Department of Automation and Systems Technology

System design and risk assessment for safety critical control software product lines

Nikolaos Papakonstantinou

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 148/2012

System design and risk assessment for safety critical control software product lines

Nikolaos Papakonstantinou

A doctoral dissertation completed for the degree of Doctor of Science (Technology) (Doctor of Philosophy) to be defended, with the permission of the Aalto University School of Electrical Engineering, at a public examination held at the lecture hall AS1 of the school on the 23rd of November 2012 at 12:15.

Aalto University
School of Electrical Engineering
Department of Automation and Systems Technology
Information and Computer Systems in Automation

Supervising professor
Prof. Kari Koskinen

Thesis advisor
Dr. Seppo Sierla

Preliminary examiners
Prof. Georg Frey, Saarland University, Germany
Dr. Jari Hämäläinen, Valtion Teknillinen Tutkimuskeskus (VTT), Finland

Opponent
Prof. Birgit Vogel-Heuser, Technische Universität München (TUM), Germany

Aalto University publication series
DOCTORAL DISSERTATIONS 148/2012

ISBN 978-952-60-4861-1 (printed)
ISBN 978-952-60-4862-8 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-60-4862-8

Unigrafia Oy
Helsinki 2012

Finland

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Nikolaos Papakonstantinou

Name of the doctoral dissertation
System design and risk assessment for safety critical control software product lines

Publisher School of Electrical Engineering

Unit Department of Automation and Systems Technology

Series Aalto University publication series DOCTORAL DISSERTATIONS 148/2012

Field of research Information and Computer Systems in Automation

Manuscript submitted 13 June 2012 **Date of the defence** 23 November 2012

Permission to publish granted (date) 16 October 2012 **Language** English

☐ **Monograph** [☒ **Article dissertation (summary + original articles)**]

Abstract

A methodology is presented for the design of safety critical product lines for control automation software. The functional failure identification and propagation risk assessment method is used in the early design phase of the mechatronic system. The applied methodology starts with the decomposition of the system into functions that are connected by energy, material and signal flows. This results in a functional model that does not make any assumptions on what components are used to realize the functions. The functions are mapped to mechatronic components in a model that can be simulated: the configuration flow graph. Functional failure logic is executed in parallel to the simulation to monitor the simulation signals and to determine the health of each function. The functional health results of the simulation, when critical events are injected, are used to identify the propagation of functional failures. Alternative designs that are described with a feature model, combinations of component parameter values and changes in the critical event scenario can be simulated. System designs that result in undesirable behavior are rejected. The purpose is to identify risks and to determine mechatronic designs with adequate safety characteristics before the design process branches into software, electrical and mechanical domains. The final deliverable of the mechatronic system design phase is a feature model capturing the design alternatives with acceptable safety characteristics. The aspect of this model containing software is the starting point for software product line engineering. In control automation, programmable logic controller targets are used, so a methodology and toolchain for supporting software product line configuration for such platforms has been developed using the PLCopen standard. Two case studies are used to demonstrate the methodology: a boiling water reactor, with a focus on reactor coolant pumps, and a mobile elevating work platform.

Keywords Risk assessment, safety, fault propagation, functional modeling, design alternative, software product line, control software, PLCopen, IEC 61131-3

ISBN (printed) 978-952-60-4861-1 **ISBN (pdf)** 978-952-60-4862-8

ISSN-L 1799-4934 **ISSN (printed)** 1799-4934 **ISSN (pdf)** 1799-4942

Location of publisher Espoo **Location of printing** Helsinki **Year** 2012

Pages 154 **urn** http://urn.fi/URN:ISBN:978-952-60-4862-8

Preface

As engineers we are trained to provide practical solutions, within constraints, to the community. This demanding task can be achieved only though a series of careful compromises. While these compromises are a necessity, some of them can eventually become social norms, and be later treated as preconditions. Many constraints change over time, and thus trade-offs that were made in the past, may now be different or may not even be necessary. In the past four years, while working as a research scientist, many times I had to strongly defend my choice to reevaluate some compromises that had evolved into habits. At this time, I feel that it was worth the effort.

Espoo, 1st of Nov. 2012

Nikolaos Papakonstantinou

Contents

List of publications

Publication I: Nikolaos Papakonstantinou, Seppo Sierla, Jarmo Alanen, Kari Koskinen: Reducing Redesign of Safety Critical Control Systems by Early Risk Assessment, IEEE INDustrial INformatics (INDIN) conference, July 13-16, 2010, Osaka, Japan. 6 pages.

Publication II: Nikolaos Papakonstantinou, Seppo Sierla, David Jensen, Irem Tumer: Capturing interactions and emergent failure behavior in complex engineered systems and multiple scales, ASME 2011 International Design Engineering Technical Conferences (IDETC) and Computers and Information in Engineering Conference (CIE), August 28-31, 2011, Washington, DC, USA. 10 pages.

Publication III: Nikolaos Papakonstantinou, Seppo Sierla, Kari Koskinen: Object oriented extensions of IEC 61131-3 as an enabling technology of software product lines, IEEE Emerging Technologies in Factory Automation (ETFA) conference, September 5-9, 2011, Toulouse, France. 8 pages.

Publication IV: Nikolaos Papakonstantinou, Seppo Sierla, Kari Koskinen: Generating and validating product instances in IEC 61131-3 from feature models, IEEE Emerging Technologies in Factory Automation (ETFA) conference, September 5-9, 2011, Toulouse, France. 8 pages.

Publication V: Seppo Sierla, Irem Tumer, Nikolaos Papakonstantinou, Kari Koskinen, David Jensen: Early integration of safety to the mechatronic system design process by the functional failure identification and propagation framework, MECHATRONICS international journal, 2012. 22(2): P. 137-151, doi:10.1016/j.mechatronics.2012.01.003. 26 pages (before final layout).

Publication VI: Nikolaos Papakonstantinou, Seppo Sierla: Early Phase Fault Propagation Analysis of Safety Critical Factory Automation Systems, IEEE INDustrial INformatics (INDIN) conference, July 25-27, 2012, Beijing, China. 6 pages.

Publication VII: Nikolaos Papakonstantinou, Seppo Sierla, Irem Tumer, David Jensen: Using fault propagation analyses for early elimination of unreliable design alternatives of complex cyber-physical systems, ASME 2012 International Design Engineering Technical Conferences (IDETC) and Computers and Information in Engineering Conference (CIE), August 12-15, 2012, Chicago, IL, USA. 9 pages.

Author's contribution

All the publications listed in this section are the result of teamwork of their authors (see List of publications section). The contribution of the author of this dissertation to the articles can be summarized as follows:

IEEE INDIN 2010 (**Publication I**): The author was responsible for part of the literature review (Section II), for the research and development of the proposed metric (Section III), development of the software tool that implements the metric and generation, presentation and interpretation of the results (Section V).

ASME IDETC/CIE 2011 (**Publication II**): The author was responsible for the presentation of the case study, the development of the simulation model for the case study and for the generation, presentation and interpretation of the results (Section "CASE STUDY").

IEEE ETFA 2011 (**Publication III**): The author was responsible for part of the literature review (Section 2), for the development of the methodology (Section 3), development of the case study in CoDeSys (Section 4), development of the software tool which is part of the proposed toolchain and for the generation, presentation and interpretation of the results (Section 5).

IEEE ETFA 2011 (**Publication IV**): The author was responsible for part of the literature review (Section 2), for the development of the case study in CoDeSys (Section 3), development of the software tools which are part of the proposed toolchain and for the generation, presentation and interpretation of the results (Section 4).

Elsevier Mechatronics journal (**Publication V**): The author was responsible for the development of the case study and its simulation model in Simulink (Section 3), and for the generation, presentation and interpretation of the results (Section 4).

IEEE INDIN 2012 (**Publication VI**): The author was responsible for the development of the case study in Simulink (Section III), extension of open source software tools to support the proposed environment and toolchain (Section IV) and for the generation, presentation and interpretation of the results (Section V).

ASME IDETC/CIE 2012 (**Publication VII**): The author was responsible for the development of the case study in Simulink, extension of open source software tools to support the proposed environment and toolchain and for the generation, presentation and interpretation of the results. (Sections "CASE STUDY", "METHODOLOGY, SIMULATION ENVIRONMENT AND TOOLCHAIN" and "RESULTS").

Summary of publications

Publications I-VII are part of this dissertation. For complete references, see the List of publications section.

Publication I proposes a metric which quantifies the amount of system design rework done, after a change request originating from risk assessment, during different phases of the system design. The results of the metric, when applied to a case study, showed that the amount of rework for a change request rises significantly the later the risk assessment is performed. The publication motivates development of early phase risk assessment methods, which are the subject of publications II, V, VI and VII.

Publication V present the extension of the Functional Failure Identification and Propagation framework to support early designs of complex systems which contain feedback loops. Previous work on the method had been in mechanical design, and this publication extends the method into the mechatronic domain by incorporating machine control software. Since the results are dependent on how the model is parameterized based on the concept phase knowledge of the system, the response of the system to design parameter changes was investigated **Publication II**.

