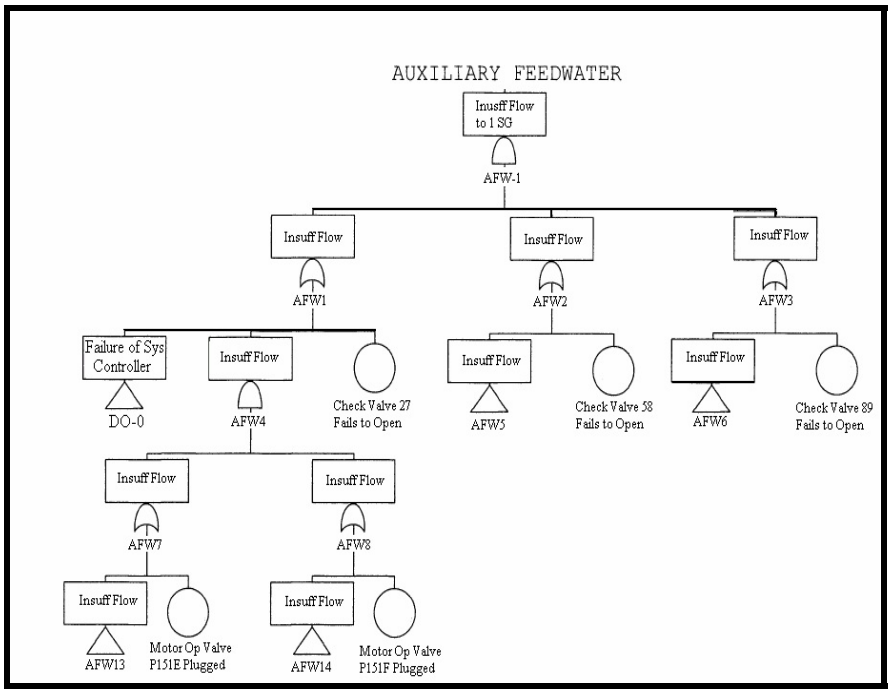

Figure 6.3.1: Appended Fault Tree to Include Imported DFWCS

In coupling quantitative information regarding plant dynamics to a qualitative event in PRA would be to branch out the qualitative event in terms of the conditionals present in the dynamic analysis. For example, the trajectory bifurcation in Fig. 2.5.7 could be possibly represented as a conventional 2-branch ET, with the branches conditional upon the timing of BFV failure, as shown in Figure 6.3.2, below. If the level location is not known, a partitioning similar to that given in Fig.

4.3.1 can be used to represent the level and branchings similar to Fig. 6.3.2 can be constructed for the other level intervals. Note that the timing of the branches is represented with respect to the initiating event. Also, note that the branches are defined in such a way so as to preserve the structure of the conventional ET and the bi-valued logic of the Boolean algebra used in the conventional PRA tools, as opposed to defining three branches originating from the initiating event (i.e. BFV does not fail, BFV fails stuck at t=43 s, BFV fails stuck at t=44 s).

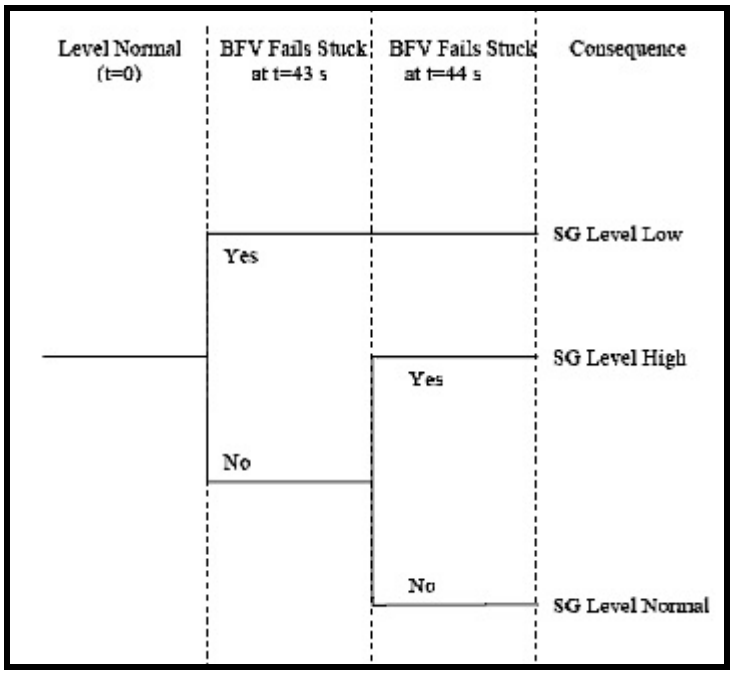

Figure 6.3.2 Incorporation of the trajectory bifurcation in Fig. 2.5.7 into a conventional PRA. Level Normal corresponds to -0.17<x<0.17 ft (see Fig. 4.3.1)

With the model appropriately placed in the plant PRA, minimum cut sets may then be generated as normal in SAPHIRE. However, note that, because some timing or other information pertaining to the dynamic model may be present, the term implicants may be more accurate in describing the output cut sets. Since SAPHIRE would generate the cut sets without regard to the timing information present in the elements of the cut sets and timing inconsistencies may be present in the implicant. In that respect, it may be desirable to export these implicants for post processing using other software or tools. The export can be performed simply by using the MAR-D feature of SAPHIRE as follows: Select the MAR-D feature under Utilities, and extract the desired fault tree, end state, or sequence cut sets to be exported. As with other files associated with the MAR-D feature, this process will create a text file with a .FTC extension (.ESC for event tree end state cut sets, or .SQC for sequence cut sets). The timing inconsistencies can be removed by searching the text file with regard to the impossible changes in the process variables and/or hardware/software/firmware states. For example, if BFV state is ZVDC/OUT at a given time step (i.e. 0 vdc out, see Section 5.4.2) then it cannot be OK in the next time step by Fig. 4.2.8. Since such impossible change information is contained in the state transition matrix of the Markov/CCMT methodology (e.g. Fig. 4.3.2) or the decision tables of DFM (e.g. Table 3.5.13 and

3.5.14) , the search process can be mechanized. A similar search process may be needed to reduce the implicants to prime implicants. It should be noted that the generated minimum cut sets/prime implicants will not be quantified unless failure data for all basic events are entered into SAPHIRE. However, quantification at this stage may not be desired if the prime implicants require post-processing. It is also possible to process the prime implicants and re-import them back into SAPHIRE as cut sets, and then quantify them, using appropriate failure data.

7. UNCERTAINTY QUANTIFICATION

Uncertainties in PRA are classified either epistemic or aleatory [143]. Aleatory uncertainties are uncertainties associated with the stochasticity of events. Epistemic uncertainties are those originating from the imprecise knowledge or modeling of phenomena under consideration. Aleatory uncertainties cannot be reduced through experimentation while epistemic uncertainties can. It is often difficult to distinguish between these two types of uncertainties. Within the context of static ET/FT approach to PRA, the probabilities associated with the branches on the ETs (which depend on failure rates or failure-per-demand) have been usually considered as originating from aleatory uncertainties since failure rates reflect the stochasticity in plant component behavior and the uncertainty in the sequencing of branch points or the tree itself has been regarded as epistemic [144, 145]. However, some works regard uncertainties in failure rates as epistemic [146], since: a) they are parameters of the ET/FT model, and, b) the uncertainty on the failure rates can be reduced by experimentation. According to the latter interpretation [146], the other two types of epistemic uncertainties are the model and completeness uncertainties. Model uncertainty originates from competing models, each of which produces a different approximation of the same reality. Completeness uncertainty represents the uncertainty due to the portion of risk that is not explicitly included in the PRA (e.g., safety culture and organizational behavior) and risks that have not been identified.

For the PRA of digital I&C systems, the distinction between aleatory and epistemic uncertainties is more difficult. For example, in determining the coverage parameters using fault injection (Section 2.4.3), the uncertainty of the coverage parameters can be regarded as epistemic within the interpretation of [146] since:

- a Markov model is used to select the injection points (Section 2.4.5) which may not truly represent the actual operation of the benchmark DFWCS (modeling uncertainty),
- the uncertainty of the coverage parameters can be reduced by increasing the number injections (parametric uncertainty), and,
- the faults injected may not represent all the conditions that may be experienced in actual operation (completeness uncertainty).

On the other hand, the failure rates obtained from the coverage parameters involve using operational plant data or data from data bases (Section 2.4.4) that will at least be affected by the stochasticity in the phenomena leading to hardware failure and, in that respect, will have an aleatory component.

Whether regarded as aleatory or epistemic, common measure of uncertainty is probability and "...probability is fundamentally the same concept regardless of whether it appears in the model of the world or in the subjective distributions for the parameters" [147]. That is why this report will not attempt to classify uncertainties associated with the methods described in Sections 2 through 5 as aleatory or epistemic, but delineate them and discuss how they may be addressed.

For both the DFM or Markov/CCMT methodology, a modeling uncertainty occurs in the representation of the digital I&C system as a finite state machine. This process requires

partitioning the continuous variables such as temperature, pressure, valve aperture and pump speed into mutually exclusive magnitude intervals (cells) and the abstraction of interactions between components, as well as discretizing time. Both the DFM and Markov/CCMT methodology represent the system dynamics as mappings of these cells onto each other in discrete time. In DFM, the mapping is described through decision tables (Section 3.5). The Markov/CCMT methodology describes the mapping through the cell-to-cell-transition probabilities (Section 4.2.4). In that respect, the partitioning has to be such that it is representative of the point(s) selected in the cell to define the mapping. To define the mapping, the DFM uses one or more rows in a decision table to determine the outputs corresponding to the same combination of inputs (Section 3.5.1). However, unlike the Markov/CCMT method, probabilities are not assigned for these multiple outputs. The Markov/CCMT method uses multiple departure point to accomplish the same objective and assigns probabilities for the arrival points to be contained within a given cell (Section 4.2.4 and 4.3.4). For both the DFM and Markov/CCMT method a misrepresentative partitioning may occur when the setpoints and/or Top Event boundaries (see Section 4.2.1 and 4.3.1) do not fall on the boundaries of the cells but on the inside. If the setpoints fall within the cell, then parts of the cells may be representing different dynamics and subsequently the mapping may not reflect the correct behavior of the system. If the Top Event boundaries fall within the cell, then the mapping may incorrectly indicate safe operation when it should indicate failure (or vice versa). Also, if the cells are too large with respect to the time discretization chosen, the system may never leave the cell within the mapping time step (i.e. t in Section 3.5 and Δt in Section 4). In most cases, the impact of the modeling uncertainties of this nature on the PRA can be investigated by progressively refining the time-space partitioning until the predicted system failure characteristics (e.g. Top Event pdfs or Cdfs) do not change with further refinement, at the expense of increased computational demand. In cases where the system dynamics lead to non-compact trajectories in its state space [148], a more effective way for the Markov/CCMT methodology would be to choose the departure points from a cell as arrival points into the cell in the determination of cell-to-cell transition probabilities (Section 4.2.4).

