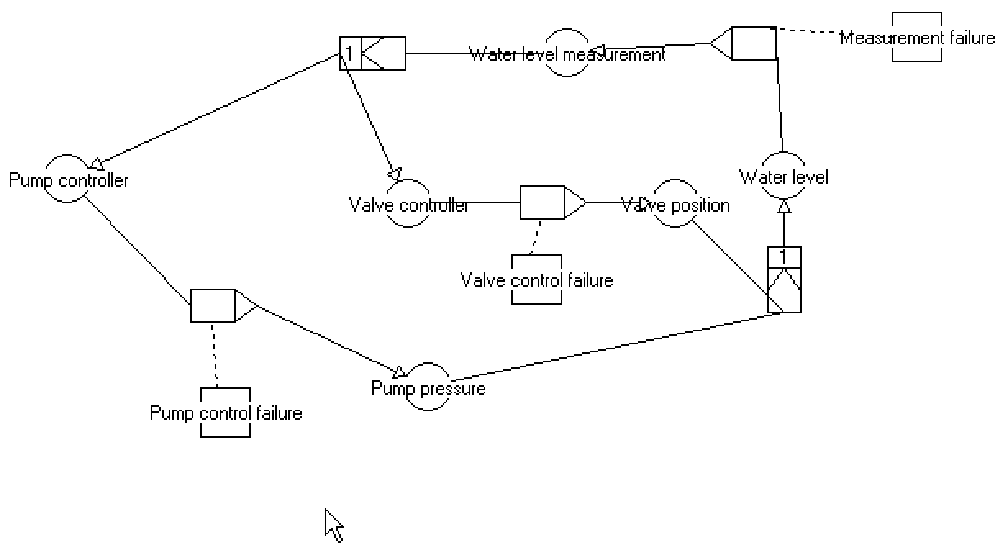
RESEARCH REPORT

NO VTT-R-01749-08 | 1.9.2009



CHARISMA/SAFIR 2010

Reliability of digital control systems in nuclear power plants — Modelling the feedwater system

Authors: Ilkka Karanta, Matti Maskuniitty

Confidentiality: Public

Report's title Reliability of digital control systems in nuclear power plants — Modelling the feedwater system	
Customer, contact person, address VYR	Order reference ad 27/2007/SAF, 12.3.2008
Project name CHAllenges in Risk-Informed Safety MAnagement	Project number/Short name 23615 Charisma
Author(s) Ilkka Karanta, Matti Maskuniitty	Pages 30/
Keywords risk analysis, digital control systems, distributed systems	Report identification code VTT-R-01749-08

Summary

Assessing the reliability of digital control systems is an important but challenging task. Traditional tools of reliability and risk analysis, such as event and fault trees, have limitations. Some dynamic methods, such as dynamic flowgraph methodology and Markov models, seem to hold promise but evidence of their suitability and utility is still scarce.

This report deals with the reliability assessment of distributed digital control systems, using the feedwater system of a BWR nuclear power plant as an example. Problems and methods of reliability analysis of distributed digital control systems are briefly surveyed.

From various methods proposed for the reliability analysis of digital systems, the dynamic flowgraph methodology (DFM) was chosen for further consideration because the physical system can be incorporated in it in a natural way. DFM is based on state and time discretization, and a DFM model is a directed graph where variables are connected to each other by edges. At each time instance, each variable can be in one of several states. The state of each variable on the next time instance depends on the states of its input variables at the present time instance; this dependence is expressed as a decision table. The analysis is based on an exhaustive enumeration of all possible state combinations; for the analysis, the values of any variables on any of the time instances covered by the analysis can be specified. The result of a DFM analysis is a set of prime implicants (multistate analoque of minimal cut sets) that can then be further analyzed to yield probability estimates for failures.

A simple model of the feedwater system was implemented in DFM. It seems that nontrivial failure modes of the feedwater control system can be found with DFM; a combinatorial explosion seems to take place in models with a large number of variables, but this can be controlled somewhat by setting proper initial and boundary conditions for the model variables.

Confidentiality	Public

Espoo May 2009

Signatures	Signatures	Signatures
Written by Ilkka Karanta senior research scientist	Reviewed by Jan-Erik Holmberg, senior research scientist	Accepted by Jari Hämäläinen, technology manager

VTT's contact address

Distribution (customer and VTT)
SAFIR2010 TR8

Preface

CHARISMA (Challenges in Risk-Informed Safety Management) is a part of the SAFIR2010 research programme funded by Valtion Ydinvoimarahasto (VYR). Its objectives are related to the use of probabilistic safety assessment (PSA) to support decision making and to intrinsic as well as practical problems in PSA techniques.

This report is a result of CHARISMA’s subtask “Reliability of automation”. The subtask aims at bringing the analysis of digital systems reliability to the systems architecture level. Different approaches for this purpose are surveyed, and one approach is selected for demonstration purposes. The application case is the feedwater system of a BWR nuclear power plant.

The authors would like to acknowledge the cooperation of Olli Paasikivi of TVO from whom valuable information about the feedwater system at Olkiluoto 1 & 2 nuclear power plants was obtained.

Espoo and Tampere, May 2009

Authors

Contents

1 Introduction

Risk and reliability analysis of digital instrumentation and control (I&C) systems is considered difficult. On the one hand, reliability analysis of software is difficult because the errors are systematic and not random. Even though a program has been extensively tested, it might still exhibit faulty behaviour under some rare circumstances, and the probability of this event should be estimated. In addition, the input space of a digital control system – all possible measurement histories and parameter configurations - is often extremely large.

Digital control systems can be analyzed on several abstraction levels. At the code (or circuit) level, the analysis is precise but the analysis effort might be prohibitively large. At the architecture level, the required effort might be considerably smaller but the analysis is in danger of missing some important failure mechanisms.

In current probabilistic safety analyses (PSA), distributed control systems are analysed and modelled very simply. In many cases, the starting point for modelling is a reliability analysis made by the vendor. Incorporating the vendor's analysis in PSA is not a straightforward task. Other questions related to distributed control systems are how to link the model to reliability models of other I&C systems and user interfaces (control room) and how the architecture of I&C systems affects, e.g., the possibility of common cause failures.

2 Goal

The purpose of the present report is twofold. First, the problems of digital system reliability assessment, and the methods for solving them are surveyed. Second, a method is chosen for the assessment of a case system, namely the feedwater control system of the Olkiluoto 1/2 nuclear power plant.

3 Some existing digital system reliability models

Quite many models are applicable or have been applied in modelling NPP control systems for reliability assessment purposes. This section gives a brief overview of some of them.

Some methods were rejected because they are applicable only to the software, leaving out the physical system to be controlled. Such models are e.g. software reliability models (see [Kar06]), software metric-based methods [Smi04] and test-based methods [Smi05].

Some other methods that are potentially applicable in the problem include Petri nets [God96] and event-sequence diagrams [Swa99].

There are also other deterministic methods for showing reliability such as methods based on testing and methods based on verification and validation by inspection, but these won't be considered here.

3.1 Formal methods

Hardware design and software can be proven to be defect-free by using formal methods [Kor04]. These are mathematically-based methods where

- a formal description is produced in a specification language,
- the desirable properties of the system are stated in the same language, and
- a formal proof is constructed that the system satisfies these properties.

Formal methods cover many methods with different background theories, e.g. model-based and algebraic specifications, abstract state machines, CSP and CCS, temporal logics, rewriting techniques, finite automata and model checking.

Constructing a proof is a verification of the system's reliability for the properties included in the model. In practise such proofs are so tedious for all except the smallest systems that they are not conducted except for the most safety-critical systems or parts. Lightweight formal methods [Eas97] which emphasize partial specification and focused application, may be used when a full formal analysis would be infeasible.

Formal proofs can be conducted manually or using a software tool. The main classes of software tools that automate formal analysis are model checkers and theorem provers. In automated theorem proving, a system attempts to produce a formal proof of the desired property/properties from scratch, given a model of the system, a set of logical axioms and a set of inference rules.