Publication VI and **Publication VII** present the extension of the Functional Failure Identification and Propagation framework towards supporting the assessment of alternative system designs. A methodology is introduced to describe the alternative designs using the feature model notation and to use the Functional Failure Identification and Propagation method to identify and then remove the unsafe designs from the product line.

Publication III presents a methodology that applies the Software Product Line paradigm to control software design. The Object Oriented extensions of IEC 61131 and the PLCopen XML file format for storing control applications are used as enabling technologies to support the implementation of feature modeling concepts. **Publication IV** presents a methodology and a supporting toolchain for adding feature model constrains into the PLCopen XML representation of the control application product line and then for configuring valid product configurations of the IEC 61131 control application.

List of abbreviations

AADL - Architecture Analysis and Design Language
BWR - Boiling Water Reactor
CFG - Configuration Flow Graph
DECOS - Dependable Embedded Components and Systems
EMS - Energy Material Signal
FFA - Functional Failure Analysis
FFIP - Functional Failure Identification and Propagation
FFL - Functional Failure Logic
FMEA - Failure Modes and Effects Analysis
FMECA - Failure Mode Effect and Criticality Analysis
FTA - The Fault Tree Analysis
HAZOP - HAZard and OPerability study
IDE - Integrated Development Environment
IEC - International Electrotechnical Commission
IT - Information Technology
LON - Local Operating Network
MDE - Model Driven Engineering
MEWP - Mobile Elevating Work Platform
OCL - Object Constraint Language
OO - Object Oriented
PCV - Pressure Control Valve
PLC - Programmable Logic Controller
POU - Program Organization Unit
PRA - Probabilistic Risk Assessment
RPM - Rotations Per Minute
SFMECA - Software Failure Mode, Effects and Criticality Analysis
SFTA - Software Fault Tree Analysis
SIL - Safety Integrity Level
SPL - Software Product Line
ST - Structured Text
TPS - Turbine Protection System
UML - Unified Modeling Language
XML - Extensible Markup Language

Introduction

1 Motivation

Safety critical machine automation is based increasingly on software. Safety, cost efficiency and product flexibility are key factors to the development of safety critical automation systems. The paradigm of Software Product Line (SPL) [1], which originates from the software engineering domain, can facilitate the design of safety critical machine automation if it is focused towards that goal. In this dissertation, the following definition for SPL is used: "a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way" [2]. There is a lack of research for a methodology to identify product line members of safety critical SPLs. Additionally, performing risk analysis on early designs can lead to less redesign and thus cost reduction [3]. Since safety is a property of the mechatronic system, it is necessary to investigate the safety critical product line from a mechatronic perspective, in order to identify potentially unsafe configurations before isolating the software aspect of the product line.

The feature modeling technique is used to describe mandatory, optional and alternative features in a product line, so that product instances are obtained by selecting among these optional and alternative features [4]. In this dissertation, the following definition for feature is used: "property of a domain concept, which is relevant to some domain stakeholder and is used to discriminate between concept instances" [4]. In this definition, a domain stakeholder is anyone involved in the SPL. Additionally, constraints may be used to prevent the selection of incompatible feature combinations. Further research is needed to develop methodologies for obtaining feature models that satisfy non-functional quality requirements; in this dissertation, the focus is on safety. Safety of machine control software cannot be evaluated independently of the physical equipment to be controlled [5], so feature modeling is applied to mechatronic designs. This model is presented as source information for risk analysis, which is used to identify unsafe combinations of features. These combinations are removed from the model either by removing features or adding constraints. After removing features that do not involve software, the resulting feature model is the starting point for software development by established SPL techniques.

The SPL paradigm was developed to generate software implementations that follow the Object Oriented software development paradigm. Recent advances to control automation development platforms attempt to add object oriented extensions to IEC 61131 [6], the industry standard for programming Programmable Logic Controllers (PLC). Efforts have also been made towards interoperability between automation software development platforms by introducing a standardized file format for describing automation projects (PLCopen Extensible Markup Language file format). In this research, the object oriented extensions and the PLCopen Extensible Markup Language (XML) file format are used in the SPL toolchain to produce vendor independent PLC software implementations, after product instances have been configured from the feature model.

A system development process is presented, starting with the development of the feature model that describes mechatronic design alternatives. A methodology for risk assessment of a mechatronic feature model is presented, aiming at restricting the model to exclude unsafe feature combinations. Even if safety of individual features has been verified, specific combinations of features may introduce risks, so a risk assessment method for studying interaction of mechatronic components is needed. This dissertation uses the FFIP (Functional Failure Identification and Propagation) framework, a methodology for tracking fault propagation over boundaries of subsystems and automation, electrical and mechanical domains [7]. A toolset is presented for performing this risk assessment and for supporting the configuration of vendor independent PLC software implementations.

This research is motivated by the lack of methods for obtaining a product line specification for a safety critical product line. A risk assessment method for addressing complex, software intensive mechatronic systems containing variability has not yet been proposed. While software engineering methods and tools for SPL development are well established, this technology has not yet been applied to machine control software development based on the PLC. This is not only an engineering problem but also a scientifically significant research problem, since the PLC introduces a fundamentally different programming and software execution paradigm.

2 Literature review

2.1 Risk analysis

Several researchers in the safety field have discussed the advantages of applying a certain risk assessment method or a set of such methods [3]. One extreme view is that only one method should be used to avoid comparability issues within a set of requirements derived by different methods; the authors recommend Enhanced Markov Analysis, a difficult and time consuming method, due to its great expressive power [8]. Other authors justify the need for several methods in order to adequately address human and organizational factors [9], to cope with situations when adequate failure data is not available [10] and to apply more work intensive methods only to the most critical parts of the system [11], [12]. A more rarely addressed criteria in the choice of risk assessment methods is the capability to identify system failures based on conceptual designs before costly design commitments are made; Kurtoglu and Tumer [7] propose the functional-failure identification and propagation (FFIP) framework for this purpose.

A variety of established approaches for performing risk analysis are available. The effects on the system when a single component enters a failure mode are examined using the Failure Modes and Effects Analysis (FMEA) [13] method. The FMEA method is used widely in safety critical industries but is still heavily based on past experience and is very laborious for complex systems [14] . The Failure Mode Effect and Criticality Analysis (FMECA) [15] is an extension of FMEA that evaluates the probability of a component entering a failure mode against the severity of the consequences to the system. Research has been made for extending FMEA to support a combination of component failures, but when the source material is detailed designs of complex systems the number of combinations is not manageable [16]. FMEA can be combined with functional models and that leads to better applicability to complex systems [17]. The Functional Failure Analysis (FFA) is a method that follows the workflow of the FMEA but operates solely on the functional model of the system [18]. The HAZard and OPerability (HAZOP) method is qualitative and uses the functional models. A set of keywords is combined with the system functions and, using past experience, potential hazards are identified [19]. Attempts have been made to reduce the manual effort of performing HAZOP by using technologies like knowledge databases [20]. The Fault Tree Analysis (FTA) method is a top – down approach, where an undesirable system state is analyzed to a combination of component failure modes [21]. The Probabilistic Risk Assessment (PRA) [22] is a method that provides quantitative results and targets to reveal potential hazards in complex system designs; the detailed designs along with reliability information for every component are required, so its applicability is only to the last stages of the design process. The aforementioned risk analysis methods require extensive expertise of the behavior and the interactions within the complex system and are geared towards the latter phases of the system development. They are time-consuming and usually are applied for verification of the safety requirements of a refined design.

The risk analysis methods that are performed manually require continuous update to keep their source information consistent with the evolution of the system design. This process causes further delays and requires considerable effort [23]. Research attempts are being made to address this weakness by proposing risk and reliability analysis methods that are integrated with the simulation of the system model. By specifying a set of detailed input parameters which are then used for simulating the model of the system, designers can evaluate the effects of these parameters to the system’s behavior. Common analysis methods include Monte Carlo simulation modeling [24], response surface models [25] and meta-modeling techniques such as in [26]. When these methods are reliability oriented [27], the probability of a system response is estimated, given specific probability distributions. The minimization of the variation of the response of the system is not considered by reliability-based methods. Robust design optimization oriented methods [28] target the minimization of the variation of the system behavior. A robust optimal design method that combines Monte Carlo Simulation and Stochastic Petri Nets is proposed in [29]. These methods are applicable to models of systems that are detailed enough to include specific component reliability information and operation tolerances, in early system designs such information is not yet available.

The effects of faults in a system and the assessment of risk factors can be analyzed by simulation-based techniques. Methods like directed graphs and Multi-Signal Flow Graphs [30] can reveal the fault propagation paths within a system. The behavior of the system components can be modeled in a qualitative level and failure propagation and loss of system function health can be reasoned based on system-level abstractions [7, 31].

The goal of the risk assessment method presented in this dissertation is to help the system designer make safety related decisions during the early design of a complex system, when the knowledge about the system is still qualitative and incomplete. This risk assessment technique utilizes the simulation model of the system which supports different abstraction levels for the system's components.

2.2 Safety and risk analysis in the industrial informatics field

Since this work is addressing safety critical automation software, a review of safety related research in the industrial informatics community is presented [3]. Most publications in this field that make some claims to safety do not clearly define safety, nor do they seem to follow definitions presented in standards. The broadly accepted safety standard IEC 61508 is often cited by those publications that operate on a clear definition of safety. This work uses the definition of safety provided in IEC 61508 part 4: "freedom from unacceptable risk" [32]. The definition implies that risk assessment must be performed before the level of safety requirements can be known.