Regarding the abstraction of interactions between components, the communication (or signaling) links between directly connected components are generally modeled as "up" or "down". However, even this simple classification may be hard to determine in actual operation. As an example, consider a digital communication link between the computer and the BFV controller as shown in Fig. 2.1.2. If the encoding scheme used on the digital connection is uni-polar encoding (where, as an example, logic "1" is represented by a non-zero voltage and logic "0" by zero voltage), a continuous level of zero volts may be either due to a failure on the driver side or a long sequence of zeros. An analog connection, such as the connection between the sensor and the computer in the same figure, suffers from similar problems. Assume that a feasible range of voltage values on the analog line is between zero and positive five volts. If the received voltage is zero over a long period of time, it is not possible to classify this as a failure or an actual reading. The detection of failures requires the usage of more advanced solutions, such as adopting return-to-zero encoding for digital or using a valid range of voltages excluding zero volts in the analog connections. Without the knowledge of these details, assumptions about the detectability of failures remain a source of uncertainty. This problem is further exacerbated by the use of communication protocols. Modeling of protocols is a sizable problem in itself. Their incorporation into the reliability modeling can only be done with simplifications hiding numerous details, which contributes to the modeling uncertainty. Similar cases are also observed for the modeling of the computers and other decision points with non-trivial operation. In such devices, critical and well-defined states of operation are explicitly modeled and accounted for in computations. All

other modes of operation, including unforeseen behavior of the hardware and the software, are represented by a common state. In Fig. 2.3.1, the state "E - Arbitrary Output" accounts for all operation modes not accounted for by the other states. Hence, any failure or unexpected operation of the computer, be it benign or disastrous, is represented by this state. Similar arguments can be made about time discretization. Section 2.2. describes the MC and BC waiting for a fixed but unspecified period of time to inform the MFV, BFV and FP controllers when a sensor goes down. This was translated into a single-state waiting period in the state-based model. If longer periods were specified/desired, then more states would be needed to represent the period length (i.e., 2 states for 2 time steps). For the Markov/CCMT methodology, it is necessary to create and use a computer program for the determination of the cell-to-cell-transition probabilities. As such, many issues of the techniques for solving the resulting systems of equations may include numerical precision issues, which can lead to wrong predictions by the model as demonstrated in Section 2.5. Inevitably, time-state discretization increases the uncertainty of the system model. It is worth noting that accounting for all details results in an extremely high number of states and may render the PRA impractical. Therefore, modeling and abstractions must achieve a good balance for realistic representation.

The discussion above on modeling uncertainties assumes that the description of system dynamics that goes as input to both the DFM and Markov/CCMT methodology (simulator) is correct. In practical applications, almost all simulators will contain modeling uncertainties themselves. Both the DFM and Markov/CCMT methodology implicitly account for such uncertainties through representation of probabilistic system dynamics in a discretized state space whose cells can be chosen as bounding covers for the uncertainties.

Once the prime implicants are obtained (Section 5) and been imported into SAPHIRE (Section 6), SAPHIRE can be used to perform both Monte Carlo and Latin Hypercube sampling for uncertainty analysis in the quantification process. In data generation through the approach described in Section 2.4, the sources of uncertainty are:

- Hardware failure data uncertainty
- Modeling uncertainty
- Operational profile uncertainty
- Fault injection experiment uncertainty

The issues associated with hardware failure data uncertainty and their remediation are well known (e.g. see [149, 150]) and will not be addressed here. The other three sources of uncertainty are discussed in Sections 7.1 through 7.3. While some form of uncertainty analysis that accounts for epistemic and aleatory uncertainty such as that in [149, 150] would highlight those areas that need attention, currently we mitigate levels of epistemic uncertainty through trial and error analysis and more conservative assumptions.

7.1 Modeling Uncertainty

In the modeling of complex systems, uncertainty manifests at several levels. At the highest level, there may be uncertainty in the representation of the model. Model uncertainty occurs because models are not perfect representations of the real world. Models make assumptions that are not valid for all situations and conditions. There is no correct model for anything, there is only appropriateness of a model for a given condition and set of circumstances. Each model only represents or predicts the property of interest (in our case the probability of system failure). One interest is the accuracy of the model given the specification of the system. The system may be *under-modeled* where specific states and transitions are left out. In this case, measures of interest may be insufficiently conservative. Conversely, the system may be *over-modeled* including system states and transitions that make model solution intractable or model size too large. Additional uncertainty may be due to the practicality of the modeling assumptions. Some of these assumptions may not hold in *real-world* scenarios. There may also be uncertainty at the parameter level. In many cases, parameters are estimated based on field data or operational profile, expert opinion, or by "good" guessing [151, 152]. The resulting uncertainty at this level complicates model solution and adds to the inaccuracy of the desired measure. Finally, it is to be realized that accounting for model uncertainty is still an emerging and growing field. A universally accepted methodology or method has yet to emerge.

This section addresses the various areas of uncertainty that exist in the quantitative assessment process. Specifically, we discuss uncertainty in the analytical, statistical, and the operational profiles, and the fault injection process. We discuss the epistemic (or reducible) and aleatory (irreducible) uncertainties that manifest in these areas. Some discussion regarding the handling of this uncertainty is presented.

7.1.1 Analytical Model

There are several types of uncertainty that can manifest in the models we have developed. The analytical models presented in Chapter 4 are developed from the system architecture and inter-component dependencies to represent the system failure modes that are of concern. We enumerate the categories of potential uncertainty in these models and address potential methods the uncertainty can analyzed :

1. *Representation of sufficient failure modes*. The failure mode models developed in Chapter 4 were developed by performing a detailed review of the system hazard analysis documents, expert opinion on system of failure modes of the DFWCS, and finally detailed analysis of component interactions under different failure scenarios. However, there may exist some uncertainty in the sufficiency of the failure modes, that is, do they represent a comprehensive set of the classes of failure modes that are of concern? We addressed this problem on several levels. First, we use different models and methods to assess the validity of the failure modes. We use different models using different assumptions. Taken

all together, this allows us to view the problem from several perspectives to ensure that we have appropriately modeled system in it environmental or plant context.

2. *Parameter Data: Hardware Failure Rates*. Failure rate data has uncertainity associated with by it's vary nature. Most failure rate data is collected in one of two ways: (1) from the field or actual operational failure data from the plant, (2) from the failure data of similar devices found in databases. These numbers represent some form of point estimate (e.g. maximum likelihood) and the variance or uncertainty factors associated with the point estimate are usually not given. If they are given they are more a qualitative measure like the uncertainty factor. In contest to this work, we had actual plant data on failures of the components of the DFWCS. We also had access to a commercial database of component failure that helped fill in the gaps where we could not get plant failure data. Finally, we conducted interviews with a vendors to get information on the MTBF of certain components. With this information, we used a Bayesian updating method to estimate the failure rates of the components. What the next step should be in our methodology is to conduct an uncertainty analysis on the failure rates of the models by the method suggested in [152].
3. *Fault Coverage Estimation*. Fault coverage is determined by a statistically controlled experiment. Therefore the uncertainty in the estimate is governed by the statistical methods and the statistics used. Uncertainty in coverage estimates usually comes from several sources. The first is insufficient number of fault injection trials to provide a narrow interval estimate. An narrow confidence interval is required to ensure that the probability that the experiment has mislead you is very low. This uncertainty can be controlled by performing the requisite number of trials as stated in Eqn.(2.4.9). The second type of uncertainty is fault selection. Normally, we select faults that are representative of the faults that would be experienced in the real world. If our fault model is not inclusive of faults that would be experienced in the real world, then coverage estimate is incomplete. We addressed this in our model by considering the set of possible faults from a variety of sources, including our own detailed fault simulation of microprocessors, legacy field data, experience reports, and other research on fault modeling of microelectronic devices. The result is fault model that is comprehensive to the best our ability for the given application

7.1.2 Statistical Model

The statistical model is developed based on those found in published literature and is used to estimate the aforementioned critical model parameters required by the analytical safety model. This statistical model is also used to calculate the number of fault injection experiments required to meet the numerical safety target for a given confidence level. There are limitations to this model. It is well known that exhaustive testing of a system will lead to an exact value for fault coverage. However, this is impractical and infeasible for real I&C systems. Instead, a large number of fault injection experiments are performed to achieve a sufficient level of precision and confidence determined by knowledge of fault occurrence in the system. A greater level of confidence is given to parameter estimates from models that have a strong mathematical foundation and validity. In the past, the method we chose for determining this robustness was through comparison of numerical results from several different statistical models [153]. There is a level of irreducible uncertainty when considering model appropriateness and validation. A

judgment may be made on the accuracy and precision of estimates in an inter-model sense, i.e. one model versus another.

7.1.3 Generic Processor Fault Model

A high-level generic processor fault model is defined to specify the types of faults (and their associated probabilities) that will be injected into the system under analysis. This process builds upon 20 years of research and is undertaken with the goal of characterizing low-level internal processor faults at the higher register-transfer level, in order to demonstrate that faults injected on the actual hardware are representative of the low-level processor faults of concern.

A model is developed that accurately represents the fault-free operation of the system and then tested against the actual system for validity. Once this has been established, fault injection experiments are performed using the model. These results are compared to those obtained from fault injections on the real system. The primary area of concern regarding uncertainty is how well the model represents the real system. Since, in most cases, simulation models for actual processors are not publicly available, an available model that closely resembles the operation of the processor of interest is used to develop the generic model. So the uncertainty issue here is how well is the representativeness and completeness of the generic fault model to the actual fault behavior of the real processor. In [119], we addressed these issues by comparing actual faults on a lower level processor model to a higher level processor and confirming that a large percentage of lower level faults can be represented by a higher level fault model.

7.2 Operational Profile Uncertainty

Operational profiles are used to drive the inputs to the system under analysis during the fault injection process. The operational profiles must be representative of the different system configurations and workloads that would be experienced in actual field operation. Coverage estimation is directly affected by the operational profile of the system. We want to establish baseline coverage for the most used profile (e.g. High Power) and then alter the profiles to represent the different modes of operation and obtain their particular coverage factors. Uncertainty arises when we consider *abnormal* operational profiles corresponding to irregular system and fault-handling mechanism behavior. These random events are unpredictable and are due to unforeseen system demands, for example. It may be mitigated using sampling of historical plant data. This is a good representation of modes and mode changes. We must assume some variability exists in this data, however, and sample from a known distribution that contains the point data. When the distribution is not known, we may apply *Statistics of the Extremes Theory* [154, 155].

In [153], it is pointed out that fault coverage estimation was possible if there was precisely detailed knowledge of the fault distribution. Otherwise, one would have to assume that the fault tolerance mechanism was equally fair with faults occurring as they would in the real system and during the fault injection process. The occurrence of a fault may be a rare event, not showing up

during the operational lifetime of the system. Again, this may be due to variable operating environment conditions, failures in the fault-handling mechanisms, or fluctuations in their behavior. Statistics of the extremes theory studies the statistical distribution of the extreme values (corresponding to rare events) found in a sample extracted from a population with a generic distribution. It relaxes the two assumptions that are typically made when estimating coverage in a classical sense: (1) that fault tolerance mechanisms behave deterministically for a given fault, and (2) that the fault distribution is a well-defined characteristic of the system. The non-deterministic behavior of the fault tolerance mechanism is related to the randomness of the operational state of the system at the moment of occurrence of the fault. Thus, by reproducing the typical activities that will be performed by the system in the experimental process during its operational lifetime, one is able to isolate the dimension of the fault space describing the system's operational profile and account for the associated randomness thereby minimizing aleatory uncertainty.