There are several ways in which formal methods may aid the reliability assessment process (these are adopted from [Hei05]):

- demonstrate the well-formedness of reliability specifications. When reliability requirements are converted to a formal notation, it can be checked that the specification is complete (no required behaviour is missing) and consistent (no behaviour in the specification is ambiguous)
- discover reliability requirement violations in the system. For example, it may be analyzed under which circumstances a system of valves and pumps produces a too high or too low water level in a container. Such violations will be corrected before implementation of the system. However, the evidence can be used an indicator of the quality of the system development, c.f. use of statistics of errors found in testing phase.
- verify critical properties. A theorem prover or a model checker may be used to verify that a software artefact, such as a requirements specification or a design specification, satisfies a critical property. Then, failure modes that depend on the absence of this critical property can be left out of reliability assessment.

In addition, reliability models for the system may be constructed from the formal specifications. Here, a formal language or notation acts as a modelling tool.

Model checking [Cla99] means formulating a logical model of the system, formulating a specification of the desired properties of the system, and verifying that the system satisfies the properties by examining through all the ways that the system can execute (or a subset that approximates all executions closely enough). The simulation is usually done using a software checker program. The program outputs a yes if the system (or, better said, the model) satisfies the specifications, and a counterexample if it does not. In the method used by VTT, a complete verification of requirements is performed based on a finite state machine model of the system [Val08]

A model checker model is same as a system reliability model with the regard to the presentation of the functional dependencies. Practically a reliability model and a model checker model can be developed with approximately same effort or jointly, e.g., based on common system FMEA.

Another popular class of formal methods is based on program semantics [Rii07]. However, these have not so far achieved the popularity of model checking.

3.2 Dynamic flowgraph methodology

The dynamic flowgraph methodology (DFM) is an approach to modeling and analyzing the behaviour of dynamic systems for reliability/safety assessment and verification [Gar95]. DFM models express the logic of the system in terms of causal relationships between physical variables and states of the control systems; the time aspects of the system (execution of control commands, dynamics of the process) are represented as a series of discrete state transitions. DFM can be used for identifying how certain postulated events may occur in a system; the result is a set of timed fault trees, whose prime implicants (multi-state analogue of minimal cut sets) can be used to identify system faults resulting from unanticipated combinations of software logic errors, hardware failures and adverse environmental conditions.

DFM has been used to assess the reliability of nuclear power plant control systems [Ald07], but also of space rockets [Yau95] and chemical batch processes [Hou00].

DFM models are directed graphs. They consist of variable and condition nodes; causality and condition edges; and transfer and transition boxes and their associated decision tables.

- Process variable nodes represent physical variables and states of the control systems that are needed to capture the essential functional behaviour of the system. If the variable is originally continuous (e.g. flow rate), it is discretized into a finite number of states.
- Causality edges connect two process variable nodes to indicate the existence of a cause-and-effect relationship between the two variables. The precise nature of the functional relationship (the transfer function) is described by a transfer box associated with the edge.
- The transfer box is always directly associated with each causality edge. It has one or more input edges, each coming from a node, and one output edge, pointing to a node. It contains a decision table. This is a generalized truth table, where for each combination of the states of input variables, the state of the output is given. The decision table tells what the state of the output variable is for a given combination of states of the inputs. Informally put, a transfer box connects nodes to indicate cause-and-effect relationships.
- Condition nodes, like process variable nodes, represent physical or software parameters. They are used to “to more explicitly identify

component failure states, changes of process operation regimes and modes, and software switching actions" [Ald07]. By nature, their states are discrete and finite in number. In principle, condition nodes would not be needed as process variable nodes represent also discrete behaviour, but they are used to more explicitly identify component failure states, changes in process operation regimes and modes, and control system commanding actions.

- Condition edges are used to represent discrete behaviour of the system. They link condition boxes to transfer boxes, indicating the possibility of using a different transfer function to map the input variable to output variable states. In principle, these would not be needed as the state of any state and state transfer in DFM are discrete, but are used to emphasize control logic.
- Transition boxes are similar to transfer boxes in all respects except that a time lag or time transition is assumed to occur between the time when the input variable states are true and the time when the output variable state associated with those inputs is reached. The length of this time delay is an attribute of the transition box. To put it in another way, a transfer box is a transition box with delay of length 0.

The graphical representations of the DFM elements are depicted in figure **1**.

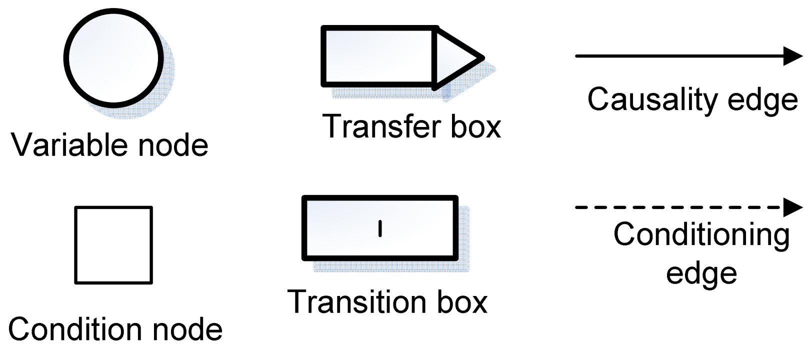


Figure 1. The modeling elements of the dynamic flowgraph methodology

After construction, the DFM model can be analyzed in two different modes, *deductive* and *inductive* [Hou02]. In inductive analysis, event sequences are traced from causes to effects; this corresponds to simulation of the model. In deductive analysis, event sequences are traced backward from effects to causes.

A deductive analysis starts with the identification of a particular system condition of interest (a top event); usually this condition corresponds to a failure. To find the root causes of the top event, the model is backtracked through the network of nodes, edges, transfer and transition boxes. This means that the model is worked backward in the cause-and-effect flow to find what states of variables (and at what time instances) are needed to produce the top event. The result of a deductive analysis is a set of prime implicants.

A prime implicant consists of a set of triplets (*V*, *S*, *T*); each triplet tells that variable *V* is in a state *S* at time *T*. The circumstances described by the set of triplets causes the top event. Prime implicants are similar to minimal cutsets of fault tree analysis, except that prime implicants are timed and prime implicants deal with multivalued variables (fault trees deal with Boolean variables). A useful analogy is that deductive analysis corresponds to minimal cut set search of a fault tree.

Once primary implicants have been found, the top event probability is quantified as in the MCS analysis of a fault tree.

In inductive analysis, all the possible consequences of a given system initial condition or boundary condition are generated. These initial or boundary conditions can be defined to represent desired and undesired states. Starting from a combination of desired states, an inductive analysis can be used to verify system requirements (e.g. that normal operation under normal conditions doesn't lead to undesired states). Starting from a combination of undesired states, inductive analysis can be used to verify the system's safety behaviour. A useful analogy is that inductive analysis corresponds to constructing an event tree.

The application scope of DFM is large. The most important current areas of application include

- determination of prime implicants or minimal cutsets for PRA purposes. These can be used to construct timed fault trees.
- inspection of a given design for requirements compliance
- setting up a testing plan by examining what events might lead to failure
- computation of probabilities of top events for PRA/PSA.

The main benefits of the approach are as follows:

- both quantitative and qualitative factors can be taken into account
- both measurable and nonmeasurable variables can be incorporated
- equipment, software, and their interaction with the environment can all be modelled within the same formalism
- temporal aspects are taken into account. The notion of discrete time delays is sufficient for most applications
- the formalism is very simple and easy to learn. The concept of an acyclic graph is immediately graspable. There is essentially only one kind of node, that with a discrete state; and one kind of interaction, that with a decision table (and possibly with a time delay)
- modelling is simpler than in e.g. dynamic system models because the laws guiding the behaviour of the system need to be known only approximately

The main drawbacks and limitations of DFM are

- a more realistic modelling easily causes a combinatorial explosion as the number of states in the decision tables grows. This can be controlled somewhat by giving more constraints in the top event
- thinking in terms of multi-valued logic might be alien to many industry representatives. However, the logics that can be implemented in DFM are simple propositional ones, and decision tables are easy to understand
- currently there exists no systematic method of constructing DFM models. However, the methods of qualitative reasoning [Kui94] may help in this respect, because they are concerned also with the qualitative description of systems with essentially a quantitative nature (e.g. physical systems)

- It would be also of interest to know how much of the system's qualitative behaviour is lost when the model is discretized in state space and in time. From a practical point of view, knowing how many time steps to include in the analysis to cover all interesting fault modes is also of interest.
- so far only one software implementation exists, and its source code or even computation algorithms are not available for inspection.