Much of the safety related research is not concerned with risk assessment, but assumes that a certain level of safety requirements is to be satisfied. This level is usually specified in terms of SIL (Safety Integrity Level), a metric which is defined by IEC safety standards [32, 33]. Restrictions to the Programmable Logic Controller (PLC) programming standard IEC 61131-3 are proposed in [34] in order to make these languages appropriate for the development of applications with high SIL requirements. Extensions to the SIL measure, as defined in IEC 61508, are proposed in [35] in order to account for failures due to the real-time properties of the system. A categorization of communication errors and relevant safety measures for several Ethernet based protocols that are rated for SIL 3 is provided in [35]. A proposal for improving the ISOBUS protocol as to obtain transmission error rates that are within the limits of SIL 3 has been devised in [36]. An overview of safety functions for drives and examples of their technical realization are presented in [37], but it is not in the scope of our research to determine if the application at hand actually needs the investment of the presented safety functions or if the safety requirements of a high integrity application are fully met by these functions. [38] describes a set of self-tests for inexpensive microcontrollers to decrease the proportion of dangerous undetected failures to satisfy the requirements of SIL 3. [39] identifies possible failure sources in the Local Operating Network (LON) protocol stack (e.g. data corruption; loss of messages; manipulation of messages) and [40] proposes extensions to the communication frame that have been certified to reduce the rate of these failures to satisfy SIL 3. While each of these papers makes a relevant contribution to the safety of industrial systems, our goal of reducing and quantifying redesign by early risk assessment is not in the scope of these papers. These research problems are encountered in a late phase of the development cycle, whereas this dissertation targets the early phases.

In [41], a risk analysis method is proposed to prioritize the use of limited resources for the maintenance of a system that is not safety critical; the approach is interesting and practical but does not conform to the rigorous expectations of IEC 61508 or other safety standards.

A smaller number of publications provide a thorough treatment of risk assessment. A partially automated, rule-based method for hazard identification based on HAZOP is described in [20], but the possibility of obtaining requirements in terms of SIL with this approach is not discussed. The Dependable Embedded Components and Systems (DECOS) project has proposed a comprehensive methodology for the systems development and safety process, with a test bench infrastructure that supports certification of systems or their parts against specific SIL requirements [5], [42]. [43] and [44] introduce a life cycle model that merges the models of safety (IEC 61508) and security (IEC 15408) standards; the focus of the papers is to resolve conflicting requirements that may result from these models that are traditionally carried out separately. While each of these papers makes a relevant contribution to the risk assessment of industrial systems, they do not address the problem of assessing complex mechatronic systems, for which traditional methods such as FMEA, FTA and HAZOP are inadequate. In order to fill this gap, a simulation-based risk assessment methodology for such systems has been proposed to this research community as a part of this dissertation [45].

2.3 Comparison of risk assessment and verification methods

This dissertation is concerned with risk assessment rather than verification methodology. Verification requires detailed system models, which are checked against safety requirements; risk assessment is a core activity in obtaining these requirements. Verification is applicable in later phases of the development process, such as hardware-in-the-loop simulation [46], or when the system design process has split into domain specific processes such as software development, in which the task is to determine formal requirements and to ensure the software implementations conformance to those requirements [47]. Much research on verification methods has been restricted to control software that is described by sufficiently formal models, such as SCADE [48], while our goal is to track fault propagation paths that cross the boundaries of software, electrical and mechanical systems. Recently, verification methods have been developed to analyze a model consisting of control software and a physical process. An approach combining Petri nets with differential equations systems to verify systems containing continuous and discrete dynamics [49]. Another approach combining theorem proving with non-linear optimization techniques is described in [50].

In [49], the verification of the software with Petri nets is done against a differential equation system describing the nominal characteristics of the physical equipment under control. In simulation-based risk assessment, the system-wide effect of component failures can be studied. In the FFIP framework used in this dissertation [51], it is possible to inject critical events to drive any component to one of several possible failure modes. This component's simulation switches to using set of difference equations capturing the faulty behavior, while the simulation continues and the abnormal flow levels caused by the failed component propagate to other parts of the system. In risk assessment, our interest is not to discover any deviation from specifications, but to determine if these abnormal levels impact the system's ability to carry out its safety related functions in the face of several component failures. The verification with Petri nets performs static analyses of the system model to make sure that software will not reach a state that is foreseen to be unsafe. FFIP simulates the software together with the equipment under control to determine the robustness of the mechatronic design in equipment failure scenarios.

2.4 Software product lines and feature modeling

SPL is an established technology in the Information Technology (IT) domain for providing a broad range of customer options without the cost and delays involved in developing code separately for individual products [1]. SPL is considered a key technology for producing software products when time to market and software reuse are critical factors for the success of a product (e.g. [1, 52]), and research is already focusing on the problem of migrating legacy software to a SPL [53, 54]. This section provides an overview of available SPL technology that mainly relies on Unified Modeling Language (UML) and object oriented technologies [55], while section 2.5 examines research that could be used to bridge the gap between the research in this section and the PLC (Programmable Logic Controller) programming languages used in industrial automation [56]. Fig. 1 presents the main activities needed to create and exploit a SPL; the concept is intentionally general and abstract, since it only serves to position the numerous papers in this section under a common context.

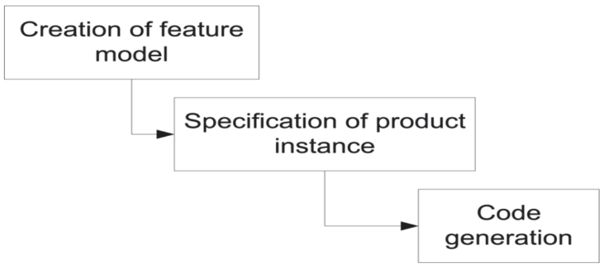


Figure 1, Main activities in creating and exploiting a SPL, adapted from [55]

Feature modeling is a key technique that supports the first phase in Fig. 1. Feature models describe mandatory, optional and alternative features that are supported by a software product line (SPL), so that customer-specific product instances may be derived by selecting among optional and alternative features [57]. A feature may also require or exclude the presence of another feature, so such rules need to be defined at the product line level and validated at the product instance level. Usually it relates only to software, but sometimes also to other system parts such as sensors [58]. The starting point for this research is that a feature model exists for describing product features that are considered desirable by market experts. Features may also describe technical implementation alternatives that are to be evaluated from a safety perspective.

The development of a SPL starts with the creation of a feature model based on the domain requirements, which specifies the mandatory, optional and alternative status of features and thus determines which combinations of features will be considered a valid product instance [59-61]. Further, the feature model may include rules specifying which features require or exclude another feature, sometimes using Object Constraint Language (OCL) [62]. Based on requirements derived from a market analysis, the feature oriented approach to creating a SPL starts with identifying the features and then placing them into a hierarchy, which corresponds to a top-down architecture of the system [63]. An alternative approach first builds a domain requirements model focusing on scenarios and goals as an intermediate step before creating the feature model [64]. The feature model is usually expressed either as some variation of the notation defined by [4, 59] as in [65] or UML [66, 67].

The second phase in Fig. 1 involves product configuration through selection of features to be included in a customer specific product instance. The visual modeling notations proposed to support this phase are adapted either from UML [68] or the feature modeling notation of [4] as in [69, 70]. An important functionality for any tool for this purpose is the ability to check the validity of the chosen configuration against the rules and features of the SPL level feature model [71]. The Rhizome approach defines modeling languages based on XML, but the user interaction is based on typing commands to a console [72]. Commercial tools to support this phase enable the user to configure the product instance based on textual list and tree views [73, 74].

After the product instance has been configured, it is necessary to generate source code that is ready to be compiled to a finished software product (the last phase in Fig. 1). The FeatureIDE eclipse plugin [69], apart from feature modeling and product configuration, supports Java code generation. The operating principle behind Rhizome is to disable or enable parts of code from templates to get the source code of the product [72]. An approach for automatic code generation with the aspect oriented paradigm is presented in [75], but it involves manual edits to the build file. In order to support targets used in industrial automation, the resulting source code should be in a programming language such as the IEC 61131, which can be used by a PLC integrated development platform.

2.5 Safety critical software product lines and mechatronic product line engineering

The risk analysis methods presented in sections 2.1 and 2.2 are designed to be applied on a single system design. The introduction of SPL as a system design paradigm that enables flexibility and cost effectiveness requires a risk analysis method that can be adapted to support the product family design. For safety critical systems, it is necessary for such a risk analysis method to address all the aspects of the mechatronic system and not to be restricted only to software.