7.3 Fault Injection Experiment Uncertainty

The fault injection process assumes that (1) the faults to be injected into the system are representative of the actual faults that occur within the system and (2) the additional software required to inject the faults does not affect the functional behavior of the system in response to the injected fault. Essentially, the assumption states that the software that is used to inject the fault is independent of the rest of the system, and that any faults present in the fault injection software will not affect the system under analysis. In our actual setup, we have found these assumptions to be somewhat valid. Our fault injection environment consists of:

- An ICE
- A host computer which runs control software for the ICE machine
- A sequencer, implemented in software, that controls the fault injection process and data collection
- The target machine (DFWCS prototype)

While this configuration is simple, there are several layers of timing complexity and several entry points of random (aleatory) uncertainty. One of the advantages to software-based fault injection is that experiments can be run in near real-time, allowing for the possibility of running a large number of fault injection experiments. We have found that this is not the case. There are limitations at the hardware level (e.g. ICE machine) that prohibit true real-time execution. Early in the development of the FI environment, it was difficult to determine if the injected faults were really being exercised. Corruptions of specific memory addresses were expected to yield predictable results; however, in many cases, there was no response from the system. It was finally determined that memory and register corruptions needed to take place immediately after those locations were accessed (i.e. written to or read from). This would ensure that the fault would take effect. This form of fault injection requires the system (or processor) to be halted so that the corruption can be made and then restarted. Although this adds some time to the execution of instructions, it is negligible and does not wholly corrupt the results.

7.4 Summary

We have addressed several forms of uncertainty in this chapter. In the literature there appears to be mature methods to deal with parameter uncertainty and are considered in this study[151, 152]. Model uncertainty is a more difficult endeavor. Techniques suggested in [149, 150] also seem to offer some promise to account for epistemic and aleatory.

8. PROPOSED BENCHMARK, PROCEDURES AND THE REQUIREMENTS FOR THE RELIABILITY MODELING OF DIGITAL INSTRUMENTATION AND CONTROL SYSTEMS

The benchmark problem in Section 2 was defined such that it satisfies most of the requirements given in Fig.8.1 below [90], as well as being representative of the digital SG feedwater control systems used in operating PWRs. In Fig.8.1, loosely-control coupled systems are defined as those systems with no direct dependency among different events occurring among the system constituents, including software/firmware but in which hardware/software/firmware may communicate only indirectly, through the physical process variables (such as temperature or pressure) and may not compete for resources. For example, Eqs. (2.2.2), (2.2.3), (2.2.8) and (2.2.11) in Section 2.2.2 show how the MFV and BFV communicate through the SG level.

Loosely-Control Coupled Benchmark System Requirements

1 A clock which regulates information sampling from the controlled/monitored process,
  - 1.1 regulates measurements,
  - 1.2 may lead to roundoff,
  - 1.3 may lead to truncation.

2 Explicit representation of the power requirements that are needed for the digital systems including
  - 2.1 loss of power,
  - 2.2 low power,
  - 2.3 power spikes.

3 Real-time constraints

4 A polling capability with
  - 4.1 events occurring in between polls,
  - 4.2 sensors that are being polled failing to report a value

5 An interrupt capability with
  - 5.1 interrupts occurring simultaneously,
  - 5.2 interrupts occurring at an excessive rate,
  - 5.3 unused interrupts that may be activated.

6 Long term storage with
  - 6.1 failures that can occur in the retrieval of information,
  - 6.2 failures that can occur in the saving of information,
  - 6.3 Loosely-Coupled Requirement 2,
  - 6.4 Loosely-Coupled Requirement 3.

7 Computation capability both based on the controlled/monitored process physics and interacting with the process physics
  - 7.1 stimulates interaction with the physical process
  - 7.2 can produce intermittent and functional failures

8 A self-diagnostic system where
  - 8.1 contradictory data can be delivered to the system,
  - 8.2 events can occur while in self-diagnostic mode.

9 A watchdog timer with
  - 9.1 instances in which there is no safe state,
  - 9.2 instances in which the watchdog timer fails.

Tightly-Control Coupled Benchmark System Requirements

1 Include Loosely-Control Coupled Requirements

2 Networking capability with
  - 2.1 failures in the networked systems,
  - 2.2 failures in connecting components (wires, routers, etc.),
  - 2.3 failures of any protocol used,
  - 2.4 failures as a result of the network topology,
  - 2.5 transient failures in the network.

3 Analog backups to digital systems that include failures in which either the digital or analog system has failed

4 Shared memory with failures which involve
  - 4.1 data races,
  - 4.2 both deadlocks and starvation.

5 Shared external resources with
  - 5.1 failures involving both deadlocks and starvation,
  - 5.2 network failures.

6 Fault tolerance capability to test Byzantine failures

7 A database with
  - 7.1 Loosely-Control Coupled Requirement 6,
  - 7.2 failures that can force the database to be inconsistent.

8 Capability to simulate different configurations/ versions of software installed on each of the duplicated components and shared resources, including all permutations of homogeneous and heterogeneous software and/or hardware.

Figure 8.1 Digital I&C Benchmark System Requirements Given in [90]

For example, the MFV position (Eq. (2.2.11)) depends on flow demand (Eq. (2.2.2)) through Eq. (2.2.8) which in turn depends on compensated water level through Eq. (2.2.3) as well as the BFV demand through Eqs. (2.2.2) and (2.2.5). Similarly, FP speed (Eq. (2.2.10)) depends on the flow demand (Eq. (2.2.2)) which in turn affects the MFV position through Eqs. (2.8) and (2.2.11). Tightly-control coupled systems include loosely-control coupled system requirements, and, in addition, have direct communication between different events in the overall system. For example, Table 2.3.1 shows that if roundoff/truncation/sampling rate errors occur in the MC, it can be detected by connecting components (e.g. MFV and BFV controller) if output is out of range or exceeds the physically possible rate and the connecting component fails the MC and switches the control to BC. The requirements in Fig. 8.1 originating from the features of the digital I&C systems found in practice but not relevant to those used in the current nuclear reactor protection and control systems are not represented by the benchmark system (e.g. networking, shared external resources). One particularly challenging feature of the benchmark system from a

reliability modeling viewpoint is that modeling of some of its fault tolerance capabilities requires consideration of the system history. For example, when both the MC and BC have failed, FP speed as well as MFV and BFV positions are determined from system history data (see Section 2.2.2). Also, Fig. 2.5.7 shows that system failure mode may depend on the exact timing of failure events, and not just the order of failure events.

Regarding how the Markov methodology and the DFM meet the requirements listed in Section 1.2.3, they both clearly satisfy Requirements 1, 2, 3, 4, 7, 8, 9 and 10:

- Neither methodology is based on purely operating experience and both have been tested on both loosely and tightly control-coupled systems (e.g. [4, 126, 129, 130, 156]). In that respect, both methodologies predict encountered and future failures well (Requirement 1).
- Both models can account for all the features of the benchmark system which is representative of the digital SG feedwater control systems used in operating PWRs as well as containing the features of digital I&C systems used in nuclear power plants, in general (Requirement 2).
- Both models make valid and plausible assumptions (Requirement 3). For example, the assumption that process dynamics can be represented through a Markov transition matrix or a decision table (of DFM) has been demonstrated through previous work [2, 6]. Similarly, the normal operation of the benchmark system and its assumed failure modes were based on operating PWRs as well as other digital I&C systems encountered in practice. Both methodologies can account for all the features of the benchmark system (Sections 3 and 4).
- Sections 3 and 4 show that both models can quantitatively represent dependencies between failure events accurately (Requirement 4).
- Sections 3 and 4 show that both methodologies can differentiate between a state that fails one safety check and those that fail multiple ones (Assumption 7), as well as between faults that cause function failures and intermittent failures (Requirement 8).
- Sections 5 shows that both methodologies have the ability to provide relevant information to users, including cut sets, probabilities of failure and uncertainties associated with the results (Requirement 9).
- Section 6 shows that both methodologies can model the digital I&C system portions of accident scenarios to such a level of detail and completeness that non-digital I&C system portions of the scenario can be properly analyzed and practical decisions can be formulated and analyzed (Requirement 10).

The challenges that need to be addressed with the methodologies are reflected in Requirements 5, 6 and 11. Both methodologies have substantially steeper learning curves and are more labor intensive (currently, Markov methodology may be somewhat more so than DFM) than the conventional ET/FT methodology. So it is not so simple for an analyst to learn the concepts and implement the methodologies in a straightforward manner (Requirement 5). On the other hand, the conventional ET/FT methodology cannot correctly represent the complex interactions and time dependent behavior of some of the digital reactor protection and control systems as exemplified by the benchmark system.

Another challenge with the proposed methodologies is that the failure data used by either methodology for quantification is not necessarily credible to a significant portion of the technical community (Requirement 6). This challenge is endemic to the nature of the digital systems rather than being specific to the proposed methodologies. There is no consensus in the technical community on how software reliability should be quantified and, in fact, whether such a concept is appropriate at all [2]. The proposed methodologies try to address this challenge by regarding the digital system as a whole and not as a sum of separate hardware and software, consistent with the conclusion of the National Research Council [16]. A similar philosophy is used in the failure data generation procedures described in Section 2.4. Chapter 7 indicates some techniques that may be used to quantify the uncertainty in failure data acquisition using the approach described in Section 2.4. It should be mentioned that the proposed methodologies can be used to obtain qualitative information on the failure characteristics of digital I&C systems (i.e. prime implicants) as well as quantitative as shown in Chapter 5. In that respect, they can be helpful in the identification of risk important event sequences even if the data issue is not resolved and their quantitative results may not have universal acceptability.

Finally, whether the proposed methodologies require highly time-dependent or continuous plant state information or not (Requirement 11) depends on the problem under consideration. For the benchmark system, such information seems to be necessary for correct reliability modeling of the system. Section 2.5 shows that system failure mode depends on the exact timing of BFV getting stuck. On the other hand, Chapters 3 and 4 show that both methodologies can be also used for simple description of the connectivity between events if the correct system behavior under normal and abnormal operation can be inferred from qualitative arguments only.

9. SUMMARY AND CONCLUSION

As follow up to NUREG/CR-6901 [2], this report

- presents a benchmark system that can be used to assess the methodologies proposed for the reliability modeling of digital I&C systems using a common set of hardware/software/firmware states, and,
- illustrates how DFM and the Markov/CCMT methodology can be used for the reliability modeling of the benchmark system.

The benchmark system specification includes procedures for system component failure mode identification and failure data acquisition. The benchmark problem contains all the features of the digital I&C systems relevant to those used in nuclear power plants, as well as being representative of the digital SG feedwater control systems used in operating PWRs. The DFM and the Markov/CCMT methodology were identified by NUREG/CR-6901 as the methodologies that rank as the top two with most positive features and least negative or uncertain features when evaluated against the requirements for the reliability modeling of digital I&C systems.

Using an example initiating event, it is shown that the DFM and Markov/CCMT methodologies can account for all the features of the benchmark system and that the results can be integrated into an existing PRA. Possible challenges with the methodologies include:

1. analyst skill levels needed for the implementation of the methodologies,
2. computational demand for the correct description of the coupling between failure events, and,
3. acceptability of the data used for quantification by a significant portion of the technical community.