The main use of DFM in PSA could be to support the construction of fault trees for dynamic systems and systems with loop dependencies (e.g. feedback control systems, back-upped electric systems).

To the present authors' knowledge, the computational complexity of DFM has not been analyzed thus far. However, it is easy to see that in the worst case - several variable combinations leading to each variable state for many variables – the number of possible states that lead to the next state grows exponentially.

3.3 Probabilistic methods

A common theme uniting the research described in this section is dynamic system analysis in the context of risk assessment. Several approaches have been proposed [Siu94]: extensions of the event tree/fault tree methodology (e.g. digraph-based methods), explicit state-transition methods (e.g. explicit Markov chain models), and implicit state-transition approaches (e.g. DYLAM, discrete event simulation).

3.3.1 An embedded Markov model with transition probabilities from event/fault trees

Mandelli et al. [Man06] describe a model meant for the analysis of a phased mission space propulsion system. The objective is to determine time-dependent reliability of the system over the planned mission duration. The system is modelled with two levels of Markov chain models: a high-level Markov chain (HLMC), and a set of low-level Markov chains (LLMC) built inside each state of the HLMC to compute the probabilities of the subsystems.

The time-dependent reliability of the system is computed with an embedded Markov model, and transition probabilities for the model are computed using known reliability data, and event trees based on this data.

This kind of model might suit well the emergency startup of the feedwater system. The low-level Markov chains would describe the operation of the different valves and pumps, and the high-level Markov chain would describe the operation of the entire system including the pipes

3.3.2 Methods for generating dynamic accident progression event trees

Traditionally, accident progression event trees (APET) have been static, which means that the time dimension is not accounted for. This causes that some rather gross simplifications have to be made. Determination of the sequence of events is a critical element in generating APETs. An alternative to static event trees is naturally dynamic event trees, in which the order and timing of events are determined by the progression of the accident [Hak06b].

There are several methods to generating dynamic APETs:

- DYLAM (Dynamic Logical Methodology) [Coj96] is a simulation driver capable of generating branches at user specified time intervals and coordinating the simulation of each branch. The actual software (e.g. MELCOR) that simulates the plant is called as a subroutine, and the top conditions of the system (e.g. “temperature above a certain value” or “pressure below a given threshold”) are analyzed by the driver.
- DETAM (dynamic event tree analysis method) [Aco94] is similar to DYLAM but explicitly addresses specific operator states and the evolution of these states over the course of a scenario (behaviour patterns), such as potential errors in decision making by the operating staff as influenced by the scenario dynamics, the crew’s previous decisions, the crew’s internal state (stress and confidence), and external factors (e.g. economics).
- ADS (accident dynamic simulator) [Kae96] explicitly considers operator states, but initiates branchings at times when the system or the operator takes an action (rather than prespecified times), accounts for possibility of repair, and maintains the plant history along branches to determine performance shaping factors for operator actions.
- Monte Carlo/Event tree hybrid method as implemented in the level 2 PSA analysis tool SPSA developed by STUK [Nie96]. It calculates and simulates probabilities dynamically within an event tree model. That is, the branch probabilities are not necessarily fixed but random variables. Conditional sequence frequencies are generated by simulating: the output is probability distributions for user defined output variables.

Other approaches include a DDET (discrete dynamic event tree) (where prediction error of process evolution due to branching only at user specified time intervals is quantified), DDET/MC hybrid methodology (generates all branchings but selects only some for further expansion to save computation time), and DENDROS (only branches when a setpoint for system intervention is crossed)

4 Problems of reliability assessment of digital systems

4.1 General considerations

The difficulties of including programmable automation in PSA-models are connected with the decomposition of the systems structure and with the problems of determining quantitative reliability estimates. The components or subfunctions of a programmable system are not always easily identified or described in such a way that they could be described easily as events of fault tree. Also the dynamic and multiple features of programmable functions cause problems.

4.1.1 Software vs. hardware

There are several features that distinguish software from hardware in terms of reliability (see also [Kar06]), e.g.:

- software errors are design flaws rather than a result of wear-and-tear.

- software errors manifest themselves only under specific (rare) circumstances. The circumstances are rare because testing has usually revealed frequently-occurring errors at development time.
- each software application is, at least to an extent, unique. Therefore experience with similar software applications is of limited use at best.

For these and many other reasons, reliability models developed for equipment are not applicable when analyzing the reliability of systems containing software. This applies even more generally to digital systems: the main emphasis in reliability studies should be on design issues.

4.1.2 Reliability at the systems level

There are at least two levels at which the reliability of digital systems can be considered:

- digital system level. This covers the reliability of communication, and processing within the digital system itself.
- whole system level. This covers both the digital system and its environment: the system that is controlled (the controlled plant physical processes), the human-machine interface etc.

Different methods and models have been developed and are applicable on these two levels.
It is evident that when assessing the reliability of NPP control systems, the digital system level is not sufficient, because the behaviour of the controlled system plays a crucial role in determining the behaviour of the control system.

4.2 Special features of the nuclear sector

Safety authorities require plant specific PSAs in many countries, and quantitative safety goals are set either on core melt frequency or safety function reliability level. The Finnish safety authority (STUK) requires plant specific PSAs, and quantitative target values are set on different levels. There is a need to include both safety automation systems and safety related control systems in a PSA model. In case of NPP safety automation systems reliability assessment has been studied in many research projects and some methods have been developed to qualify and license programmable systems. [Haa97,Hel03].

In current PSAs, distributed control systems are analysed and modelled very simply. In many cases, the starting point for modelling is a reliability analysis made by the vendor. Including the vendor’s analysis in a PSA is not a straight-forward task. Other questions are how to link the model to reliability models of other I&C systems and user interfaces (control room) and how the architecture of I&C systems affects, e.g., the possibility of common cause failures.

In the case of programmable systems, the usual reliability modelling principles are difficult to apply. The hardware of programmable systems can be identified, and fault trees can be constructed to describe the failures of the hardware. The decomposition of software-based systems into components is not straightforward. It may be possible to identify parts of the software, i.e. the software functions. Examples of this are the platform/system software and the application software. However, the application software functions may be located to several processors, which introduce dependencies between safety functions. In principle, this can be described in fault-trees, but there is not much experience in this type of modelling.

Digital automation systems and components are dynamic systems, in which software and hardware interact. Their reliability models would likely need to include both the normal and failure behaviour of both the hardware and software, in order to properly account for all of the interdependencies of the systems.

A particular problem is how one can utilise the results from safety analysis of the existing system. A general requirement is that the reliability of the new equipment should be at least as good as of the old. However, even slight changes in the system's logic might lead to very different behaviour from the reliability point of view.

Redundancy is often used in conventional systems to obtain high reliability. A basic assumption is that redundant channels fail independently, and the problem is the potential existence of (rare) common cause failures. Software may be a typical source of common cause failures, if it is the same for all trains/channels.

The failures of programmable automation systems are connected to accidents in several ways. They may cause initiating events and even so called common cause initiators, which cause both an initiating event and simultaneous unavailability of redundant safety functions. The failures may be also latent, i.e. they have actually occurred in past before initiating event. As the failures of hardware components, also the failures of programmable systems may be dependent on each other. The common cause failures of programmable systems form a very difficult problem area.

Unplanned dependencies between software artefacts are an important source of faults in programmable systems. An example from another field illuminates the matter. The self-destruction of Ariane 5 rocket in 1996 was caused by execution of a data conversion from a 64-bit floating-point number to a 16-bit signed integer value. The value of the floating-point number was greater than what could be represented by a 16-bit signed integer. The result was an operand error. The data conversion instructions (in Ada code) were not protected from causing operand errors, although other conversions of comparable variables in the same place in the code were protected. The part of the software which caused the accident was based on a requirement of previous rocket Ariane 4. The requirements were however different in this case. Complexity of the software based system and lack of management of requirements were key factors which lead to the accident.