For safety critical applications, SPL remains an academic research problem without widespread industrial adoption; this is a serious problem considering the sharp growth of software intensiveness in safety critical domains such as transportation, manufacturing, aerospace and nuclear power. The majority of research has a unidisciplinary focus on safety issues arising from the software itself, such as ensuring that software features do not interact to bring the system to an unsafe state [76, 77] or exhaustive verification of product instances for certain required properties [78].The safety criticality of software is due to its controlling of electronic and mechanical devices, which are able to cause harm. A fundamental discovery in the field of mechatronics has been that no sophistication on the part of the software can overcome poor electromechanical designs [79], and this has resulted in a recognized need for concurrent design of software, electronic and mechanical aspects of a system [80]. A recent advance toward a mechatronic direction has been the integration of AADL (Architecture Analysis and Design Language) to fault tree analysis (FTA) [81], but this is still limited to the software and execution hardware. The need for verification of software requirements against system safety requirements is recognized and addressed by applying safety analysis methods such as Software Fault Tree Analysis (SFTA) and Software Failure Mode, Effects and Criticality Analysis (SFMECA) on the SPL [82, 83], but this avoids the more fundamental question of how to perform risk analysis of the product line at the system level in order to decide what functionality is implemented in software, what are the potentially hazardous interactions between this software and the rest of the system, and how this knowledge is used to define constraints on valid combinations of features in the SPL.

2.6 Software Product Lines for industrial control software applications

Based on this literature review of the state-of-the-art in SPL, which heavily exploits object oriented technologies, it is possible to investigate the application of SPL to object oriented function block based machine control software [84]. Model driven SPL approaches for object-oriented targets are emerging [69], but fundamental changes would be required in order to support PLC targets. In the industrial software domain, an approach towards industrial product line software development based on the IEC 61499 programming standard is presented in [85], although actual generation of executable IEC 61499 applications is left for further work. However, IEC 61499 follows only partially the object oriented paradigm and has not gained widespread industrial acceptance [86]. The IEC 61131-3 programming standard has been widely accepted in industry [56] and recently supports object-orientation [6], but the SPL approach in this context has not yet been researched and no SPL tools to support IEC 61131-3 application generation are available. While both standards could feasibly support SPL for industrial control, IEC 61131-3 is chosen for the above mentioned reasons. The goal of this research is to support SPL of machine control software with IEC 61131-3.

Many years before the introduction of object-oriented extensions to IEC 61131-3, researchers started to investigate the benefits of object orientation in the context of industrial control software. A UML model of the software is used to support formal design and verification techniques in [87]. An object oriented tool for programming control applications generating IEC 61131 code (without the object oriented extensions) is presented in [88]. Several researchers have proposed guidelines for implementing object oriented designs with one of the IEC 61131-3 languages [89] [90] [91].

Object oriented extensions of IEC 61131-3 are presented in [6] and the mapping of UML class diagrams to the extended function blocks is described in [92], which illustrates some of the benefits of the new modeling constructs. In this research, the inheritance mechanism is exploited to obtain a mapping from feature models to IEC 61131-3 applications. The vendor independent PLCopen XML standard for exchanging IEC 61131-3 applications supports open toolchains in several recent initiatives based on the AutomationML [93]; in this paper, PLCopen XML is used to store the SPL and product instance level models as well as the source code. Since the language is based on a schema, it has the same expressive power as UML metamodels, and is thus a possible technology for supporting the chain illustrated in fig. 1.

The PLCopen XML schema does not yet cover object-oriented constructs, but a part of the schema (the <addData> element) enables extensions, which the CoDeSys tool has exploited to describe the new object-oriented features of IEC 61131-3. In our research the <addData> element [55] is used to support feature modeling constructs. This is also a proposal for standardization to the PLCopen community, which will become topical if other feature modeling approaches are proposed.

3 Objectives

Risk analysis methods typically expect a single design as source information, and do not support analysis of alternative designs or optional features. It is here posited that in order to provide starting information for simulation-based risk assessment of a mechatronic product line, feature modeling is used to describe a mechatronic product line; this model is subjected to risk analysis, so that unsafe feature combinations are eliminated. The resulting feature model can then be handed to software developers, and the software design and implementation, even in the face of evolution and maintenance pressures, can be addressed with the existing body of research in SPL as long as targets used in industrial automation are supported.

In Fig. 2 the methodology for system design and risk assessment for safety critical control software product lines is drawn in steps. The first step is the design of the functional model of the system, which captures the desired functionality without making assumptions on how it is implemented. Then different system design alternatives are specified as a feature model (step 2). Simulation-based risk assessment is applied systematically to every valid configuration of the feature model and unsafe design alternatives are eliminated, resulting in a feature model describing the range of possible products that are considered safe (step 3). In step 4, the features that do not involve software are removed from this model, after which it is possible to apply existing SPL technology to develop it further. In this work, a tool able to configure applications for PLC targets has been developed (step 5). This methodology is presented in this dissertation using two safety critical case studies: a boiling water reactor's recirculation pumps and a mobile elevating work platform.

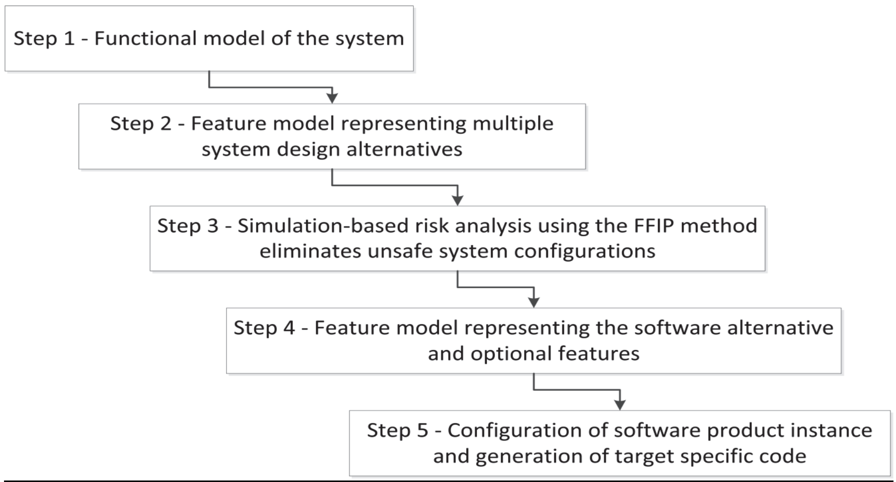

Step 1 - Functional model of the system
Step 2 - Feature model representing multiple system design alternatives
Step 3 - Simulation-based risk analysis using the FFIP method eliminates unsafe system configurations
Step 4 - Feature model representing the software alternative and optional features
Step 5 - Configuration of software product instance and generation of target specific code

Figure 2, Workflow overview for system design and risk assessment for safety critical control software product lines

The objectives for the methodology presented in this dissertation are:

1. The modeling approach should support filtering designs based on results of risk assessment

2. The behavioral simulation in previous work on FFIP does not support any kind of study of several positive or negative feedback loops affecting the same process variable. The discrete qualitative enumeration for flow values [zero, low, nominal, high] is insufficient to capture, even in a qualitative way, how several feedback mechanisms reinforce or oppose each other. An objective of this work is to propose a continuous range of values to qualitatively describe flow values, so that simulation models are able to capture how several feedback mechanisms reinforce or oppose each other.

3. The modeling approach used in behavioral simulation should support comparative evaluation of the safety of alternative designs or different parameter values for the same design. As the number of alternatives increases, the scalability of the method should be evaluated.

4 Case studies

4.1 Mobile elevating work platform product family

The Mobile Elevating Work Platform (MEWP) product family is one of the two case studies of this dissertation (see Fig. 3). It will be used in section 5.1 as an example for the introduction of the feature modeling methodology and section 6.4 for demonstrating the integration of feature modeling and IEC 61131 PLC software.

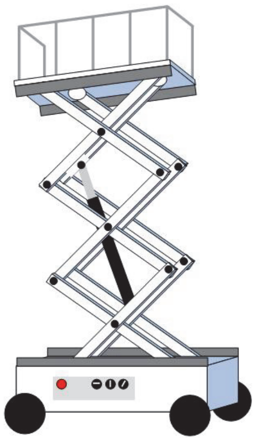


Figure 3, Mobile Elevating Work Platform system concept [3]

The basic function of a MEWP is to safely move the person(s) on its platform to a certain height in order to perform some work required there. The requirements for the product family of the MEWP state that all the machines should support manual vertical movement. Optionally, automatic movement could be available as an extra feature. The automatic movement, if present on a machine, requires additional user interface controls: two quick height buttons. If a quick height button is pushed longer than a predefined time, the current platform height is stored in the memory of the control application. If that button is pushed for less than the predefined time, the platform moves automatically to the previously stored height. Two buttons make it possible to store two different positions in memory.

A specific product in this product family may either have binary or proportional valves to control the movement cylinder (see the partial hydraulics of the MEWP in Fig. 4). Correspondingly, the software will include either a binary (on-off) or analog control algorithm. With binary control, the automatic mode drives the platform at full speed to the stored position. With analog control, the automatic mode can support two additional safer variations: “Slow auto” and “Smooth auto”, in which the platform moves at a reduced speed or with restricted acceleration, respectively. There are two variations for the user input device: a joystick is used with analog control and buttons are provided for binary control. In the software, there may be a cylinder simulator for testing purposes.