These challenges originate from the complexity and diverse nature of the phenomena to be accounted for and are not specific to DFM or the Markov/CCMT methodology. Another challenge is the limitation in the capabilities of the existing ET/FT based plant PRA tools. Currently, most plant PRA tools cannot distinguish between the timing of events. As indicated in Chapter 6, this limitation may require post-processing of the results obtained from plant PRAs after the integration of the digital I&C system reliability models to remove timing inconsistencies between minimal cut set events.

It may be possible to alleviate limitations posed by Challenges 1 and 2 by appending/modifying the existing ET/FT based plant PRA tools. For example, the Markov/CCMT methodology described in Chapter 4 and the procedure described in Section 5.3 can be implemented on the SAPHIRE (or other automated ET/FT tools) platform to reduce, respectively, the analyst's burden of constructing the Markov model and converting the results to DETs. Software already exists

for mechanized construction of such Markov models (e.g. [85]). Post-processing of the minimal cutsets could be also part of this appended package. Similar arguments can be made for DFM. The limitation posed by Challenge 2 may be alleviated using distributed computing environments.

Challenge 3 is perhaps the most difficult to address. As discussed in some detail in NUREG/CR-6901 [2] there is no consensus in the technical community on how software reliability should be quantified and in fact whether such a concept is appropriate at all. Also as discussed in NUREG/CR-6901, the data challenge is not specific to the methods described in this report but to all existing methods. However, the proposed methodologies can be used to obtain qualitative information on the failure characteristics of digital I&C systems (i.e. prime implicants) as well as quantitative, and, in that respect, can be helpful in the identification of risk important event sequences even if the data issue is not resolved.

One other challenge is that both the Markov/CCMT methodology and the DFM may require highly time-dependent or continuous plant state information for correct reliability modeling of the system failure modes if the system failure modes depend on the exact timing of the events. Since an existing plant PRA is often based on the ET/FT approach (and has no time information), such highly time-dependent plant states may make the integration of the digital I&C system reliability model into the existing PRA difficult. For example, several initial conditions may need to be considered in the form of an event tree with each branch providing the starting point for a Markov of DFM analysis (e.g. see Fig.6.3.2). On the other hand, as indicated in Section 7, both methodologies can be also used for simple description of the connectivity between events if the correct system behavior under normal and abnormal operation can be inferred from qualitative arguments only.

Finally, the properties of the benchmark system considered in this study may not apply to all the reactor protection and control systems in nuclear power plants. For digital I&C systems which may have less complex interaction between the failure events, the conventional ET/FT approach may be adequate for the reliability modeling of the system. It is also important to note that the report presents only a proof-of-concept study. Additional work is needed to validate the practicality of the proposed methods for other digital systems and resolve the challenges identified.

Research to resolve these challenges to the practical implementation of dynamic methods would involve the following:

1. A stand-alone reliability modeling of the full benchmark system using the DFM, Markov/CCMT methodology and the conventional ET/FT approach.
2. Qualitative comparison of the event combinations that lead to the benchmark system failure as obtained by the DFM, Markov/CCMT methodology and the conventional ET/FT approach.
3. Quantitative evaluation of the models in Item 1 using data obtained through the procedure described in Section 2.4. as well as other means (e.g. field data, data libraries).

4. Incorporation of models in Item 1 into an existing PRA for selected initiating events (e.g. turbine trip, station blackout, loss of main feedwater).
5. Specification of another benchmark problem reflecting the operation of the reactor protection system.
6. Performing Items 1 through 4 for the new benchmark problem.
7. Software development for mechanized construction of Markov models and dynamic flowgraphs.
8. SAPHIRE extension for in-line post-processing of the minimal cutsets.
9. Implementing Items 7 and 8 using distributed computing environments.

10. REFERENCES

[1] Guideline for Performing Defense-in-Depth and Diversity Assessments for Digital I&C Upgrades - Applying Risk-Informed and Deterministic Methods, 1002835, EPRI, Palo Alto, CA (2004)

[2] T. ALDEMIR, D. W. MILLER, M. STOVSKY, J. KIRSCHENBAUM, P. BUCCI, A. W. FENTIMAN, and L. M. MANGAN, Current State of Reliability Modeling Methodologies for Digital Systems and Their Acceptance Criteria for Nuclear Power Plant Assessments, NUREG/CR-6901, U. S. Nuclear Regulatory Commission, Washington, D.C. (2006)

[3] C. J. GARRET and G. E. APOSTOLAKIS, "Automated Hazard Analysis of Digital Control Systems", *Reliab.Engng & System Safety,* **77**, 1-17 (2002).

[4] S. GUARRO, M. YAU, and M. MOTAMED, Development of Tools for Safety Analysis of Control Software in Advanced Reactors, NUREG/CR-6465, U.S. Nuclear Regulatory Commission, Washington, D.C. (1996)

[5] T. ALDEMIR, "Utilization of the Cell-To-Cell Mapping Technique to Construct Markov Failure Models for Process Control Systems", G. APOSTOLAKIS (Ed.), *Probabilistic Safety Assessment and Management: PSAM1*, 1431-1436, Elsevier, New York (1991).

[6] T. ALDEMIR, "Computer-Assisted Markov Failure Modeling of Process Control Systems", *IEEE Transactions on Reliability*, **R-36**, 133-144 (1987).

[7] C. L. SMITH, J. KNUDSEN, M. CALLEY, S. BECK, K. KVARFORDT and S. T. WOOD, *SAPHIRE Basics: An Introduction to Probability Risk Assessment Via the Systems Analysis Program for Hands-on Integrated Reliability Evaluations (SAPHIRE) Software*, Idaho National Laboratory, Idaho Falls,ID (2005).

[8] CAFTA For Windows, Version 3.0c, SAIC, Los Altos, California (1995)

[9] RISKMAN 7.1 for Windows, ABS Consulting, Irvine, California (2003)

[10] K. D. RUSSELL, C. L. HOFFMAN, K. J. KVARFORDT, E. LOIS, C. L. SMITH, and S. T. WOOD, Systems Analysis Programs for Hands-on Integrated Evaluations (SAPHIRE) Version 6.0, System Overview Manual, NUREG/CR-6532, U.S. Nuclear Regulatory Commission, Washington, D.C. (1999)

[11] G. J. PAI and J. DUGAN, "Automatic Synthesis of Dynamic Fault Trees From UML System Models", *Proceedings of the 13th International Symposium on Software Reliability Engineering (ISSRE '02)*, 243-254, IEEE Computer Society, Washington, D.C. (2002).

[12] N.ADDOUCHE, C.ANTOINE and J MONTRAIN, "Combining Extended UML Models and Formal Methods to Analyze Real Time Systems", R.WINTHER, B.A.GRAN and G.DAHL (Eds.), *SAFECOMP 2005*, 24-27, Springer-Verlag, Berlin, Germany (2005).

[13] S.BERNARDI, S.DONATELLI and J.MERSEGUER, "From UML Sequence Diagrams and Statecharts to Analysable Petri Net Models", S.BALSAME, P.INVERARDI and B.SELIC (Eds.), *Proceedings of the 3rd international workshop on Software and performance*, 35-45, ACM Press, New York (2002).

[14] L.BARESI and M.PEZZE, *On Formalizing UML with High-Level Petri Nets*, p 276, G.A.AGHA., F DECINDIO and G.ROZENBERG, Eds., Springer-Verlag, Berlin, Germany (2001).

[15] *Update of Chapter 7 (I&C) of NUREG-0800, Standard Review Plan*, U.S. Nuclear Regulatory Commission, Washington, D.C. (1997).

[16] NATIONAL RESEARCH COUNCIL, Digital Instrumentation and Control Systems in Nuclear Power Plants: Safety and Reliability Issues, National Academy Press, (1997)

[17] *ACRS Letter Report to L. Joseph Callan, "Regulatory Guidance on the Implementation of Digital I&C Systems* (1997).

[18] T. ALDEMIR, N. SIU, A. MOSLEH and P. C. CACCIABUE, *Reliability and Safety Assessment of Dynamic Process Systems*, Springer-Verlag, Heidelberg (1994).

[19] D. W. MILLER, E. L. QUINN, S. A. ARNDT, L. J. BOND, D. B. JARRELL, OHARE, J. M. and R. T. WOOD, *Instrumentation, Controls and Human-Machine Interface (IC&HMI) Technology Workshop*, U. S. Department of Energy, Gaithersburg, Maryland (2002).

[20] *Hard-Coded* (2007).

[21] *Hard-Coded* (2007).

[22] R. L. CAMPBELL, *What Is Built-In Self Test And Why Do We Need It?*, Nelson Publishing, Unc. (1996).

[23] J. BULLOCK, *Ladder Logic*, Seattle Robotics Society, Seattle, WA (1997).

[24] R. WALKER, *Intel 8088 Central Processing Unit* (1980).

[25] D. JOHNSON, *The Sampling Theorem* (2004).

[26] Part 21 Reports, LD-99-036, (1999)

[27] Part 21 1999-34-1, (1999)

[28] Event Number 39392, (2002)

[29] Part 21 2002-32-0, (2002)

[30] EA-96-442, (1996)

[31] Information Notice 93-49, (1993)

[32] Power Reactor Event Number 36518, (1999)

[33] J. RUSHBY, *Critical System Properties: Survey and Taxonomy*, pp 189-219 (1994).

[34] J. DEVOOGHT and C. SMIDTS, "Probabilistic Reactor Dynamics I: The Theory of Continuous Event Trees", *Nuclear Science and Engineering*, **111**, 229-240 (1992).

[35] B. TOMBUYES and T. ALDEMIR, "Dynamic PSA of Process Control-Systems Via Continuous Cell-To-Cell-Mapping", 1541-1546, Elsevier, New York (1996).

[36] A. AMENDOLA and G. REINA, DYLAM-1, A Software Package for Event Sequence and Consequence Spectrum Methodology, EUR-924, CEC-JRC, ISPRA, Commission of the European Communities, (1984)

[37] G. COJAZZI, "The DYLAM Approach to the Dynamic Reliability Analysis of Systems", *Reliab.Engng & System Safety*, **52**, 279-296 (1996).

[38] C. ACOSTA and N. SIU, "Dynamic Event Trees in Accident Sequence Analysis: Application to Steam Generator Tube Rupture", *Reliab.Engng & System Safety*, **41**, 135-154 (1993).

[39] C. SMIDTS and S. SWAMINATHAN, "Improvements to Discrete Dynamic Methodologies", *PSA-96*, 159-166, American Nuclear Society (1996).

[40] H. KAE-SHENG and A. MOSLEH, "The Development and Application of the Accident Dynamic Simulator for Dynamic Probabilistic Risk Assessment of Nuclear Power Plants", *Reliab.Engng & System Safety*, **52**, 297-314 (1996).

[41] J. M. IZQUIERDO, J. HORTAL, J. SANCHES-PEREA and E. MELENDEZ, "Automatic Generation of Dynamic Event Trees: A Tool for Integrated Safety Assessment", T. ALDEMIR, N. SIU, A. MOSLEH, P. C. CACCIABUE and B. G. GOKTEPE (Eds.), *Reliability And Safety Assessment of Dynamic Process Systems*, **120**, 135-150, Springer-Verlag, Heidelberg (1994).