Unplanned dependencies may be prevented by following strict software engineering discipline in development, and can be detected by the rigorous use of formal methods and by thorough testing. However, it is it is as of yet unclear how to incorporate them or their effects in quantitative reliability or risk analysis models.

Another issue is that the reliability of programmable systems is a property of the system's operational environment as well as that of the system itself. In other words, the reliability of programmable systems depends on the operational profile, which as the probability distribution of input sequences, varies from one environment to another and is generally difficult to capture, especially in the design phase. This restricts the use of generic operational experience in determination of reliability parameters.

4.3 Case feedwater system control of a BWR

The case we will consider is the feedwater control system of Olkiluoto 1&2 nuclear power plant. However, the system will only be used as an example: the resulting model is very simplified, and no conclusions on the particular plant can be made based on the present study.

4.3.1 Description of the process and automation systems modelled

The feed water system (system 312, system 445) consists of a feedwater circuit (figure 2). The feed water flow is induced by four parallel-coupled feed water pumps 445 P301-P304 and the feed water flow is controlled by two control valves 312 V14 and V312 V17. A shut-off valve 312 V13, controlled by feed water flow, steam flow and certain RPS signals (reactor scram), is fitted in parallel with the control valves. The recirculation flow is controlled by serially coupled control valve 445 V371 and shut-off valve 445 V379. Pumps 441P1-P4 are needed to raise the water pressure to a level suitable for 445P301-P304; if the pressure is too low, the latter cannot operate.

The condensate system (system 441) has no safety task but in some abnormal situations and disturbances it is needed to utilize the water volume in condenser and to keep the normal feedwater system functioning. The duty of the condenser circuit is thus to ensure the adequate suction pressure of feedwater pumps by keeping sufficient number of condensate pumps 441P1-P4 on (figure 2).

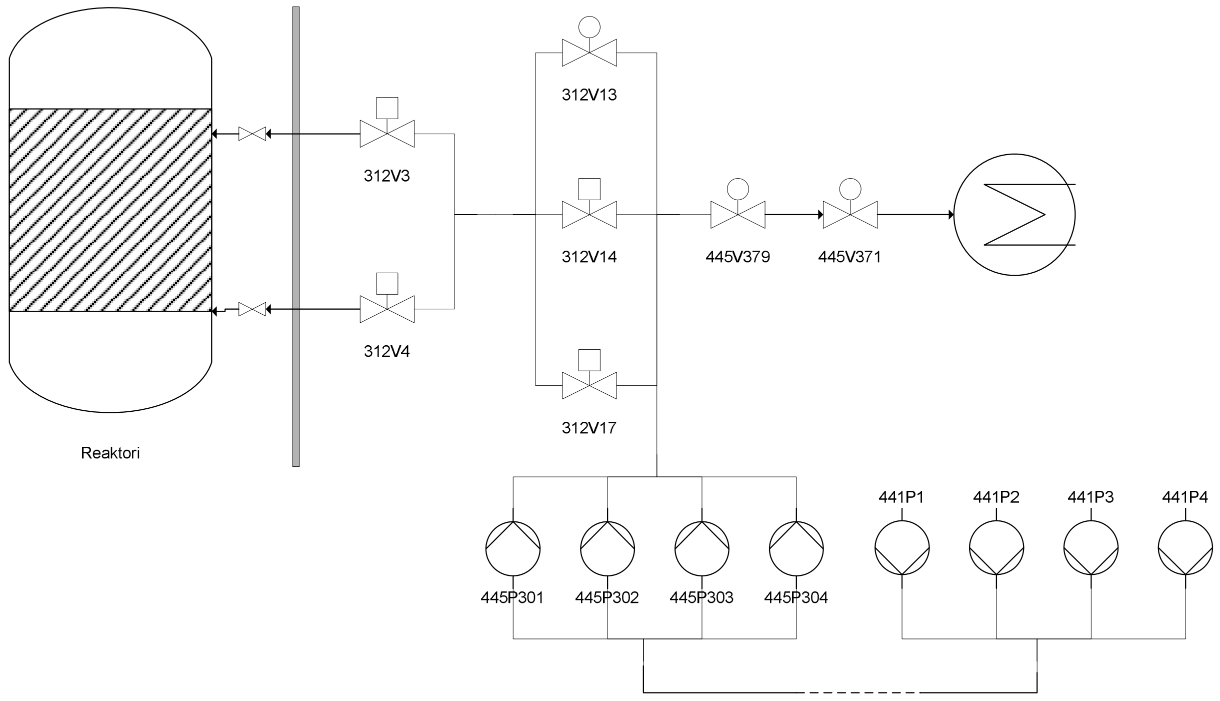


Figure 2. The feedwater circuit of the Olkiluoto 1&2. Only the most essential parts are shown.

The feedwater control system (system 537) controls the speeds of the feed water pumps and the opening of the control valves (figure 3). The feedwater control system receives information from transducers, monitors and the safety system of the power station. These signals are processed in a master controller consisting of various sub-units, in the succeeding slave controllers and in a low power controller. The slave controllers apply control signals to the actuators of the hydraulic couplings driving the feed water pumps and to the actuator of the control valves.

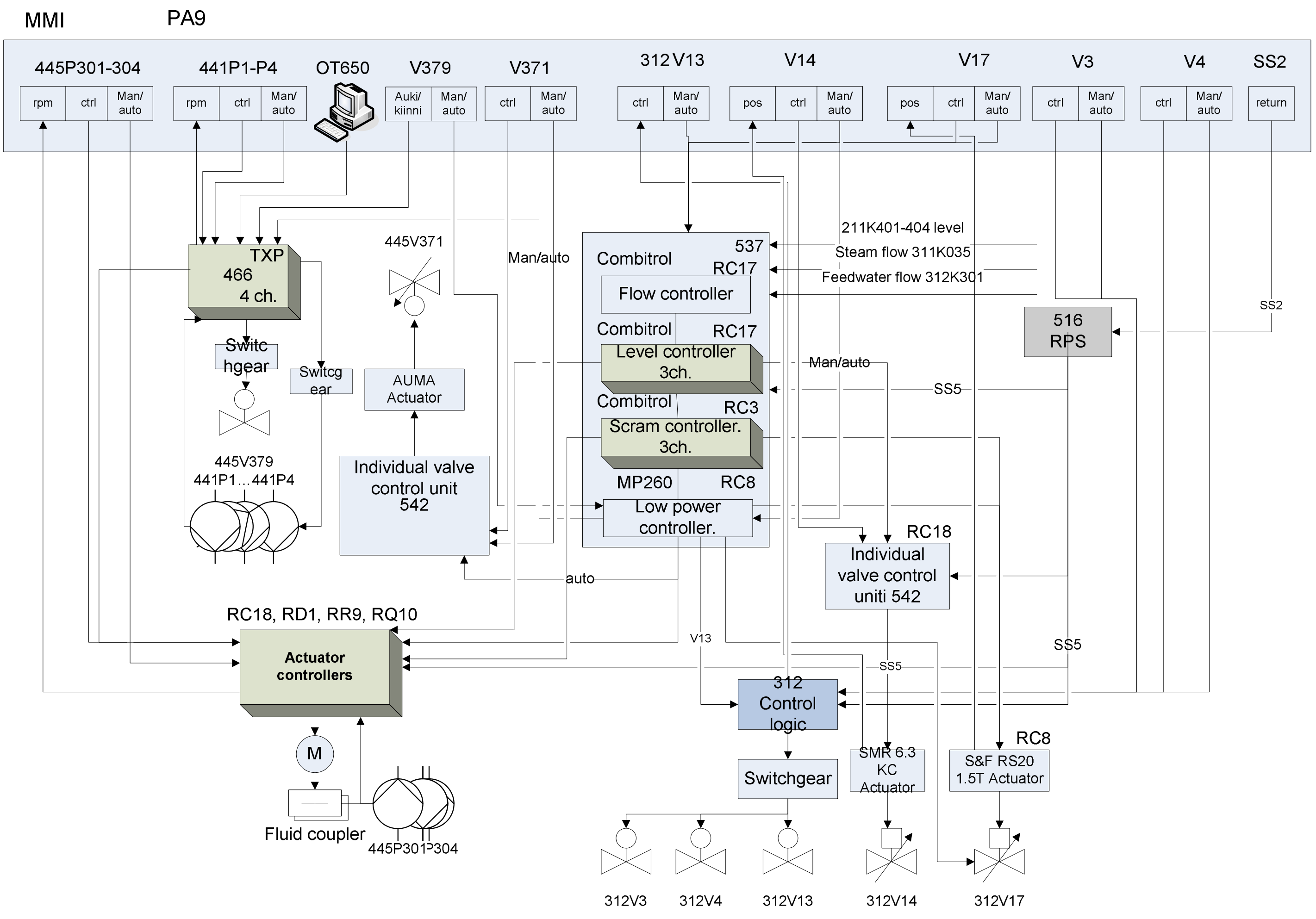


Figure 3. The feedwater control system of the Olkiluoto 1&2.