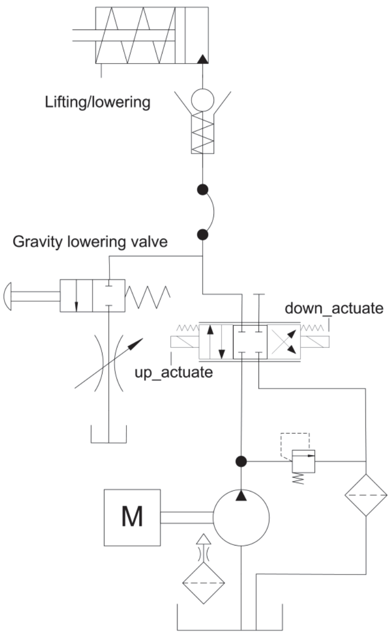


Figure 4, Mobile Elevating Work Platform partial hydraulics diagram [3]

4.2 Boiling Water Reactor

The second case study of this dissertation is a Boiling Water Reactor (BWR) in its early design phase. The scope of the case study is the reactor core, the coolant circulation and the main steam outlets. The main principle of operation of a BWR is the generation of thermal energy due to nuclear fission (see the BWR concept P&I diagram in Fig 5). Within the reactor vessel are fuel rods that are packed as fuel assemblies, which are arranged by reactor physicists to form the reactor core. The spaces between the fuel rods within the fuel assemblies are called fuel channels. Under normal operation, the fuel assemblies are submerged under water, which acts as both coolant and moderator. The circulation of coolant is upward through the fuel channels and then downward along the sides of the reactor vessel.

In a boiling water reactor, the thermal power output is directly proportional to the flow rate through the reactor coolant pumps. The presence of steam in the fuel channels has a negative impact on the number of thermal neutrons that sustain the nuclear chain reaction. The increased circulation of water through the fuel channels reduces the void fraction, which is defined as the proportion of steam in the coolant. This lower void fraction leads to more effective moderation of neutrons and increased power production. Thus in power control mode, the coolant pumps receive their setpoint from the power control subsystem. In emergency situations, the pumps have the dual task of maintaining sufficient circulation to prevent steam buildup at the core while at the same time reducing the thermal power output of the reactor. These two tasks are accomplished by driving the rotations per minute of the pumps down by a ramp.

High pressure steam is produced in the reactor vessel, and the reactor pressure is controlled by the pressure control subsystem by using a pressure control valve in the pipeline joining the reactor vessel and the turbine. The pipeline also has a quick close valve that is closed immediately if for any reason the turbine is not ready to accept steam. In such situations, the steam from the reactor is dumped directly into the condenser through pipes that bypass the turbine; this pipeline contains a dumper valve that is controlled by the turbine protection subsystem.

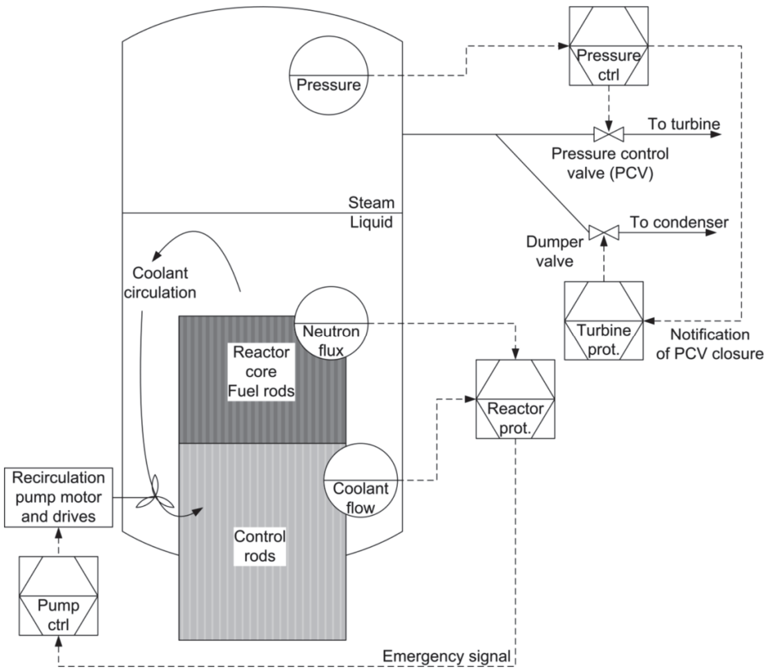


Figure 5, P&I concept diagram of a boiling water reactor, its steam outlets and related control and protection systems [51]

The energy production in the reactor core is affected by several feedback loops. An increased flow of coolant through the fuel channels decreases the void fraction and thus increases the neutron flux and thermal power output. The neutron flux is a measure of neutrons in the thermic range capable of causing fission reactions [94]. An increase of pressure has a similar effect, as it compresses steam bubbles and thus decreases the void fraction. Due to the Doppler effect [95], an increase in fuel rod temperature decreases the neutron flux.

An emergency shutdown of the reactor can be initiated by the reactor protection system. This shutdown process, which is referred to as SCRAM, can be triggered by a number of initiating events such as increased neutron flux in the reactor core, low coolant flow rate and loss of external power (from the national grid). In case of a SCRAM, the coolant pumps must go into a safe mode of operation and ramp down the coolant flow in order to reduce the power output of the plant while maintaining minimum coolant circulation needed to reduce the void fraction. The control rods are also inserted into the core to shut down the nuclear reaction, but without the timely action of the coolant pumps, fuel rod damage may occur before the control rods are in place. In order to ensure the operation of the coolant pumps in the case of voltage sags or transients in the power rail supplying these pumps, local emergency power supplies are available.

A known potential hazard that a BWR design must be able to handle is related to the control of the pressure in the reactor vessel. If the pressure control subsystem that controls the pressure control valve closes the steam path to the turbine, then the turbine protection system is notified and opens the dumper valve that releases steam to the condenser. A hazard emerges if the pressure control subsystem closes the pressure control valve due to a software malfunction. In that case the turbine protection system is not notified, the dumper valve does not open and a pressure shockwave propagates from the pressure control valve back to the reactor vessel, resulting in compression of steam bubbles, decreased void fraction and increased neutron flux over the SCRAM threshold. The coolant pumps play a key role into maintaining this process under control.

Design alternatives at this early concept phase of this case study can be assessed (see section 6.2 and 6.3) for their behaviour against the afore-mentioned hazard. In Fig 6 two alternative designs are presented to the basic concept of Fig 5. The most basic configuration is that the turbine protection system does not receive a pressure measurement, so that it is up to the reactor protection system to identify overpressure and trigger a SCRAM. One alternative is to connect the measurement from the pressure sensor inside the reactor vessel directly to the turbine protection system. Another alternative for slightly earlier detection is to introduce an additional pressure sensor into the pipeline before the pressure control valve, and to connect it to the turbine protection system. The purpose of providing this measurement to the turbine protection system, in the context of pressure shockwave hazard scenario presented earlier, is to identify an emerging pressure shockwave and to open the dumper valve to relieve the pressure.

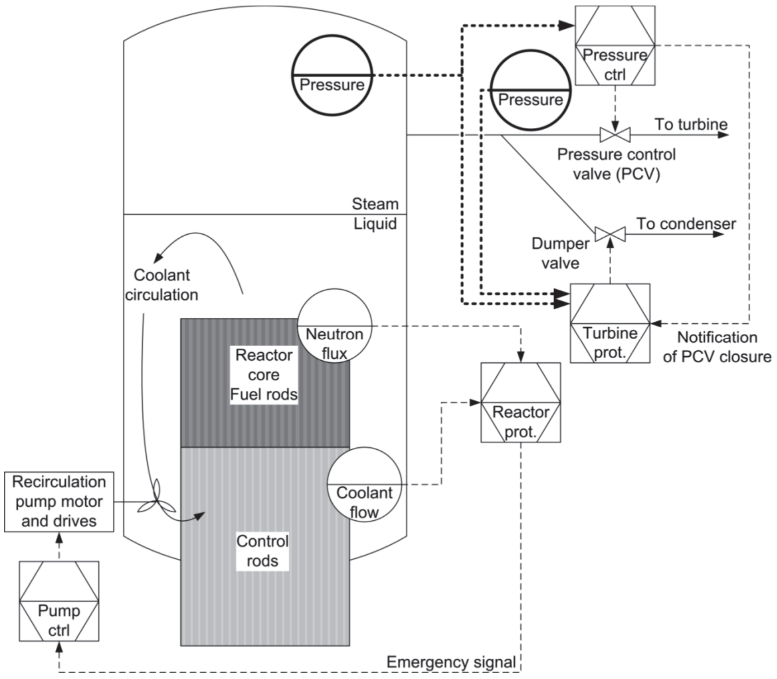


Figure 6, Boiling Water Reactor concept design after the introduction of optional pressure sensor information (in bold), either before the pressure control valve or in the reactor vessel [45].

5 Methods

5.1 Feature modeling

Feature models provide the necessary modeling notation, graphically [69] or textually [61], to present in a tree hierarchy the alternative and optional features. In this dissertation, feature modeling is used both in systems engineering and later in software development. A feature is defined as a "property of a domain concept, which is relevant to some domain stakeholder and is used to discriminate between concept instances" [4]. Thus, features can be used to express design alternatives submitted to risk assessment as well as requirements to a software product line. Feature models can represent graphically or by text the constraints on the compatibility between features. The information that a feature model contains enables the specification of valid software product configurations.