[42] S. MARCHAND, B. TOMBUYES and P. LABEAU, "DDET and Monte Carlo Simulation to Solve Some Dynamic Reliability Problems", *PSAM 4*, **3**, 2055-2060, New York (1998).

[43] Y. DUTUIT, "Dependability Modeling and Evaluation by Using Stochastic Petri Nets: Application to Two Test Cases", *Reliab.Engng & System Safety*, **55**, 117-124 (1997).

[44] J. L. PETERSON, "Petri Nets", *ACM Computing Surveys*, **9**, 223-252 (1977).

[45] S. SWAMINATHAN and C. SMIDTS, "The Mathematical Formulation of the Event Sequence Diagram Framework", *Reliab.Engng & System Safety*, **65**, 103-118 (1999).

[46] T. MATSUOKA and M. KOBAYASHI, "An Analysis of a Dynamic System by the GO-FLOW Methodology", P. C. CACCIABUE and I. A. PAPAZOGLOU (Eds.), *Probabilistic Safety Assessment and Management '96*, 1547-1436, Elsevier, New York (1991).

[47] T. MATSUOKA and M. KOBAYASHI, "GO-FLOW: A New Reliability Analysis Methodology", *Nuclear Science and Engineering*, **98**, 64-78 (1988).

[48] M. MARSAN and G. CONTE, "A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems", *ACM Transactions on Computer Systems*, **2**, 93-122 (1984).

[49] R. MOCK, "The Usage of Generalized Stochastic Petri Nets for Reliability Assessments of Passive Safety Installations of SWR 1000 - A Pilot Application", S. KONDO and K. FURUTA (Eds.), *PSAM 5 - Probabilistic Safety Assessment and Management*, 1615-1621, Universal Academy Press Inc, Tokyo, Japan (2000).

[50] L. M. KAUFMAN and B. W. JOHNSON, Embedded Digital System Reliability and Safety Analyses, NUREG/GR-0020, U.S. Nuclear Regulatory Commission, Washington, D.C. (2001)

[51] D. T. SMITH, T. A. DELONG and B. W. JOHNSON, "A Safety Assessment Methodology for Complex Safety-Critical Hardware/Software Systems", *International Topical Meeting on Nuclear Plant Instrumentation, Controls, and Human-Machine Interface Technologies*, Washington, D.C. (2000).

[52] J. D. LAWRENCE, Software Reliability and Safety in Nuclear Reactor Protection Systems, UCRL-ID-114839, Lawrence Livermore National Laboratory, Livermore, California (1993)

[53] M. CEPIN and B. MAVKO, "A Dynamic Fault-Tree", *Reliab.Engng & System Safety*, **75**, 83-91 (2001).

[54] J. D. ANDREWS and J. DUGAN, "Dependency Modeling Using Fault-Tree Analysis", *Proceedings of the 17th International System Safety Conference*, 67-76, The System Safety Society, Unionville, Virginia (1999).

[55] K. K. VEMURI, J. DUGAN and K. J. SULLIVAN, "Automatic Synthesis of Fault Trees for Computer-Based Systems", *IEEE Trans.Reliability*, **48**, 394-402 (1999).

[56] BALAKRISHMAN, M. and K. TRIVEDI, "Stochastic Petri Nets for Reliability Analysis of Communication Network Applications With Alternate Routing", *Reliab.Engng & System Safety*, **53**, 243-259 (1996).

[57] T. S. LIU and S. B. CHIOU, "The Application of Petri Nets to Failure Analysis", *Reliability Engineering and System Safety*, 129-142 (1997).

[58] M. GRIBAUDO, A. HORVAACUTE, A. BOBBIO, E. TRONCI, E. CIANCAMERLA and M. MINICHINO, *Fluid Petri Nets and Hybrid Model-Checking: A Comparative Case Study*, pp 239-57 (2006).

[59] N. E. FENTON, B. LITTLEWOOD, M. NEIL, L. STRIGINI, D. R. WRIGHT and P. J. COURTOIS, *Bayesian Belief Network Model for the Safety Assessment of Nuclear Computer-Based Systems*, pp 349-63, CRC Press, Inc., Boca Raton, FL (2000).

[60] B. A. GRAN and A. HELMINEN, "A Bayesian Belief Network for Reliability Assessment", U. VOGES (Ed.), *SAFECOMP 2001*, **LNCS 2187**, 33-45, Springer-Verlag, Heidelberg, Germany (2001).

[61] A. HELMINEN, *Reliability Estimation Of Safety-Critical Software-Based Systems Using Bayesian Networks*, Helsinki, Finland (2001).

[62] R. I. ZEQUIERA, "A Model for Bayesian Software Reliability Analysis", *Qual.Reliab.Engng.Int.*, **16**, 187-193 (2000).

[63] B. LI, M. LI and C. SMIDTS, "Integrating Software into PRA: A Test-Based Approach", C. SPITZER, U. SCHMOKER and V. N. DANG (Eds.), Springer – Verlag, London, U.K. (2004).

[64] N. F. SCHNEIDEWIND, "Analysis of Error Processes in Computer Software", *Proc.Int'l Conf.Reliable Software*, 76-78, IEEE CS Press (1975).

[65] N. F. SCHNEIDEWIND and T. W. KELLER, "Application of Reliability Models to the Space Shuttle", *IEEE Trans.Software Engineering*, **9**, 28-33 (1992).

[66] K. W. MILLER, L. J. MORELL, R. E. NOONAN, S. P. PARK, D. M. NICOL, B. W. MURRILL and J. M. VOAS, "Estimating the Probability of Failure When Testing Reveals No Failures", *IEEE Trans.Software Eng.*, **18**, 33-43 (1992).

[67] Y. YU and B. W. JOHNSON, "The Quantitative Safety Assessment for Safety-Critical Software", *Proc.29th Annual IEEE/NASA Software Engineering Workshop*, 193-198, IEEE, Piscataway, NJ (2005).

[68] A. L. GOEL and K. OKUMOTO, "Time-Dependent Error Detection Rate Model for Software and Other Performance Measures", *IEEE Trans.Reliability*, **28**, 206-211 (1979).

[69] K. MATSUMO, K. INOUE, T. KIKUMO and K. TORII, "Experimental Evaluation of Software Reliability Growth Models", *IEEE Proceedings of FTCS-18*, 148-153 (1988).

[70] S. YAMADA, H. OHTERA and H. NARIHISA, "Software Reliability Growth Models With Testing Effort", *IEEE Trans.Reliability*, **R-35**, 19-23 (1986).

[71] J. MAY, *Component-Based Software Reliability Analysis*, Department of Computer Science, University of Bristol, U.K. (2002).

[72] C. SMIDTS and M. LI, *Software Engineering Measures for Predicting Software Reliability in Safety Critical Digital Systems*, U.S. Nuclear Regulatory Commission, Office of Nuclear Regulatory Research, Washington, D.C. (2000).

[73] C. SMIDTS and M. LI, *Validation of A Methodology For Assessing Software Quality*, University of Maryland (2002).

[74] A. RAUZY, "Mode Automata and Their Compilation into Fault Trees", *Reliab.Engng & System Safety*, **78**, 1-12 (2002).

[75] M. A. BOYD and S. J. BAVUSO, "Simulation Modeling for Long Duration Spacecraft Control Systems", *Proc.Annual Reliability and Maintainability Symposium*, 106-113, IEEE, New York, NY (1993).

[76] K. S. TRIVEDI and V. G. KULKARNI, "FSPNs: Fluid Stochastic Petri Nets", M. A. MARSAN (Ed.), *Proc.14th Int.Conf.on Applications and Theory of Petri Nets*, 24-31, Springer-Verlag, Heidelberg (1993).

[77] G. JOHNSON and J. D. LAWRENCE, *Conceptual Software Reliability Prediction Models for Nuclear Power Plant Safety Systems*, Lawrence Livermore National Laboratory, Livermore, California (2000).

[78] Y. ZANG and M. M. GOLAY, "Development of a Method for Quantifying The Reliability of Nuclear Safety-Related Software", *PSAM6: Proceedings of the 6th International Conference on Probabilistic Safety Assessment and Management, CD-ROM Version*, Elsevier Science Ltd. (2002).

[79] G. PAI, S. DONOHUE and DUGAN.J., *Estimating Software Reliability From Process and Product Evidence*, Elsevier Science Ltd. (2002).

[80] B. LITTLEWOOD and D. WRIGHT, "Some Conservative Stopping Rules for the Operational Testing of Safety-Critical Software", *IEEE Trans.Software Engineering*, **23**, 673-683 (1997).

[81] C. SMIDTS, D. SOVA and G. K. MANDELA, "An Architectural Model for Software Reliability Quantification", *Proceedings of the Eighth International Symposium on Software Reliability Engineering (ISSRE '97)*, 324-336, IEEE Computer Society, Washington, D.C. (1997).

[82] S. YANG, N. SANG and G. XIONG, "Safety Testing of Safety Critical Software Based on Critical Mission Duration", 97-102, IEEE Computer Society, Washington, D.C. (2004).

[83] Y. CHEN and D. SINGPURWALLA, "Unification of Software Reliability Models by Self-Exciting Point Process", *Adv.Appl.Prob.*, **20**, 337-352 (1997).

[84] P. HAAPANEN, J. KORHONEN and U. PULKKINEN, *Licensing Process for Safety-Critical Software-Based Systems*, Radiation and Nuclear Safety Authority, Helsinki, Finland (2000).

[85] C. SMIDTS and M. LI, Preliminary Validation of a Methodology for Assessing Software Quality, NUREG/CR-6468, U.S. Nuclear Regulatory Commission, Washington, D.C. (2004)

[86] M. A. CARUSO, M. C. CHEOK, M. A. CUNNIGHAM and ET AL, *An Approach for Using Risk-Informed Decisions on Plant-Specific Changes to the Licensing Basis*, p 242 (1999).

[87] REGULATORY GUIDE 1.174, *An Approach for Using Probabilistic Risk Assessment in Risk Informed Decisions on Plant-Specific Changes to the Licensing Bases*, U.S. Nuclear Regulatory Commission, Washington, D.C. (1998)

[88] C. A. C. O. R. S. LETTER FROM MARIO V.BONACA and E. D. F. O. N. R. C. TO LUIS A.REYES, *Subject: Digital Instrumentation and Control Research Program*, U. S. Nuclear Regukatroy Commission, Washington, D.C. (2004).

[89] C. PERROW, *Normal Accidents, Living with High-Risk Technologies*, Princeton University Press, Princeton, New Jersey (1999).

[90] J. KIRSCHENBAUM, M. STOVSKY, P. BUCCI, T. ALDEMIR and S. A. ARNDT, "Benchmark Development for Comparing Digital Instrumentation and Control System Reliability Modeling Approaches", American Nuclear Society, LaGrange Park, IL (2005).

[91] T. J. MCCABE, *A Complexity Measure* (1976).

[92] *Code of Federal Regulations, Title 10 (Energy), Part 50 (Domestic Licensing of Production and Utilization Facilities)*.

[93] C. E. SHANNON and W. WEAVER, *The Mathematical Theory of Communication*, University of Illinois Press, Urbana, IL (1964).

[94] L. LAMPORT, S. SHOSTAK and P. PEASE, "The Byzantine Generals Problem", *ACM Transactions on Programming Languages and Systems*, **4**, 382-401 (1982).