The master controller consists of a level controller and a flow controller. The two controllers together form a level control circuit, which maintains the water level in the reactor pressure vessel at the required value. The master controller is implemented by conventional ASEA Combitrol technology and the individual control units and actuator controls are implemented by conventional Combimatic and other technologies.

The feedwater flow is normally controlled by adjusting the speed of the feedwater pumps. Control valves 312 V14, 312 V17 and shut-off valve 312 V13 in the feedwater pipework are then fully open. At feedwater flows below about 16 % of full flow, the feedwater flow is controlled by a combination of differential pressure control of the feedwater pump used and flow and level control of the control valves.

The programmable ABB MasterPiece 260 system is intended only for low power use at startup and shutdown. When in operation (feedwater flow below about 16 % of full flow), it controls the feedwater pump in use to keep a constant differential pressure between the feedwater pump and the reactor pressure vessel, and the control valves which are controlling feedwater flow and reactor water level. In addition, it controls the shut-off valve 312 V13 and the recirculation valves 445 V379 and 445 V371.

The Teleperm XP (system 466) is a normal process control automation system whose purpose is to control the operation of the turbine plant. In this case it is intended to perform the control task in starting, stopping and doing a switch over of the feedwater and condensate pumps (figure 3).

4.3.2 Description of the transient modelled in plant PSA

We consider the startup of the feedwater system after a reactor scram.

The main feedwater system is not designed as a safety system, and is not usually credited in deterministic safety analyses. However, the main feedwater system can be utilized for high pressure reactor core cooling in certain types of plant disturbances e.g. in case of initiating events during power operation. After an initiating event, where also reactor scram occurs, the main feedwater system either stops or continues operation depending on accident progression (the required mission time is 24 hours in PSA).

A typical initiating event derived from plant PSA is the loss of offsite power. The transient starts with loss of 400 kV line, in which a break exists in onsite power supply i.e. the 6 kV (system 642) main bus bars. The onsite power (642) can be recovered depending on the cause of disturbance and the state of 400 and 110 kV grid.

In the event of the station tripping from the grid, the plant will change over to a house turbine operation. The feedwater control equipment will remain in operation and the flow will be reduced to below 40 % of full flow in ~30 s, since power control system (system 535) will reduce the reactor power by reducing the speeds of the recirculation pumps and the partial scram is actuated.

In the event of failure to change over to house turbine operation and simultaneous loss of supply from the normal auxiliary power system, the pumps in the feedwater system will stop. Reactor scram will take place and the auxiliary feedwater system (system 327) and core spray system (system 323) will be started, with power supply from the diesel-backed system. In this case, the feedwater control system will be inoperative, since the feedwater pumps have stopped. All slave controllers for the feedwater pumps will run down the actuators of the hydraulic couplings to the predetermined set value for the condition when the pump is shut down.

In very short breaks of onsite power supply, when house turbine operation fails, but switchover to 110 kV is successful, automatic restart of the main feed water system should operate in order to avoid the loss of feed water system. In case of failure of switchover or some other faults in external grid, that cause longer break in 642 bus bars, the main feed water system can be restarted manually, if necessary after the recovery of the system 642. In the case of total loss of core cooling (i.e., failure of the auxiliary feedwater system and core spray system), the feedwater pumps should be put into operation within about 30 minutes and the flow to the reactor should be at least 20 kg/s. After successful restart, the makeup water to condenser is needed within a certain time window to ensure the main feedwater operation.

4.3.3 Description of the the existing reliability analysis of feedwater restart

A reliability analysis has been made for some control functions of feedwater and condensate system. The aim of the subcontractor's analysis was to quantify the reliability of the Teleperm XP (TXP) system (system 466) in the control task in starting, stopping and doing a switch over of the feed water and condensate pumps. The TXP is a normal process control system whose purpose is to control the operation of the turbine plant.

The reliability analysis was focused to the functions, which are required to be performed by the system during an accident. The main feedwater system can be utilized for high pressure reactor core cooling after the reactor trip.

The reliability analysis is performed for the following top events:

- system fails to restart the high pressure feed water pumps with associated equipment; the success criterion is 1 out of 4 trains
- system fails to restart the condensate pumps with associated equipment; the success criterion is 1 out of 4 trains

The results of the supplier's analysis is detailed but covers only the assessment of probabilities for feedwater pump and condensate pump startup after very short breaks of onsite power supply. The time window of the transient modelled in plant PSA is longer and the order of starting different systems is simplified in plant PSA so that main feedwater comes to operation after the safety systems (327, 323) have started.

If the feedwater starts or fails to start immediately after a short break of electrical power, the consequences may be different than in the case of manual start after 30 minutes. Some kind of dynamic approach should be useful for modelling the feedwater startup sequences. The supplier's analysis covers also the feedwater and condensate pumps, which represents only a small part of the circuit that is intended to operate during the transient. How to use the results of the supplier's analysis in plant PSA model is not a straightforward task.

5 Application case

This section describes a model of the feedwater system in the dynamic flowgraph methodology formalism (see section 3.2). Also a Markov model was considered and some modelling was done, but it was decided that, at least at this phase, effort is concentrated on the DFM model. This decision was justified by noting that probability data for the Markov model may not be available, and the modelling effort of a Markov model seems to be higher than that of a DFM model.

5.1 Model scope

The model is a simplified representation of the feedwater system of the Olkiluoto 1&2 units. The purpose of the model is to aid in generating cut sets and structure of fault trees.

The model was constructed to analyze restart after a short term loss of offsite power transient, which causes a reactor scram and stop of the feedwater system. Several simplifications have been made. The full power flow control valve, 312V13, has been left out because it does not participate in the restart. Only one of the four pumps 445P1-4 has been incorporated in the model because even one working pump will supply sufficient water flow, and therefore the rest only affect the reliability and availability of the pump subsystem.

A significant simplification has also been made in the number of states of the process parameter variables. Currently, no variable has more than 3 states. This is typical in reliability analysis, since the primary purpose is to analyse particular system states, and not to accurately model process dynamics. Determining the sufficient amount of states for each variable is a case specific issue.

It should be emphasized that this model has been constructed to demonstrate the principles and study the applicability of the dynamic flowgraph methodology (see section 3.2). The results should not at this phase be used in evaluating the reliability of the feedwater system.

5.2 Model architecture

The model is depicted in figure 4.