A software product should provide some functionality that satisfies requirements common among all the members of the product family (these software features are called in feature modeling terms “mandatory”) and possibly some additional functionality that is not shared by all members of the product family (these features are called “optional”). A feature can have several alternatives. A feature can contain one or more sub-features, which can be related to each other with an “and” relationship (all mandatory sub-features should be present) or an “or” relationship (one or more of the sub-features needs to be selected). Constraints can be set on the feature model; features can “exclude” or “require” each other [4].

The feature model needs to be configured in order to define a product instance. A valid configuration must contain all the mandatory features. Optional features may or may not be selected. If a selected feature has alternative implementations, at least one of the alternatives needs to be selected. A valid configuration should also satisfy the compatibility constraints among the features.

An introduction to feature modeling will be presented in this section using as an example a Mobile Elevating Work Platforms (MEWPs) (see section 4.1 where this case study is presented) SPL. The requirements for the MEWP product family lead to the design of the feature model presented in Fig 7. The root of the tree is the MEWP itself with the mandatory sub-features “Input”, “Control”, “CylinderControl” and the optional feature “CylinderSimulator”. The “VerticalUserCommand” feature has two alternative variations, the “Joystick” and the “Buttons”. The “AutomaticControl” feature, an optional sub-feature of the “Control”, has three alternative variations as described in the requirements.

To complete the feature model, the constraints are added. For example, the “AutomaticControl” feature “Requires” the “QuickButtons” feature. As an example of the “Excludes” constraint, the “BinaryControl” (an alternative variation of the “CylinderControl” feature) “Excludes” the “Joystick” feature.

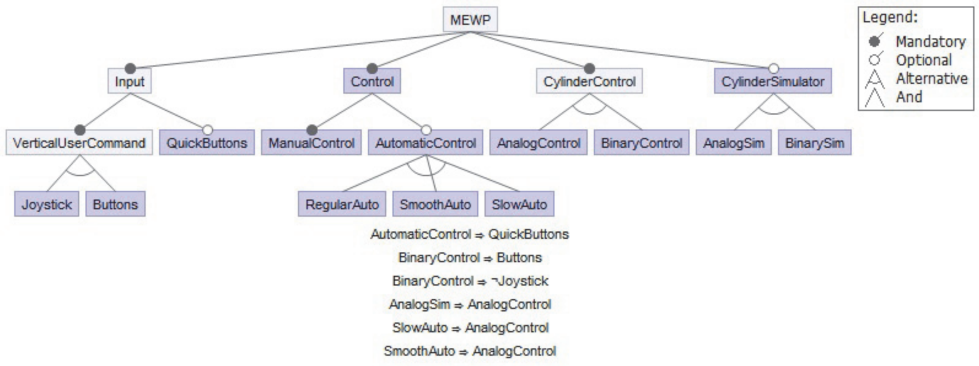


Figure 7, Feature model of the MEWP control software product family created using the FeatureIDE tool [55]

After the feature model for the software product family is defined, the user must select a set of features in order to configure of a specific product. An algorithm that checks the user choices against the feature model determines if the product is valid (see Fig. 8).

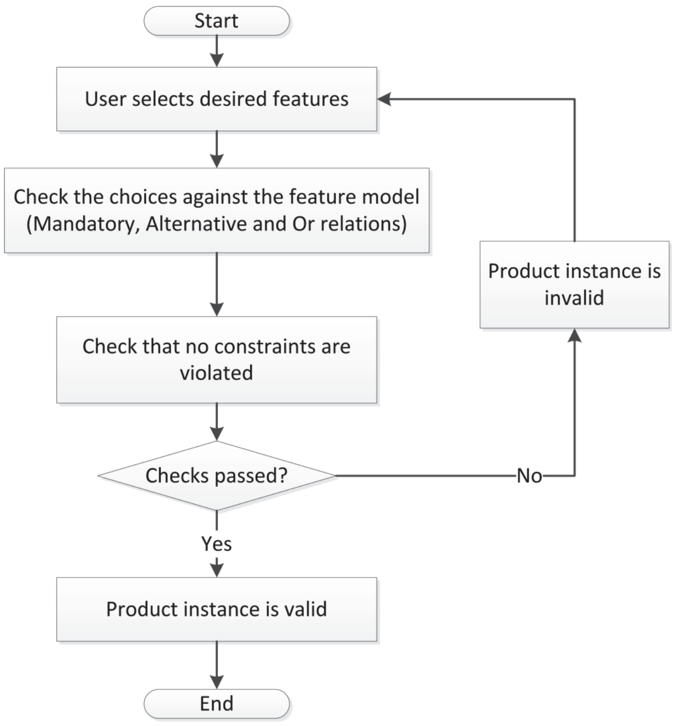

Start
User selects desired features
Check the choices against the feature model (Mandatory, Alternative and Or relations)
Check that no constraints are violated
Checks passed?
No
Yes
Product instance is invalid
Product instance is valid
End

Figure 8, An algorithm for checking whether a product instance is valid

5.2 Simulation based risk assessment

The Function Failure Identification and Propagation (FFIP) framework was developed to study the propagation of failure in complex systems early in the design stage and to present the effects in terms of functional losses [7, 31, 96, 97]. The simulation and reasoning approach in FFIP has its roots in qualitative physics [98] and qualitative reasoning [99-101]. FFIP represents system behavior as a finite set of component and flow status values, and performs reasoning based on qualitative relationships between functional and behavioral models of system components.

The first step in building a FFIP framework is the development of a Functional Model of the system under study. Functions are then mapped to the components in a Configuration Flow Graph, which captures the design. The components contain behavior logic that includes failure mode behavior, so the Configuration Flow Graph can be used to simulate the system-wide effects of component failures that are injected by the user. A Function Failure Logic monitors simulation signals and determines the health of the system functions in the Functional Model. Product line design alternatives can be specified as variants in the Configuration Flow Graph, after which a specific variant may be assessed by simulating the system response to a critical event scenario. Simulation runs that result in the loss of health of safety functions are identified for elimination from the product line.

The remainder of this section introduces the FFIP methodology by the simplest example that is sufficient for illustrating the different elements of the FFIP framework, so that the reader may understand how the results in the later sections were obtained. The example is a simplified water cooling system for a microprocessor (Fig. 9).

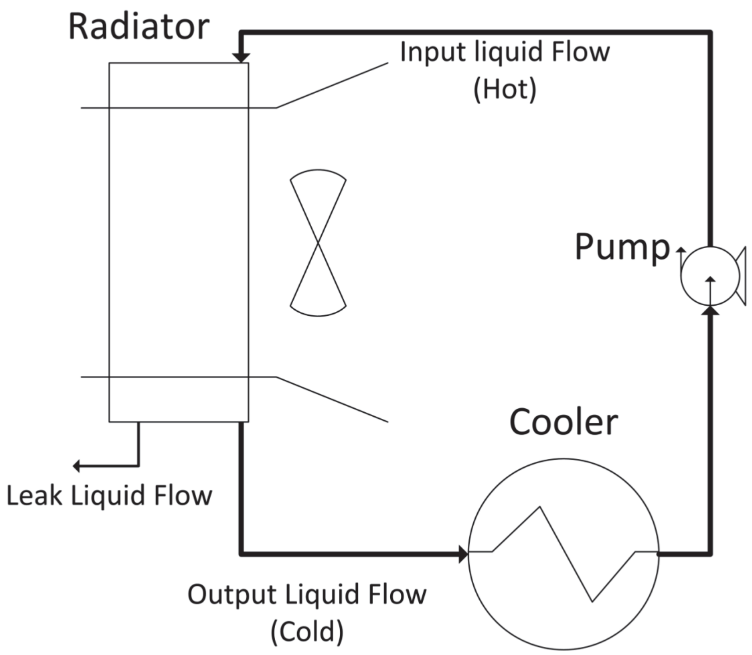


Figure 9, Piping and Instrumentation Diagram of the example water cooling process

The cooler is used to transfer heat from the microprocessor to the water flowing through the cooler. Warm water coming out from the cooler enters the radiator. The air flow generated by a fan attached to the radiator reduces the temperature of the water that flows through. The purpose is to obtain cold water that will flow back to the cooler. When the water in the radiator is at reference level, this circulation cools it adequately. If the water level is less than the reference level due to a leak, the output liquid flow is compromised and less flow of water is directed to the cooler. Additionally, if the water level in the radiator is less than the reference level, the water is not cooled efficiently and the temperature of the output flow is higher than normal. Also if there is less water in the system, then the water temperature will rise because the total available heat capacity will decrease. The transfer of heat that is performed by the cooler requires sufficient water flow and low water temperature.

If the water flow to the cooler is less than normal or the temperature of the water is higher than normal, then the temperature of the component that is generating heat, the microprocessor, will rise. The system detects this rise of temperature and acts when it reaches two alarm threshold values, tempA and tempB. If the microprocessor's temperature reaches tempA, the system starts the shutdown process which involves storing the system's state to non-volatile memory and notifying users and other networked systems of the upcoming shutdown. After the shutdown process is complete, the system powers off. If the microprocessor's temperature reaches tempB, in order to prevent physical damage to the system, the power is cut abruptly regardless of the progress of the shutdown process; this behavior can cause loss of data and may leave the system in an inconsistent state. The FFIP method is used to assess different design alternatives and determine which ones have an acceptable behavior in case of a critical failure scenario (a leak in the radiator).