[95] F. L. LIAN, J. R. MOYNE and D. M. TILBURY, "Performance Evaluation of Control Networks", *IEEE Control Systems Magazine*, **21**, 66-83 (2001).

[96] F. L. LIAN, J. R. MOYNE and D. M. TILBURY, "Network Design Consideration for Distributed Control Systems", *IEEE TRansactions on Control Systems Technology*, **10**, 297-307 (2002).

[97] *IEEE 802.5 Standard* (2001).

[98] *IEEE 802.3 Standard* (2005).

[99] Guidelines for Evaluating Electromagnetic and Radio-Frequency Interference in Safety-Related Instrumentation and Control Systems, RG 1.180, Revision 1, U.S. Nuclear Regulatory Commission, Washington, D.C. (2003)

[100] *IEEE 802.11 Standard* (2005).

[101] N. PLOPLYS, P. KAWKA and A. ALLEYNE, "Closed Loop Control Over Wireless Networks", *IEEE Control Systems Magazine*, 58-71 (2004).

[102] *REGULATORY GUIDE 1.152, Revision 2*, U.S. Nuclear Regulatory Commission, Washington, D.C. (2006).

[103] *Functional Safety of Electrical / Electronic / Programmable Electronic Safety-Related Systems*, International Electrotechnical Commission, Geneva, Switzerland (2000).

[104] *Design Assurance for Airborne Electronics Hardware*, Radio Technical Commission for Aeronautics (2000).

[105] B. W. JOHNSON, *Design and Analysis of Fault Tolerant Digital Systems*, Addison-Wesley (1989).

[106] J.-C. LAPRIE, *Dependable Concepts: Basic Concepts and Terminology*, Springer Verlag (1992).

[107] J. ARLAT, A. COSTES, Y. CROUZET, J.-C. LAPRIE and D. POWELL, "Fault Injection and Dependability Evaluation of Fault-Tolerant Systems", *IEEE Transactions on Computers*, **42**, 913-923 (1993).

[108] S. D. YOUNG and C. ELKS, "*Performance Assessment of Fault Tolerant Byzantine Computer*", *Computers in Aerospace*, American Institute of Aeronautics and Astronautics (1989).

[109] R. K. IYER and P. VELARDI, *Hardware-Related Software Errors: Measurement and Analysi*, pp 223-31 (1985).

[110] M. REYNOLDS and C. ELKS, *A Contemporary Perspective on Fault Injection Methods and Tools*, *CSCS-2006-002 rev 01*, University of Virginia, Charlottesville, Virginia (2006)

[111] R. K. IYER and D. J. ROSETTI, *A Measurement-Based Model for Workload Dependence of CPU Errors*, pp 511-9 (1986).

[112] C. CONSTANTINESCU, "Impact of Deep Submicron Technology on Dependability of VLSI Circuits", *Proceedings of the International Colnference on Dependable Systems and Networks*, 23-26 (2002).

[113] L. KAUFMAN, B. W. JOHNSON and J. DUGAN, *Coverage Estimation Using Statistics of the Extreme for When Testing Reveals No Failures* (2002).

[114] T. D. SMITH and B. W. JOHNSON, *A Variance Reduction Technique Via Fault Expansion for Fault Coverage Estimation*, pp 366-76 (1997).

[115] E. CUTRIGHT, T. DELONG, and B. W. JOHNSON, *Numerical Safety Evaluation Process for Safety-Critical Systems*, UVA-CSCS-NSE-001, University of Virginia, Charlottesville, Virginia (2003)

[116] E. CUTRIGHT and B. W. JOHNSON, Analytical Safety Model, UVA-CSCS-NSE-002, University of Virginia, Charlottesville, Virginia (2003)

[117] G. APOSTOLAKIS and J. S. WU, *The interpretation of probability, De Finetti.s representation theorem, and their implications to the use of expert opinions in safety assessment*, pp 311-22, R. E. BARLOW and C. A. CLAROOTI, Eds., Chapman&Hill, London, U. K. (1993).

[118] *PRISM: System Reliability Assessment Software*, Reliability Analysis Center (RAC), Rome, New York (2003).

[119] E. CUTRIGHT, T. DELONG, and B. W. JOHNSON, Generic Processor Fault Model, UVA-CSCS-NSE-004, University of virginia, Charlottesville, Virginia (2003)

[120] J. G. CHOI and P. H. SEONG, "*Dependability Estimation of Digital System by Operational Profile-Based Fault Injection* ", *PSA'99*, **I**, 499-506, American Nuclear society, LaGrange Park, Illinois (1999).

[121] D. T. SMITH, B. W. JOHNSON and J. A. PROFETA, *System Dependability Evaluation Via a Fault List Generation Algorithm*, pp 974-9 (1996).

[122] S. S. BHIDE. '*Dependability Modeling of Digital Embedded Systems Accounting for Common-Mode and Common-Cause Failures*', (University of Virginia2000).

[123] T. DELONG, T. D. SMITH and B. W. JOHNSON, *Dependability Metrics to Assess Safety Critical System* (2006).

[124] U.S.NUCLEAR REGULATORY COMMISSION, Severe Accident Risks: An Assessment for Five U.S. Nuclear Power Plants, NUREG-1150, U.S. Nuclear Regulatory Commission, Washington, D.C. (1990)

[125] N. E. TODREAS and M. S. KAZIMI, *Nuclear Systems: Thermal Hydraulic Fundamentals*, Hemisphere Pulishing. Corp. (1990).

[126] M. HASSAN and T. ALDEMIR, "A Data Base Oriented Dynamic Methodology for the Failure Analysis of Closed Loop Control Systems in Process Plants", *Reliability Engineering & System Safety*, **27**, 275-322 (1990).

[127] B. LEIMKUEHLER and S. V. SHERIKAR, "Getting Optimum Performance Through Feedwater Control Valve Modifications" (1997).

[128] M. YAU. 'Dynamic Flowgraph Methodology for the Analysis of Software Based Controlled Systems', (University of California, Los Angeles1997).

[129] M. YAU, G. APOSTOLAKIS and S. GUARRO, "The Use of Prime Implicants in Dependability Analysis of Software Controlled Systems", *Reliability Engineering and System Safety*, **62**, 23-32 (1998).

[130] M. YAU, M. WETHERHOLT and S. GUARRO, "Safety Analysis and Testing of Critical Space Systems Software", *Proceedings, 4th International Conference on Probabilistic Safety Assessment and Management (PSAM-4)*, Paper # September 13-18, New York, NY (1998).

[131] S. A. LAPP and G. J. POWERS, "Computer-Aided Synthesis of FaultTtrees", *IEEE Transactions on Reliability*, **R-26**, 2-13 (1977).

[132] S. L. SALEM, G. APOSTOLAKIS and D. OKRENT, *A New Methodology for the Computer-Aided Construction of Fault Trees*, pp 417-33 (1977).

[133] S. L. SALEM, J. S. WU and G. APOSTOLAKIS, *Decision Table Development and Application to the Construction of Fault Trees*, pp 51-64 (1979).

[134] E. J. HENLEY and H. KUMAMOTO, *Probability Risk Assessment: Reliability Engineering, Design, and Analysis*, IEEE Press (1992).

[135] S. GARRIBA, E. GUAGNINI and P. MUSSIO, *Multiple-Valued Logic Trees: Meaning and Prime Implicants*, pp 463-72 (1985).

[136] E. J. SHIELDS, G. APOSTOLAKIS and S. B. GUARRO, "Determining the Prime Implicants for Multi-State Embedded Systems", G. APOSTOLAKIS and J. S. WU (Eds.), *Proceedings of PSAM-II*, 7-12, International Asoociation for Probabilistic Assessment and Management, San Diego, California (1994).

[137] C. S. HSU, *Cell-to-cell Mapping: A Method of Global Analysis for Nonlinear Systems*, Springer-Verlag, New York, NY (1987).

[138] A. MOSLEH, K. N. FLEMING, G. W. PARRY, and ET AL, Procedures for Treating Common Cause Failures in Safety and Reliability Studies, NUREG/CR-4780 (EPRI NP-5613), U.S. Nuclear Regulatory Commission, Washington, D.C. (1989)

[139] M. BOUISSOU, "Boolean Logic Driven Markov Processes: A Powerful New Formalism for Specifying and Solving Very Large Markov Models", *PSAM6: Proceedings of the 6th International Conference on Probabilistic Safety Assessment and Management, CD-ROM Version*, Elsevier Science Ltd. (2002).

[140] P. BUCCI, J. KIRSCHENBAUM, T. ALDEMIR, C. L. SMITH and T. S. WOOD, "Constructing Dynamic Event Trees From Markov Models", M. STAMATALETOS and H. S. BLACKMAN (Eds.), *PSAM8: Proceedings of the 8th International Conference on Probabilistic Safety Assessment and Management, CD-ROM Version*, Paper # 369, ASME Press, Inc. (2006).

[141] S. RUSSELL and P. NORVIG, *Artificial Intelligence: A Modern Approach*, Prentice-Hall, New Jersey (2003).

[142] K. D. RUSSELL, M. B. SATTISON, and D. M. RASMUSON, Integrated Reliability and Risk Analysis System (IRRAS) Version 2.0 User's Guide, NUREG/CR-5111, U.S. Nuclear Regulatory Commission, Washington, D.C. (6-1-1990)

[143] G. A. APOSTOLAKIS, A Commentary on Model Uncertainty, NUREGICP-0138, U.S. Nuclear regulatory Commission, Washington, D.C. (1994)

[144] G. W. PARRY, "The Characterization of Uncertainty in Probabilistic Risk Assessments of Complex Systems", *Reliab.Engng & System Safety*, **54**, 119-126 (1996).

[145] T. NILSEN and T. AVEN, "Models and Model Uncertainty in the Context of Risk Analysis", *Reliab.Engng & System Safety*, **79**, 309-317 (2003).

[146] J. M. REINERT and G. A. APOSTOLAKIS, *Including Model Uncertainty in Risk-Informed Decision Making*, pp 354-69 (2006).

[147] G. APOSTOLAKIS, "The Concept of Probability in Safety Assessments of Technological Systems", *Science*, **250**, 1359-1364 (1990).

[148] T. ALDEMIR, "A Finer Mesh Is Not Always a Better Mesh - The Case of Non-Compact Support in Probabilistic Dynamics", *Mathematics and Computation, Supercomputing, Reactor*

Physics and Nuclear and Biological Applications, **CD-ROM**, American Nuclear Society, La Grange Park, IL (2005).

[149] I. MEZIC and T. RUNOLFSSON, "Uncertainty Analysis of Complex Dynamical Systems", *Proceedings of the 2004 American Control Conference*, **3**, 2659-2664 (2006).

[150] J. KNUDSEN and C. L. SMITH, "Estimation of System Failure Probability Uncertainty Including Model Success Criteria", *Proceedings of the 6th International Conference on Probabilistic Safety Assessment and Management (PSAM 6)*, **1** (2002).

[151] K. GO¡SEVA-POPSTOJANOVA and S. KAMAVARAM, "Assessing Uncertainty in Reliability of Component–Based Software Systems", *Proceedings of the 14th International Symposium on Software Reliability Engineering (ISSRE'03)*, 307-320 (2003).