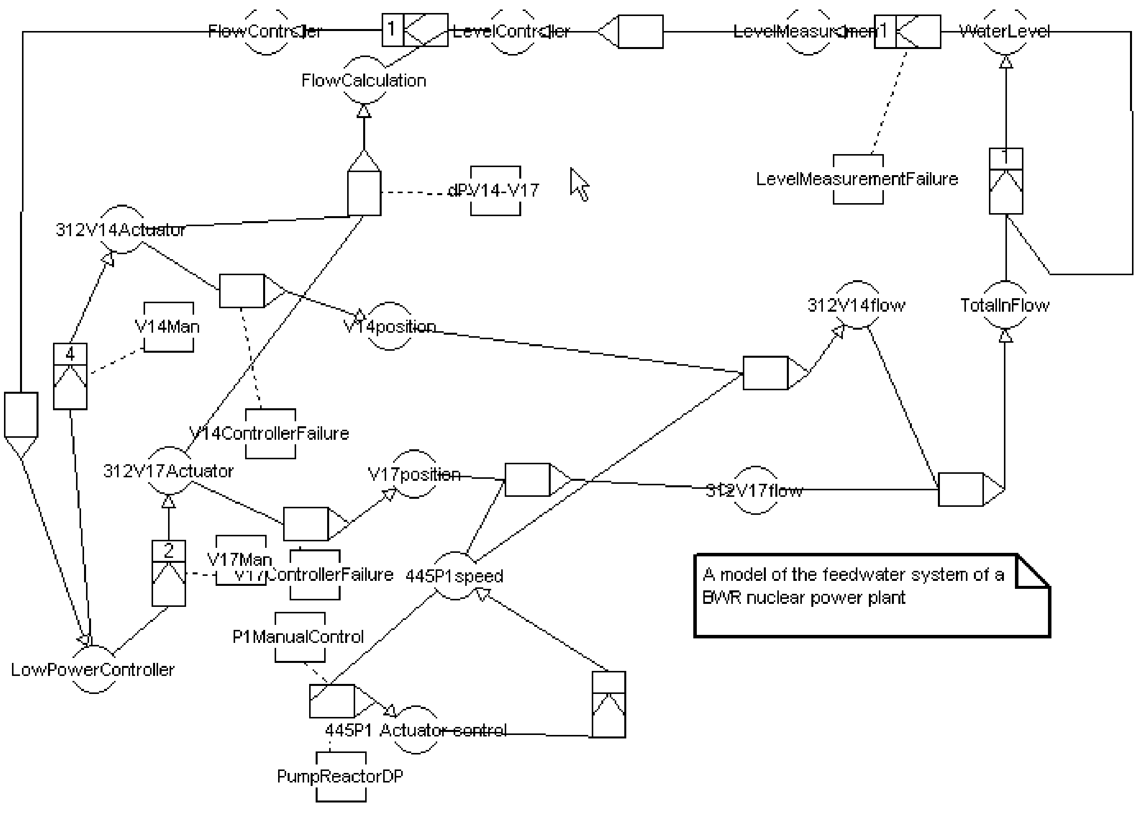


Figure 4. A DFM model of the feedwater system

The variables are explained in table 1.

Variable name	Description	Possible states
312V14Actuator	the direction to which the actuator tries to direct the valve position	increase, stationary, decrease
312V14Flow	water flow through the valve 312V14	high, normal, low
312V17Actuator	Action of the controller that controls valve 312V17. This contains both valve and control circuit	increase, stationary, decrease

Variable name	Description	Possible states
312V17Flow	water flow through the valve 312V17	high, normal, low
445P1ActuatorControl	control signal for the pump 445P1	increase, stationary, decrease
445P1Speed	The rotation speed of the pump 445P1	high, normal, low
dPV14-V17	Fault condition variable: Is the differential pressure between the valves 312V14 and 312V17 computed incorrectly?	false, true
FlowCalculation	a software program that computes the flow to the reactor	high, normal, low
FlowController	control signal to the low power controller	increase, stationary, decrease
LevelController	control signal of the water level in the reactor	increase, stationary, decrease
LevelMeasurement	measurement of the water level in the reactor	too_low, normal, too_high
LevelMeasurementFailure	does the level measurement work incorrectly?	false, true
LowPowerController	control signal for the situations the reactor is not producing electricity	increase, stationary, decrease
P1ManualControl	is the pump 445P1 controlled manually?	false, true
PumpReactorDP	A signal from TXP to start up the pump	works, fails
TotalInFlow	Total flow into the reactor	high, normal, low
V14ControllerFailure	Fault condition variable: Does the controller fail to give signal to valve 312V14?	yes, no
V14Man	is the valve 312V14 controlled manually?	false, true
V14Position	the relative position of the valve 312V14	too_open, appropriate, too_close
V17ControllerFailure	Fault condition variable: Does the controller of the valve 312V17 give the correct signal to the servo controlling the valve's position?	false, true
V17Man	Fault condition variable: Is the valve 312V17 in manual control?	false, true
V17Position	the position of the valve 312V17 relative to the situation	too_open, appropriate, too_close
WaterLevel	Water level in the reactor	too_low, normal, too_high

Table 1. The variables of the simple DFM model. Each variable corresponds to a node in figure **4**.

A sample interpretation of the variable values is given for WaterLevel in Table 2.

State	Interpretation
too_high	the water level in the reactor is > 9 meters (from the top of the reactor core)
normal	the water level in the reactor is between 5 and 9 meters
too_low	the water level is below 5 meters

Table 2. Interpretations of the possible values for the state WaterLevel

A sample decision table is given for the water level in reactor in table 3.

TotalInFlow @ T=t-1	WaterLevel @ T=t-1	WaterLevel @ T=t
high	too_low	normal
high	normal	too_high
high	too_high	too_high
normal	too_low	too_low
normal	normal	normal
normal	too_high	too_high
low	too_low	too_low
low	normal	too_low
low	too_high	normal

Table 3. The decision table for the variable WaterLevel.

The model contains three kinds of variables:

- physical state variables describe a physical variable such as amount of flow, a measurement variable, or an electrical signal. In this model, these are 312V14Flow, 312V17Flow, 445P1Speed, 445P1ActuatorControl, LevelMeasurement, LevelController, TotalInFlow, V14Position, V17Position and WaterLevel
- logical variables. These are variables that describe the behaviour of a software or networking component. In this model, these are dPV14-V17, FlowCalculation, FlowController, LevelController, LowPowerController, P1ManualControl, V14Man, and V17Man
- fault condition variables. These are indicator variables that tell whether the system component they are attached to is working properly or not. In this model, these are LevelMeasurementFailure, V14ControllerFailure, V17ControllerFailure and PumpReactorDP.

As an example top event for the system, consider the case that the water level of the reactor is too low for three consecutive time instances. This is expressed as WaterLevel=too_low @ t=0 AND WaterLevel=too_low @ t=-1 AND WaterLevel=too_low @ t=-2.

When setting the total number of analysis periods to 3, DYMONDA gives 2082 prime implicants for this top event. Two sample prime implicants are listed in Table 4 and Table 5.

Node	**State**	**Time**
312V14Actuator	decrease	-3
312V17Actuator	decrease	-3
445P1speed	low	-3
PumpReactorDP	Incorrect measurement	-3
WaterLevel	normal	-3
PumpReactorDP	Incorrect measurement	-2

Table 4. Prime implicant # 3 for the top even described above

The prime implicant of Table 4 can be interpreted as follows. If the control signals for both valve 312V14 and 312V17 try to decrease the flow through the valves, and the pump speed is low, water level is normal and the PumpReactorDP signal fails for two consecutive moments, the top event of water level being too low for three consecutive time instances occurs.

Note that the initial conditions of the model can be inserted as a part of the top event; thus, if we need to state that initially, the actuator of the valve 312V14 is stationary, we can insert the condition 312V14Actuator=stationary @ t=-3 to the top event.

Node	State	Time
445P1speed	normal	-3
V14ControllerFailure	No	-3
V17ControllerFailure	F	-3
WaterLevel	too_low	-3
312V17Actuator	decrease	-2
PumpReactorDP	Fails	-2
V17ControllerFailure	T	-2

Table 5. Prime implicant # 2080 for the top event described above

The prime implicant of Table 5 can be interpreted as follows. If water level is too low, and in the next time step the controller of the valve V17 fails and also pump reactor DP fails, the top event of water level being too low for three consecutive time instances occurs.

An interesting feature of DFM that was found experimentally is that the running time of the algorithm can be cut quite dramatically by setting initial and boundary conditions for certain variables.

5.3 Model implementation

The model was implemented in DYMONDA, available from Asca, Inc. (http://www.ascainc.com/). The software is currently in beta version, and it was received by VTT for testing and evaluation.

DYMONDA also contains the capability of computing the top event probability from cause probabilities. However, as of the writing of this report, this functionality is not functioning correctly; therefore, computation of the probabilities of the top events and evaluation of the method are not included in this report. As explained earlier, the computation of probabilities of prime implicants goes like the computation of probabilities of minimal cut sets..