The desired functionality is expressed as a functional model, according to the standard functional basis defined in [102](Fig. 10). The ultimate goal of failure propagation analyses with FFIP is to identify degradation or loss of these functions. A component failure may not necessarily result in health degradation of functions, and failure propagation may cause the loss of a function that is not necessarily associated with failed components. For these reasons, component failures are not interesting in their own right. FFIP simulates fault propagation in a component model and then reasons about the loss or degradation of functions in the functional model.

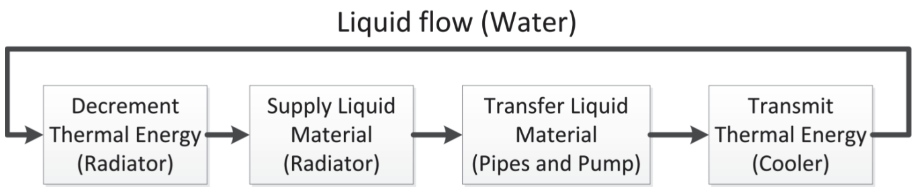

Liquid flow (Water)
Decrement Thermal Energy (Radiator)
Supply Liquid Material (Radiator)
Transfer Liquid Material (Pipes and Pump)
Transmit Thermal Energy (Cooler)

Figure 10, Functional model of the example water cooling process (see Fig. 9)

The functional model is not simulated, since simulation requires information about components, their connections and internal behavior. This information is captured in a configuration flow graph (CFG), which has been implemented in the Simulink tool (Fig. 11). The CFG and functional model have the same flows between functions and components, following the taxonomy defined in [102]. This makes it possible for a Function Failure Logic (FFL) to passively observe how abnormal flow levels propagate in the simulated CFG, and to use this information to determine if a function defined in the functional model (Fig. 10) has been degraded, lost recoverable or lost. The thresholds for these function health statuses are configured separately for each function; this depends on the application and an example is given below in the discussion related to Table 1. The FFL for the Supply liquid material (Radiator) function is shown in Fig. 12; it compares the output flow of the radiator with the reference flow shown in Fig. 11. The FFL for the Transmit Thermal Energy function is shown in Fig. 13; it evaluates the temperature of the cooler component shown in Fig. 11.

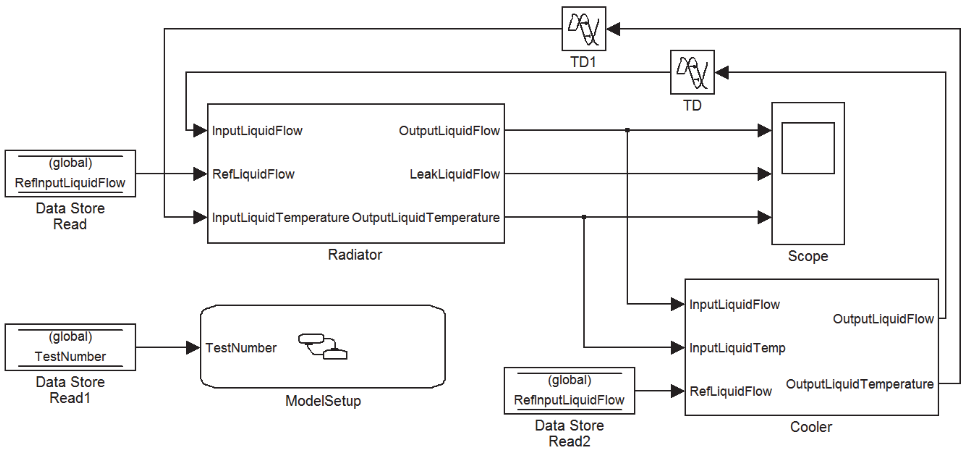


Figure 11, Configuration flow graph of the example water cooling process (see Fig. 9)

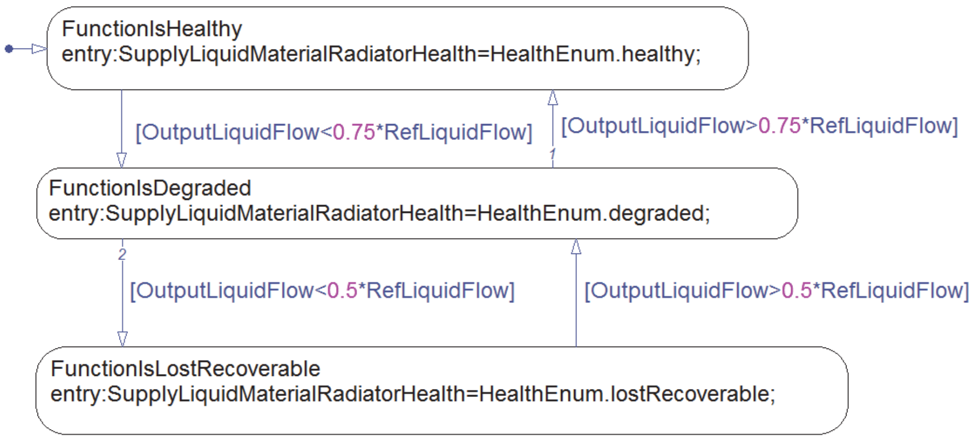

FunctionIsHealthy
entry:SupplyLiquidMaterialRadiatorHealth=HealthEnum.healthy;
[OutputLiquidFlow<0.75*RefLiquidFlow]
[OutputLiquidFlow>0.75*RefLiquidFlow]
1
FunctionIsDegraded
entry:SupplyLiquidMaterialRadiatorHealth=HealthEnum.degraded;
2
[OutputLiquidFlow<0.5*RefLiquidFlow]
[OutputLiquidFlow>0.5*RefLiquidFlow]
FunctionIsLostRecoverable
entry:SupplyLiquidMaterialRadiatorHealth=HealthEnum.lostRecoverable;

Figure 12, Function Failure Logic for the Supply liquid material (Radiator) function, which is related to the Radiator component (see Fig. 11).

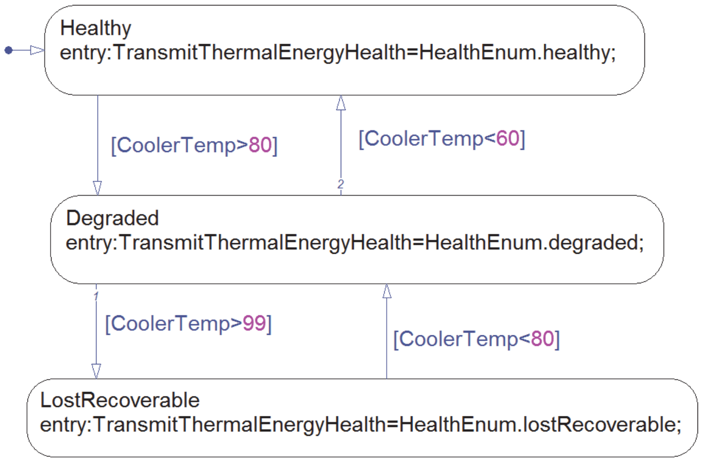

Healthy
entry:TransmitThermalEnergyHealth=HealthEnum.healthy;
[CoolerTemp>80]
[CoolerTemp<60]
Degraded
entry:TransmitThermalEnergyHealth=HealthEnum.degraded;
[CoolerTemp>99]
[CoolerTemp<80]
LostRecoverable
entry:TransmitThermalEnergyHealth=HealthEnum.lostRecoverable;

Figure 13, Function Failure Logic for transmit thermal energy function related to the Cooler component (see Fig. 11).

The relationship between input and output flows of a component in a CFG is defined by a behavioral model (BM). The BM for the radiator tank is shown in Fig. 14. Statecharts are used in behavioral modeling, and a state is defined for each nominal and failed mode of the component. In this case there is one failed mode: the radiator is leaking. Critical events may be injected to the simulation at any time, and these cause mode changes (e.g. the leakFailure event triggers a transition to the LeakingMode state.) The behavioral model of the cooler is shown in Fig 15.