[152] Y. LIANG, M. A. J. SMITH and K. TRIVEDI, "Uncertainty Analysis in Reliability Modeling", *2001 Proceedings of the Annual Reliability and Maintainability Symposium*, 229-234 (2001).

[153] E. CUTRIGHT, M. PESCOSOLIDO, T. DELONG, and B. W. JOHNSON, Statistical Model, Technical Report UVA-CSCS-NSE-003, Center for Safety Critical Systems, University of Virginia, Chalottesville, Virginia (2002)

[154] L. KAUFMAN. 'Dependability Analysis for Ultra-Dependable Systems Using Statistics of the Extremes', (University of Virginia, Charlottesville, Virginia2006).

[155] L. KAUFMAN, J. DUGAN and B. W. JOHNSON, *Using Statistics of the Extremes for Software Reliability Analysis*, pp 292-9 (1999).

[156] T. ALDEMIR, "Quantifying Setpoint Drift Effects in the Failure Analysis of Process Control Systems", *Reliability Engineering & System Safety*, **24**, 33-50 (1989).

Appendix A

Steam Generator Model

Notation

Symbol		Subscript	
A	Flow area	b	Bubble
g	32.2 ft/s^2	c	Condensate
h	Enthalpy	cs	Condensate on spray
J	778 ft-lbf/Btu	F	Fluid in the lower region
M	Mass	Ff	Fluid portion in the lower region
$\dot{m}$	Mass flow rate	Fg	Vapor portion in the lower region
P	Pressure	f	Saturated liquid
t	Time	fg	Saturated liquid to vapor
U	Internal energy	G	Fluid in the upper region
u	velocity	Gf	Liquid portion in the upper region (condensate)
V	Volume	Gg	Vapor portion in the upper region
v	Specific volume	g	Saturated vapor
a	Void fraction	i	Summation convention indicating boundary flows
r	Density	sp	Spray
t	Time constant	HTR	Heater
Q,q	Heat flow rate	FG	Interfacial transport
b	Bubble	Loss	Indicating heat loss to the environment
c	Condensate	in	Indicating flow into
cs	Condensate on spray	o	Flow out of

The steam generator is modeled as a two-region volume (see Fig.2.2.8). These two regions may have the following combinations:

- Upper region (G) superheated steam, lower region (F) subcooled liquid,
- Region G superheated steam, region F saturated liquid with bubbles forming,
- Region G saturated steam with droplets forming, region F subcooled liquid, and
- Region G saturated steam with droplets forming, region F saturated liquid with bubbles forming.

The steam generator model accounts for heat and mass transfer between the two regions. Mass transfer is modeled in the bubble rise and condensate drop models. The governing equations for the four possible combinations of states in the steam generator are as given in Sections A.1 through A.4.

A.1 Upper Region Superheated, Lower Region Subcooled

$$\frac{dM_G}{dt} = \sum_i \dot{m}_{Gi} \tag{A.1}$$

$$\frac{dM_F}{dt} = \sum_i \dot{m}_{Fi} \tag{A.2}$$

$$\frac{dh_G}{dt} = \frac{1}{M_G}\left[\sum_i (\dot{m}h)_{Gi} - h_G \sum_i \dot{m}_{Gi} + \frac{144}{J} V_G \frac{dP}{dt}\right] \tag{A.3}$$

$$\frac{dh_F}{dt} = \frac{1}{M_F}\left[\sum_i (\dot{m}h)_{Fi} + Q_{in} - h_F \sum_i \dot{m}_{Fi} + \frac{144}{J} V_F \frac{dP}{dt}\right] \tag{A.4}$$

$$\frac{dP}{dt} = -\left\{ v_F \sum_i \dot{m}_{Fi} + v_G \sum_i \dot{m}_{Gi} + \frac{\partial v_F}{\partial h_F}\left[\sum_i (\dot{m}h)_{Fi} + Q_{in} - h_F \sum_i \dot{m}_{Fi}\right] + \frac{\partial v_G}{\partial h_G}\left[\sum_i (\dot{m}h)_{Gi} - h_G \sum_i \dot{m}_{Gi}\right]\right\} \Big/ \left\{ M_F \frac{\partial v_F}{\partial P} + \frac{144}{J} V_F \frac{\partial v_F}{\partial h_F} + M_G \frac{\partial v_G}{\partial P} + \frac{144}{J} V_G \frac{\partial v_G}{\partial h_G} \right\} \tag{A.5}$$

$$M_F v_F = V_F \tag{A.6}$$

$$M_G v_G = V_G \tag{A.7}$$

$$V = V_F + V_G \tag{A.7a}$$

A.2 Upper Region Superheated, Lower Region Saturated

$$\frac{dM_G}{dt} = \sum_i \dot{m}_{Gi} \tag{A.8}$$

$$\frac{dM_F}{dt} = \sum_i \dot{m}_{Fi} \tag{A.9}$$

$$\frac{dh_G}{dt} = \frac{1}{M_G}\left[\sum_i (\dot{m}h)_{Gi} - h_G \sum_i \dot{m}_{Gi} + \frac{144}{J} V_G \frac{dP}{dt}\right] \tag{A.10}$$

$$\frac{dM_{Fg}}{dt} = \sum_i \dot{m}_{Fi} - \frac{dM_{Ff}}{dt} \tag{A.11}$$

$$\frac{dM_{Ff}}{dt} = -\frac{1}{h_{fg}}\left[\sum_i (\dot{m}h)_{Fi} + Q_{in} - h_g \sum_i \dot{m}_{Fi} - M_{FH}\frac{dP}{dt}\right] \tag{A.12}$$

$$M_{FH} = M_{Ff}\frac{dh_F}{dP} + M_{Fg}\frac{dh_g}{dP} - \frac{144}{J}V_F \tag{A.13}$$

$$\begin{aligned}\frac{dP}{dt} = -\Bigg\{ & v_{fg}\left[\sum_i (\dot{m}h)_{Fi} + Q_{in} - h_g \sum_i \dot{m}_{Fi}\right] + \\ & h_{fg}\left\{v_g \sum_i \dot{m}_{Fi} + v_G \sum_i \dot{m}_{Gi} + \left[\sum_i (\dot{m}h)_{Gi} - h_G \sum_i \dot{m}_{Gi}\right]\frac{\partial v_G}{\partial h_G}\right\}\Bigg\} \\ & \Big/\left\{h_{fg}\left[M_{Fv} + M_G\frac{\partial v_G}{\partial P} + \frac{144}{J}V_G\frac{\partial v_G}{\partial h_G}\right] - v_{fg}M_{FH}\right\}\end{aligned} \tag{A.14}$$

$$M_{Fv} = M_{Ff}\frac{dv_f}{dP} + M_{fg}\frac{dv_g}{dP} \tag{A.15}$$

$$M_G v_G = V_G \tag{A.16}$$

$$M_{Fg} v_g + M_{Ff} v_f = V_F \tag{A.17}$$

$$V = V_F + V_G \tag{A.18}$$

A.3 Upper Region Saturated, Lower Region Subcooled

$$\frac{dM_G}{dt} = \sum_i \dot{m}_{Gi} \tag{A.19}$$

$$\frac{dM_F}{dt} = \sum_i \dot{m}_{Fi} \tag{A.20}$$

$$\frac{dh_F}{dt} = \frac{1}{M_F}\left[\sum_i (\dot{m}h)_{Fi} + Q_{in} - h_F \sum_i \dot{m}_{Fi} + \frac{144}{J} V_F \frac{dP}{dt}\right] \tag{A.21}$$

$$\frac{dM_{Gg}}{dt} = \sum_i \dot{m}_{Gi} - \frac{dM_{Gf}}{dt} \tag{A.22}$$

$$\frac{dM_{Gf}}{dt} = -\frac{1}{h_{fg}}\left[\sum_i (\dot{m}h)_{Gi} - h_g \sum_i \dot{m}_{Gi} - M_{GH}\frac{dP}{dt}\right] \tag{A.23}$$

$$M_{GH} = M_{Gf}\frac{dh_f}{dP} + M_{Gg}\frac{dh_g}{dP} - \frac{144}{J}V_G \tag{A.24}$$

$$\begin{aligned}\frac{dP}{dt} = -\Bigg\{ & v_{fg}\left[\sum_i (\dot{m}h)_{Gi}\right] - h_g \sum_i \dot{m}_{Gi} + \\ & h_{fg}\left\{ v_g \sum_i \dot{m}_{Gi} + v_F \sum_i \dot{m}_{Fi} + \left[\sum_i (\dot{m}h)_{Fi} + Q_{in} - h_F \sum_i \dot{m}_{Fi}\right]\frac{\partial v_F}{\partial h_F}\right\}\Bigg\} \\ & \Big/\left\{ h_{fg}\left[M_{Gv} + M_F \frac{\partial v_F}{\partial P} + \frac{144}{J} V_F \frac{\partial V_F}{\partial h_F}\right] - v_{fg} M_{GH}\right\}\end{aligned} \tag{A.25}$$

$$M_{Gv} = M_{Gf}\frac{\partial v_f}{\partial P} + M_{Gg}\frac{\partial v_g}{\partial P} \tag{A.26}$$

$$M_{Gg} v_g + M_{Gf} v_f = V_G \tag{A.27}$$

$$M_F v_F = V_F \tag{A.28}$$

$$V = V_F + V_G \tag{A.28a}$$

A.4 Upper Region Saturated, Lower Region Saturated

$$\frac{dM_G}{dt} = \sum_i \dot{m}_{Gi} \tag{A.29}$$

$$\frac{dM_F}{dt} = \sum_i \dot{m}_{Fi} \tag{A.30}$$

$$\frac{dM_{Gf}}{dt} + \frac{dM_{Gg}}{dt} = \sum_i \dot{m}_{Gi} \tag{A.31}$$

$$\frac{dM_{Ff}}{dt} + \frac{dM_{Fg}}{dt} = \sum_i \dot{m}_{Fi} \tag{A.32}$$

$$\frac{dM_{Gf}}{dt} = -\frac{1}{h_{fg}}\left[\sum_i (\dot{m}h)_{Gi} - h_g \sum_i \dot{m}_{Gi} - M_{GH}\frac{dP}{dt}\right] \tag{A.33}$$

$$\frac{dM_{Ff}}{dt} = -\frac{1}{h_{fg}}\left[\sum_i (\dot{m}h)_{Fi} - h_f \sum_i \dot{m}_{Fi} + Q_{in} - M_{FH}\frac{dP}{dt}\right] \tag{A.34}$$

$$M_{FH} = M_{Ff}\frac{dh_f}{dP} + M_{Fg}\frac{dh_g}{dP} - \frac{144}{J}V_F \tag{A.35}$$

$$\frac{dP}{dt} = -\left\{v_{fg}\left[\sum_i (\dot{m}h)_{Gi} + \sum_i (\dot{m}h)_{Fi} - \left(\sum_i \dot{m}_{Gi} + \sum_i \dot{m}_{Fi}\right)h_g + Q_{in}\right]\right.$$
$$\left. + h_{fg}\left[v_g\left(\sum_i \dot{m}_{Gi} + \sum_i \dot{m}_{Fi}\right)\right]\right\}$$
$$/\left\{h_{fg}\left(M_{Gv} + M_{Fv}\right) - v_{fg}\left(M_{GH} + M_{FH}\right)\right\} \tag{A.36}$$

$$M_{GH} = M_{Gf}\frac{dh_f}{dP} + M_{Gg}\frac{dh_g}{dP} - \frac{144}{J}V \tag{A.37}$$

$$M_{Gv} = M_{Gf}\frac{dv_f}{dP} + M_{Gg}\frac{dv_g}{dP} \tag{A.38}$$

$$M_{Fv} = M_{Ff}\frac{dv_f}{dP} + M_{Fg}\frac{dv_g}{dP} \tag{A.39}$$

$$M_{Gg}v_g + M_{Gf}v_f = V_G \tag{A.40}$$

$$M_{Fg} V_g + M_{Ff} V_f = V_F \tag{A.41}$$

$$V = V_F + V_G \tag{A.41a}$$

Appendix B

The Failure Modes and Effects Analysis (FMEA) of the benchmark DFWCS

The Failure Modes and Effects Analysis (FMEA) of the benchmark digital feedwater control system is presented in tabular form below.