6 Discussion

The main benefits of dynamic flowgraph methodology (DFM) are

- the simplicity of its formalism: the main entities – discrete state space, state transition tables and delays – are immediately graspable
- possibility to model time-dependencies and loop dependencies
- possibility to model multi-state logic and incoherent reliability structures
- a single system model can be used to analyse different top events
- expressive power: in practice, most logical or physical entities can be modelled to arbitrary accuracy.

The main issues concerning DFM are

- the amount of computation may become excessive even when the model size is moderate. This can be ameliorated by careful modelling of variables, connections and the top event, but this requires effort and expertise.
- only one working implementation of DFM has been published, and the source code is not available.
- systematic methods for making modeling decisions such as selecting the number of states for a variable has not been presented thus far.

The scope of applications of DFM is quite large. In addition to those outlined in section 3.2, DFM can be used for diagnosis: determining which event(s) caused the failure of a system. DFM could also be used to compute rough performance measures for a system, such as the mean delay in completing a task, the expected number of times that a variable visits a certain (costly) state when a system is performing a given task, etc. An intriguing research direction would also be to find out how to evaluate proposed software architecture with DFM before the software is implemented.

DFM seems a promising way to assess the reliability of digital systems: its central elements – state transitions based on a multivalued propositional logic and time delays – would seem sufficient for modelling a control system to desired accuracy. Its representation is more geared towards modelling of hardware-software systems such as the control systems at an NPP than e.g. the models commonly used in model checking. However, it still remains uncertain whether the only working DFM software known to the authors, DYMONDA, produces valid results; thus far, the results seem quite promising but a rigorous validation is still a task for the future.

Though it has been found that the DFM program DYMONDA is usable at least for moderate-sized problems, some research issues need still to be addressed. First, how well DFM scales up to realistic-size problems is an open issue. It seems that the feedwater control system can be modelled with DFM to a sufficient accuracy. It still remains an open question whether sufficiently accurate DFM models of the feedwater control system are computationally tractable. Second, how tedious it is to arrive at top-event probability estimates from the prime implicants. Third, how to integrate the results of DFM (including probability estimates) with event and fault tree approaches generally used in the nuclear sector.

It could be worth considering whether a program implementation of some other methodology (e.g. one developed for model checking) could be used for analyzing a DFM model. Another avenue of research would be to develop a program implementation of DFM.

If a DFM program is implemented, it could contain also various extensions to DFM. For example, an interface that allowed actual control programs to be executed as part of DFM would be worth considering. In practice, the outputs of the program would be discretized to work as inputs to DFM, and the outputs of DFM to the program would be somehow generated from the states of the DFM input variables to the program. This would allow the reliability of the actual program (rather than that of a simplified model of it) to be analyzed.

One line of research would be to combine a simulation model of the system with a DFM model. Here, the simulation model are used to compute the values of variables of interest, and the DFM model is used to track the logical part of the model based on these variable values. These hybrid models would have the advantage that the physical properties are modelled accurately while keeping the logical part of the model tractable.

A fifth line of research would be trying to develop a formal, or at least a systematic, method for constructing DFM models. A DFM model could perhaps be constructed from a flow diagram of the system, logic descriptions of the discrete (e.g. software) variables, and physical models of the continuous variables. Methods of qualitative reasoning (a subfield of artificial intelligence) could be tried; whether appropriate methods have been developed in qualitative reasoning and whether they are applicable in DFM model construction, remains a research issue.

Sixth, it would be interesting to evaluate how well a simple model fares up in comparison to a more sophisticated (and larger) model. The simplest way to do this would be to construct a simplified model of the feedwater system, and then find out how many of the prime implicants considered important in the larger model would be found by analyzing the simpler model. Of course, here we should consider two prime implicants similar if they represent the same sequence of events. In a more sophisticated analysis, the number of nodes, states in each node, and edges would be varied systematically; this would be based on some kind of formal method of simplification.

7 Conclusions

The problem of assessing the reliability of digital control systems has been considered. A dynamic flowgraph methodology (DFM) model has been considered for an application case, which was a feedwater system for a BWR. A DFM model can be used to generate timed fault trees; an initial model for finding prime implicants for a limited system function has been constructed.

The main benefits of DFM are

- the simplicity of its formalism: the main entities – discrete state space, state transition tables and delays – are immediately graspable
- possibility to model time-dependencies and loop dependencies
- possibility to model multi-state logic and incoherent reliability structures
- a single system model can be used to analyse different top events

The main issues concerning DFM are

- the amount of computation may become excessive even when the model size is moderate. This can be ameliorated by careful modelling of variables, connections and the top event, but this requires effort and expertise.
- only one working implementation of DFM has been published, and the source code is not available.

The scope of applications of DFM is quite large. DFM seems a promising way to assess the reliability of digital systems. Its representation is more geared towards modelling of hardware-software systems such as the control systems at an NPP than e.g. the models commonly used in model checking. However, it still remains uncertain whether the only working DFM software known to the authors, DYMONDA, produces valid results; thus far, the results seem quite promising but a rigorous validation is still a task for the future.

Though it has been found that the DFM program DYMONDA is usable at least for moderate-sized problems, some research issues need still to be addressed: how well DFM scales up to realistic-size problems is an open issue, how tedious it is to arrive at top-event probability estimates from the prime implicants, how to integrate the results of DFM (including probability estimates) with event and fault tree approaches generally used in the nuclear sector.

References

[Aco94] Acosta, C. and N. Siu. Dynamic event trees in accident sequence analysis: application to steam generator tube rupture. *Reliability Engineering and System Safety*, Vol. 41 (1994), No. 2, 135-154.

[Ald07] Aldemir, T., M.P. Stovsky, J. Kirschenbaum, D. Mandelli, P. Bucci, L.A. Mangan, D.W. Miller, X. Sun, E. Ekici, S. Guarro, M. Yau, B. Johnson, C. Elks, and S.A. Arndt. Dynamic reliability modeling of digital instrumentation and control systems for nuclear reactor probabilistic risk assessments. NUREG report NUREG/CR-6942, October 2007. Available in the www as http://www.nrc.gov/reading-rm/doc-collections/nuregs/contract/cr6942/cr6942.pdf

[Cla99] Clarke, E.M., Grumberg, O. and D.A. Peled. *Model checking*. MIT Press, Cambridge 1999.

[Coj96] Cojazzi, G. The DYLAM approach for the dynamic reliability analysis of systems. *Reliability Engineering and System Safety*, Vol. 52 (1996), No. 3, 279-296.

[Eas97] Easterbrook, S. and Callahan, J. Formal methods for V&V of partial specifications: an experience report. Proceedings of the Third IEEE International Symposium on Requirements Engineering, 6-10 Jan. 1997, 160-168.

[Gar95] Garrett, C.J., S.B. Guarro and G.E. Apostolakis. The dynamic flowgraph methodology for assessing the dependability of embedded software systems. *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 25, No. 5, May 1995, 824-840.

[God96] Goddard, P.L. A combined analysis approach to assessing requirements for safetycritical real-time control systems. Reliability and Maintainability Symposium, Proceedings. Annual International Symposium on Product Quality and Integrity, IEEE Press 1996.

[Haa97] Haapanen, P., Pulkkinen, U. & Korhonen, J. Usage models in reliability assessment of software-based systems. Finnish Center for Radiation and Nuclear Safety (STUK), technical report STUK-YTO-TR 128, April 1997.

[Hak06b] Hakobyan, A. Severe accident analysis using dynamic accident progression event trees. PhD Dissertation, The Ohio State University 2006. http://www.ohiolink.edu/etd/send-pdf.cgi?acc_num=osu1158672136 (fetched 27.9.2007)

[Hak06] Hakobyan, A., R. Denning, T. Aldemir, S. Dunagan and D. Kunsman. A methodology for generating dynamic accident progression event trees for level-2 PRA. PHYSOR-2006, ANS Topical Meeting on Reactor Physics. Vancouver, BC, Canada, September 10-14, 2006.

[Hei05] Heitmeyer, C. A panacea or academic poppycock: formal methods revisited. Proceedings of the Ninth IEEE International Symposium on High-Assurance Systems Engineering (HASE '05), 12-14 Oct. 2005, pages 3-7.