Table B.1: FMEA Chart

Component	***Failure Mode***	***Detection of Failure***	***Effect of Failure***
Main Computer	Loss of all analog Inputs	Main Computer detects loss of inputs and fails itself	Fail over to Backup Computer, no change in water level
Backup Computer	Loss of all analog Inputs	Backup Computer detects loss of inputs and fails itself	Fail over to MFV/BFV/FP controllers, water level will drift based on physical process
Main Computer	Loss of Power	MFV/BFV/FP controllers detect failure and initiate fail over to Backup	Fail over to Backup Computer, no change in water level
Backup Computer	Loss of Power	MFV/BFV/FP controllers detect failure and initiate fail over to MFV/BFV/FP controllers	Fail over to MFV/BFV/FP controllers, water level will drift based on physical process
Main Computer	Output sent to MFV/BFV/FP controller is differnt from the output sent to the MFV/BFV/FP.	Main Computer fails itself.	Fail over to Backup Computer, no change in water level
Backup Computer	Output sent to MFV/BFV/FP controller is differnt from the output sent to the MFV/BFV/FP.	Backup Computer fails itself.	Fail over to MFV/BFV/FP controllers, water level will drift based on physical process
Main Computer	Excessive change in rate of outputs	MFV/BFV/FP controllers detect failure and initiate fail over to Backup	Fail over to Backup Computer, no change in water level
Backup Computer	Excessive change in rate of outputs	MFV/BFV/FP controllers detect failure and initiate fail over to MFV/FP/BFV controllers	Fail over to MFV/BFV/FP controllers, water level will drift based on physical process
Main/Backup Computer	Arbitrary Output	No detection.	Unknown.

Main/Backup Computer	Loss of one sensor.	Computer ignores lost input and uses the old signal.	Fail over to MFV/BFV/FP controllers, water level will drift based on physical process
Main/Backup Computer	Loss of both sensors.	Computer ignores lost input and uses the old signals.	Fail over to MFV/BFV/FP controllers, water level will drift based on physical process
Main Computer	Loss of both redundant sensors through out range errors or excessive rate	Computer uses old values and indicates that both sensors have failed	Fail over to Backup Computer
Backup Computer	Loss of both redundant sensors through out range errors or excessive rate	Computer uses old values and indicates that both sensors have failed	Failover to MFV/BFV/FP controller, water level will drift based on physical process
MFV Controller	Loss of input from Main Computer	MFV detects loss of input and starts fail over to Backup	None
MFV Controller	Loss of input from Backup Computer	MFV detects loss of input and starts failover procedure if needed to MFV/BFV/FP controller control	No change unless Main already failed, if so the water level with drift as a result of physical process changes
MFV Controller	Loss of analog output	PDI controller detects loss of output and uses the old value of the MFV controller to the MFV and transfer control to the PDI/BFV/FP controllers	Water level will drift as a result of physical process changes
MFV Controller	Loss of setpoint output	Computers detect the out of range setpoint signal and revert to a predetermined setpoint	Water level will change to follow the predetermined setpoint
MFV Controller	Arbitrary Output to MFV	No detection	Valve will follow output, causing water level to respond accordingly
MFV Controller	Late delivery of output to MFV	No detection	Unknown
MFV Controller	Loss of Main Computer failed signal (failed off)	MFV controller will initiate fail over to Backup Computer	None
MFV Controller	Loss of Backup Computer failed signal (failed off)	None	Backup Computer is unavailable for fail over.
MFV Controller	Loss of Main Computer failed signal (failed on)	MFV controller will be unable to tell if the Main has failed unless the Main fails its watchdog timer check	Water level output could be controlled by a failed Main Computer, causing it to change in unknown ways

MFV Controller	Loss of Backup Computer failed signal (failed off)	MFV controller will be unable to tell if the Backup has failed unless the Backup fails its watchdog timer checks	Water level output could be controlled by a failed Backup Computer, causing it to change in unknown ways
MFV Controller	Loss of watchdog timer status for Main Computer (failed off)	MFV controller will initiate fail over to Backup Computer	None
MFV Controller	Loss of Main Computer watchdog timer signal (failed on)	MFV controller will be unable to use the watchdog timer to fail the Computer.	Failed Main Computer could control the MFV, causing unknown changes in the water level
MFV Controller	Loss of watchdog timer status for Backup Computer (failed off)	MFV controller will transfer control to its own internal buffer of old output values	Water level will drift due to changes in the physical process
MFV Controller	Loss of Backup Computer watchdog timer signal (failed on)	MFV controller will be unable to use the watchdog timer to fail the Computer.	Failed Backup Computer could control the MFV, causing unknown changes in the water level
MFV Controller	Main Computer failed (through watchdog timer or failed signal)	MFV controller initiates transfer of control to the Backup Computer	None
MFV Controller	Backup Computer failed (through watchdog timer or failed signal)	MFV controller initiates transfer of control to its own internal buffer of an old value of the MRV output	Water level will drift due to changes in the physical process.
MFV Controller	Arbitrary Output	No detection unless output drops to 0.	Unknown if undetected. PDI will hold the valve and cause a transfer out of automatic control if detected.
BFV Controller	Loss of input from Main Computer	BFV controller detects loss of input and starts the fail over procedure transferring control to the Backup Computer	None
BFV Controller	Loss of input from Backup Computer	BFV controller detects loss of input and starts fail over procedure if needed to MFV/BFV/FP control	No change unless Main already failed, if so the water level with drift as a result of physical process changes
BFV Controller	Loss of analog output	None	Water level will increase as the BFV will close
BFV Controller	Arbitrary output to BFV	No detection	Valve will follow output, causing water level to respond accordingly
BFV Controller	Late delivery of output to BRV	No detection	Unknown

BFV Controller	Loss of Main Computer failed signal (failed off)	BFV controller will initiate failover to Backup Computer	None
BFV Controller	Loss of Backup Computer failed signal (failed off)	None	Backup Computer is unavailable for fail over.
BFV Controller	Loss of Main Computer failed signal (failed on)	BFV controller will be unable to tell if the Main has failed unless the Main fails its watchdog timer check	Water level output could be controlled by a failed Main Computer, causing it to change in unknown ways
BFV Controller	Loss of Backup Computer failed signal (failed off)	BFV controller will be unable to tell if the Backup has failed unless the Backup fails its watchdog timer checks	Water level output could be controlled by a failed Backup Computer, causing it to change in unknown ways
BFV Controller	Loss of watchdog timer status for Main Computer (failed off)	BFV controller will initiate fail over to Backup Computer	None
BFV Controller	Loss of Main Computer watchdog timer signal (failed on)	BFV controller will be unable to use the watchdog timer to fail the Computer.	Failed Main Computer could control the BFV, causing unknown changes in the water level
BFV Controller	Loss of watchdog timer status for Backup Computer (failed off)	BFV controller will transfer control to its own internal buffer of old output values	Water level will drift due to changes in the physical process
BFV Controller	Loss of Backup Computer watchdog timer signal (failed on)	BFV controller will be unable to use the watchdog timer to fail the Computer.	Failed Backup Computer could control the BFV, causing unknown changes in the water level
BFV Controller	Main Computer failed (through watchdog timer or failed signal)	BFV controller initiates transfer of control to the Backup Computer	None
BFV Controller	Backup Computer failed (through watchdog timer or failed signal)	BFV controller initiates transfer of control to its own internal buffer of an old value of the BRV output	Water level will drift due to changes in the physical process.
BFV Controller	Arbitrary Output	No detection.	Unknown.
FP Controller	Loss of input from Main Computer	FP controller detects this and starts the fail over procedure transferring control to the Backup Computer	None

FP Controller	Loss of input from Backup Computer	FP controller detects this and starts fail over procedure if needed to MFV/BFV/FP controller control	No change unless Main already failed, if so the water level with drift as a result of physical process changes
FP Controller	Loss of analog output	No detection	Water level will change in unknown ways
FP Controller	Arbitrary Output to FP	No detection	Pump will follow output, causing water level to respond accordingly
FP Controller	Late delivery of output to FP	No detection	Unknown
FP Controller	Loss of Main Computer failed signal (failed off)	FP controller will initiate fail over to Backup Computer	None
FP Controller	Loss of Backup Computer failed signal (failed off)	None	Backup Computer is unavailable for fail over.
FP Controller	Loss of Main Computer failed signal (failed on)	FP controller will be unable to tell if the Main has failed unless the Main fails its watchdog timer check	Water level output could be controlled by a failed Main Computer, causing it to change in unknown ways
FP Controller	Loss of Backup Computer failed signal (failed off)	FP controller will be unable to tell if the Backup has failed unless the Backup fails its watchdog timer checks	Water level output could be controlled by a failed Backup Computer, causing it to change in unknown ways
FP Controller	Loss of watchdog timer status for Main Computer (failed off)	FP controller will initiate fail over to Backup Computer	None
FP Controller	Loss of Main Computer watchdog timer signal (failed on)	FP controller will be unable to use the watchdog timer to fail the Computer.	Failed Main Computer could control the feedwater pump, causing unknown changes in the water level
FP Controller	Loss of watchdog timer status for Backup Computer (failed off)	FP controller will transfer control to its own internal buffer of old output values	Water level will drift due to changes in the physical process
FP Controller	Loss of Backup Computer watchdog timer signal (failed on)	FP controller will be unable to use the watchdog timer to fail the Computer.	Failed Backup Computer could control the feedwater pump, causing unknown changes in the water level
FP Controller	Arbitrary Output	No detection.	Unknown.
FP Controller	Main Computer failed (through watchdog timer or failed signal)	FP controller initiates transfer of control to the Backup Computer	None

FP Controller	Backup Computer failed (through watchdog timer or failed signal)	FP controller initiates transfer of control to its own internal buffer of an old value of the FP output	Water level will drift due to changes in the physical process.
PDI Controller	Loss of analog inputs	PDI controller will output an old MFV controller signal to the MFV output	MFV will open more, causing an increase in water level
PDI Controller	Loss of power	No detection.	None, unless the MFV fails. The resulting failure would be unknown.
PDI Controller	Arbitrary Failure	No detection.	Unknown.
PDI Controller	Loss of analog outputs	PDI will be unable to intervene if the MFV controller fails	If the MFV controller fails then the water level will change in unknown ways