[Hel03] Helminen, A. & Pulkkinen, U. Reliability assessment using Bayesian networks – case study on quantitative reliability estimation of a software-based motor protection relay. Finnish Center for Radiation and Nuclear Safety (STUK), technical report STUK-YTO-TR 198, June 2003.

[Hou00] Houtermans, M., G. Apostolakis, A. Brombacher and D. Karydas. Programmable electronic system design & verification utilizing DFM. In *SAFECOMP 2000* (F. Koornneef and M. van der Meulen, eds.), Lecture Notes in Computer Science 1943 (2000), 275-285.

[Hou02] Houtermans, M., G. Apostolakis, A. Brombacher, and D. Karydsas. The dynamic flowgraph methodology as a safety analysis tool: programmable electronic system design and verification. *Safety Science*, Vol. 40 (2002), 813-833.

[Kae96] Kae-Sheng, K. and A. Mosleh. The development and application of the accident dynamical simulator for dynamic probabilistic risk assessment of nuclear power plants. *Reliability Engineering and System Safety*, Vol. 52 (1996), No. 3, 297-314.

[Kar06] Karanta, I. Methods and problems of software reliability estimation. VTT Working Paper 63, Espoo 2006.

[Kor04] Kordon, F. and M. Lemoine (eds.). *Formal Methods for Embedded Distributed Systems - How to master the complexity*. Kluwer, Dortrecht 2004.

[Kui94] Kuipers, B. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. MIT Press, Cambridge, MA 1994.

[Lyu92] Lyu, M.R. & Nikora, A. Applying reliability models more effectively. *IEEE Software*, Vol. 9 No. 4 (July 1992), 43–52.

[Man06] Mandelli, D., T. Aldemir and E. Zio. An event tree/fault tree/embedded Markov model approach for the PSAM-8 benchmark problem concerning a phased mission space propulsion system. Proceedings of the 8th International Conference on Probabilistic Safety Assessment and Management, May 14-18, 2006, New Orleans, USA. ASME 2006, 9 pp.

[Nie96] Niemelä, I. SPSA level 2 user's guide. Addendum to SPSA user's guide. 51 p. Helsinki: Finnish centre for radiation and nuclear safety, 1996.

[Rii07] Riis Nielson, H. and F. Nielson. *Semantics with applications: an appetizer*. Springer, Berlin-Heidelberg-New York 2007.

[Siu94] Siu, N. Risk assessment for dynamic systems: an overview. *Reliability Engineering and System Safety*, Vol. 43 (1994), No. 1, 43-73.

[Smi04] Smidts, C. S. and M. Li. Preliminary Validation of a Methodology for Assessing Software Quality (NUREG/CR-6848). U.S. Nuclear Regulatory Commission, July 2004. Available as http://www.nrc.gov/reading-rm/doc-collections/nuregs/contract/cr6848/cr6848.pdf

[Smi05] Smidts, C.S. Integrating Software into PRA: A Test-Based Approach. *Risk Analysis*, Vol. 25, No. 4, pp. 1061-1077, August 2005

[Swa99] Swaminathan, S. and C.S. Smidts. The mathematical formulation of the event sequence diagram framework. *Reliability Engineering and System Safety*, Vol.65 (1999), 103-118.

[Val08] Valkonen, J., Pettersson, V., Björkman, J. Holmberg, J.-E., Koskimies, M., Heljanko, K. and Niemelä, I. Model-Based Analysis of an Arc Protection and an Emergency Cooling System – MODSAFE 2007 Work Report. VTT Working Papers 93, 2008.

[Yau95] Yau, M., S. Guarro and G. Apostolakis. Demonstration of the dynamic flowgraph methodology using the Titan II space launch vehicle digital flight control system. *Reliability Engineering and System Safety*, Vol. 49 (1995), 335-353.

Appendix A: Decision tables of the feedwater system model

Table 6: 312V14Actuator (lag 4)

LowPowerController	V14Man	312V14Actuator
increase	F	increase
increase	T	stationary
stationary	F	stationary
stationary	T	stationary
decrease	F	decrease
decrease	T	stationary

Table 7:312V14Flow

V14position	445P1speed	312V14flow
too_open	high	high
too_open	normal	high
too_open	low	normal
appropriate	high	high
appropriate	normal	normal
appropriate	low	low
too_close	high	normal
too_close	normal	low
too_close	low	low

Table 8: 312V17Actuator

LowPowerController	V17Man	312V17Actuator
increase	F	increase
increase	T	stationary
stationary	F	stationary
stationary	T	stationary
decrease	F	decrease
decrease	T	stationary

Table 9: 312V17flow

V17position	445P1speed	312V17flow
too_open	high	high
too_open	normal	high
too_open	low	normal
appropriate	high	high
appropriate	normal	normal
appropriate	low	low
too_close	high	normal
too_close	normal	low
too_close	low	low

Table 10: 445P1 Actuator control

PumpReactorDP	*P1ManualControl*	*445P1speed*	*445P1 Actuator*
Works	*F*	*high*	*decrease*
Works	*F*	*normal*	*stationary*
Works	*F*	*low*	*increase*
Works	*T*	*high*	*decrease*
Works	*T*	*normal*	*stationary*
Works	*T*	*low*	*increase*
Fails	*F*	*high*	*decrease*
Fails	*F*	*normal*	*decrease*
Fails	*F*	*low*	*decrease*
Fails	*T*	*high*	*decrease*
Fails	*T*	*normal*	*decrease*
Fails	*T*	*low*	*decrease*

Table 11: 445P1speed (lag 1)

445P1 Actuator control	445P1speed
increase	high
stationary	normal
decrease	low

Table 12: FlowCalculation

312V14Actuator	312V17Actuator	dPV14-V17	FlowCalculation
increase	increase	F	high
increase	increase	T	high
increase	stationary	F	high
increase	stationary	T	normal
increase	decrease	F	normal
increase	decrease	T	low
stationary	increase	F	high
stationary	increase	T	normal
stationary	stationary	F	normal
stationary	stationary	T	normal
stationary	decrease	F	low
stationary	decrease	T	normal
decrease	increase	F	high
decrease	increase	T	normal
decrease	stationary	F	low
decrease	stationary	T	normal
decrease	decrease	F	low
decrease	decrease	T	low

Table 13: FlowController (lag 1)

LevelController	FlowCalculation	FlowController
increase	high	increase
increase	normal	increase
increase	low	increase
stationary	high	decrease
stationary	normal	stationary
stationary	low	increase
decrease	high	decrease
decrease	normal	decrease
decrease	low	decrease

Table 14: LevelController

LevelMeasurement	LevelController
too_low	increase
normal	stationary
too_high	decrease

Table 15: LevelMeasurement (lag 1)

WaterLevel	LevelMeasurementFailure	LevelMeasurement
too_low	F	too_low
too_low	T	normal
normal	F	normal
normal	T	normal
too_high	F	too_high
too_high	T	normal

Table 16: LowPowerController

FlowController	LowPowerController
increase	increase
stationary	stationary
decrease	decrease

Table 17: TotalInFlow

312V14flow	312V17flow	TotalInFlow
high	high	high
high	normal	high
high	low	normal
normal	high	high
normal	normal	normal
normal	low	normal
low	high	normal
low	normal	normal
low	low	low

Table 18: V14position

312V14Actuator	V14ControllerFailure	V14position
increase	N	appropriate
increase	Y	too_open
stationary	N	appropriate
stationary	Y	appropriate
decrease	N	appropriate
decrease	Y	too_close

Table 19: V17position

312V17Actuator	V17ControllerFailure	V17position
increase	F	appropriate
increase	T	too_open
stationary	F	appropriate
stationary	T	appropriate
decrease	F	appropriate
decrease	T	too_close

Table 20: WaterLevel (lag 1)

TotalInFlow	WaterLevel	WaterLevel
high	too_low	normal
high	normal	too_high
high	too_high	too_high
normal	too_low	too_low
normal	normal	normal
normal	too_high	too_high
low	too_low	too_low
low	normal	too_low
low	too_high	normal