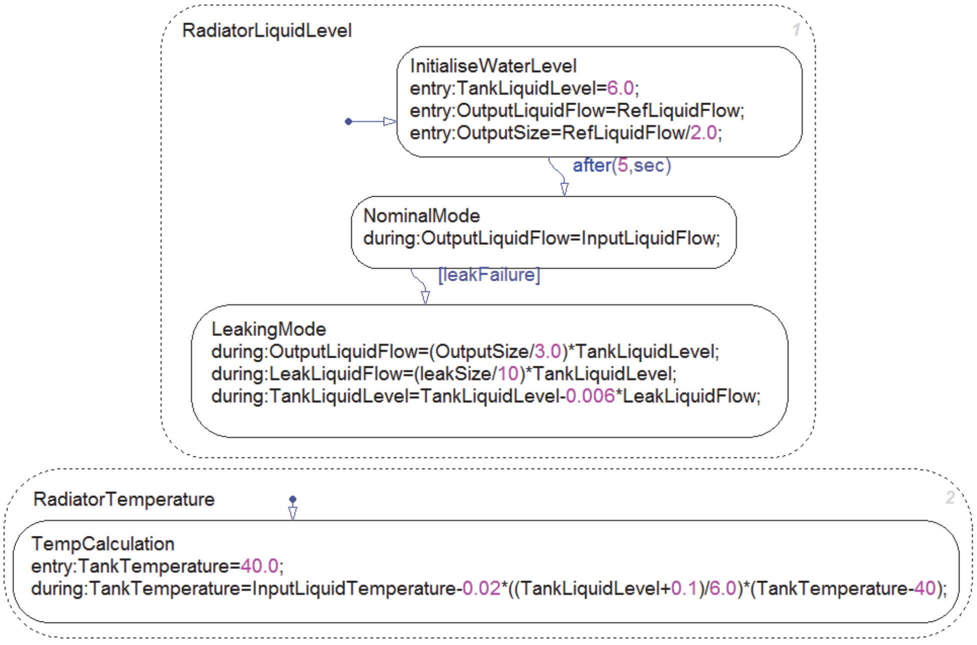

RadiatorLiquidLevel
InitialiseWaterLevel
entry:TankLiquidLevel=6.0;
entry:OutputLiquidFlow=RefLiquidFlow;
entry:OutputSize=RefLiquidFlow/2.0;
after(5,sec)
NominalMode
during:OutputLiquidFlow=InputLiquidFlow;
[leakFailure]
LeakingMode
during:OutputLiquidFlow=(OutputSize/3.0)*TankLiquidLevel;
during:LeakLiquidFlow=(leakSize/10)*TankLiquidLevel;
during:TankLiquidLevel=TankLiquidLevel-0.006*LeakLiquidFlow;
RadiatorTemperature
TempCalculation
entry:TankTemperature=40.0;
during:TankTemperature=InputLiquidTemperature-0.02*((TankLiquidLevel+0.1)/6.0)*(TankTemperature-40);

Figure 14, Behavioural model of the Radiator component (see Fig. 11).

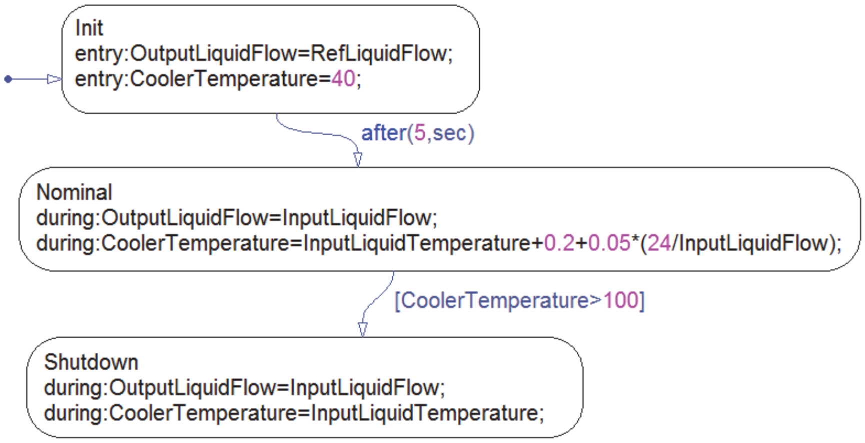

Init
entry:OutputLiquidFlow=RefLiquidFlow;
entry:CoolerTemperature=40;
after(5,sec)
Nominal
during:OutputLiquidFlow=InputLiquidFlow;
during:CoolerTemperature=InputLiquidTemperature+0.2+0.05*(24/InputLiquidFlow);
[CoolerTemperature>100]
Shutdown
during:OutputLiquidFlow=InputLiquidFlow;
during:CoolerTemperature=InputLiquidTemperature;

Figure 15, Behavioural model of the Cooler component (see Fig. 11).

In earlier versions of the FFIP framework, flow levels were described with an enumeration [zero, low, nominal, high] [7], but this approach is insufficient for our boiling water reactor case study, in which several positive and negative feedback loops affect a single flow. Would two feedback loops simply cancel each other out, if one of them increases a flow and the other decreases it? How can the effect of varying design parameters be captured in the simulation? In order to describe feedback loops in early phase designs, flow levels may be any value in the range [0..10] in this simulation. First order linear difference equations are used to relate input and output flows to each other. Consider the state NominalMode in Fig. 14. Since the behavioral model is executed by a fixed-step solver, the radiator level is a sequence, with one value for each simulation step. The first line of code in the nominal state is a linear difference equation that relates the current and previous elements in the sequence. The behavioral model is thus a system of first order linear difference equations relating the input flows, output flow and component's internal variables (such as the radiator level). The coefficients may either be fixed numbers or parameters that may be changed between successive simulation runs; an example of the latter is "leakSize" in the LeakingMode state.

Since during early phase design detailed component dimensions are not known and the range [0..10] is used for flow levels, the results obviously depend on how the model is parameterized. One article in this dissertation is focused on studying the effects of parameter changes in the behavioral models [103]. These parameters may either be design parameters, timing of critical event scenarios or parameters of faults. Changes in values of two parameters are studied in this example. The RefInputLiquidFlow parameter in Fig. 11 is the reference liquid flow that should go through the radiator and the cooler in normal operation of the process; it is a design parameter. In Fig. 14, within the LeakingMode state, the leakSize is a parameter of a fault.

The methodology used is to define a number of values of interest for the parameters to be varied and to systematically perform FFIP simulation to identify those combinations of parameter values that result in degradation or loss of functions. The choice of values is done by an analyst who understands the application domain. The flowchart in Fig. 16 illustrates the methodology in the case of two parameters, and additional parameters can be handled by adding a nested loop for each new parameter. Fig. 17 displays the graph that is obtained in the flowchart when parameter 1 is RefInputLiquidFlow, S_{p1} (the set of values for parameter p1) is [6.0, 12.0, 24.0], parameter 2 is leakSize and S_{p2} is [1.2, 2.4, 4]; each curve in the graph is the trend for the radiator level in one simulation run. Some interesting simulation parameters can be logged and presented in graphs. The liquid level of the radiator is shown in Fig. 17 and the temperature of the cooler is show in Fig. 18. The output of Function Failure Logic for the simulation is shown in Table 1. The Decrement Thermal Energy function is lost when the temperature of the radiator rises above a threshold. The status is degraded, since partial cooling is still achieved. The Transmit Thermal Energy function, which is associated with the cooler (thermal energy is transmitted from the hot process element to the cooler), is first degraded as the first temperature alarm limit is exceeded; in this case the microprocessor is no longer able to perform its normal tasks but it still may perform a controlled shutdown as explained previously. When the second alarm limit is exceeded, the processor is powered off immediately to prevent hardware damage, so the function status is lost. It is lost recoverably, since no hardware damage occurred and the system may resume normal functioning after the radiator is fixed.

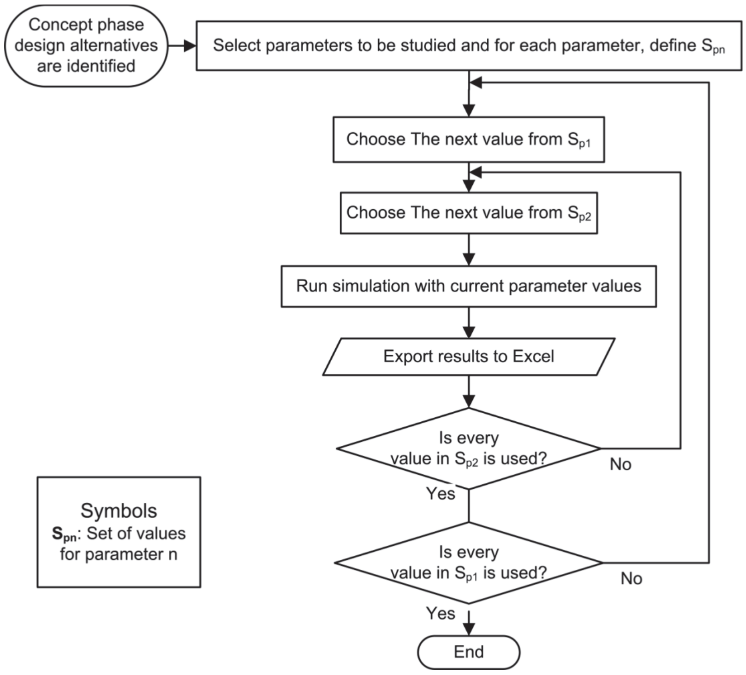

Concept phase design alternatives are identified
Select parameters to be studied and for each parameter, define S_{pn}
Choose The next value from S_{p1}
Choose The next value from S_{p2}
Run simulation with current parameter values
Export results to Excel
Is every value in S_{p2} is used?
No
Yes
Is every value in S_{p1} is used?
No
Yes
End
Symbols
S_{pn}: Set of values for parameter n

Figure 16, A flowchart describing how every combination of parameter values is simulated systematically to determine those combinations of parameter values that result in degradation or loss of functions

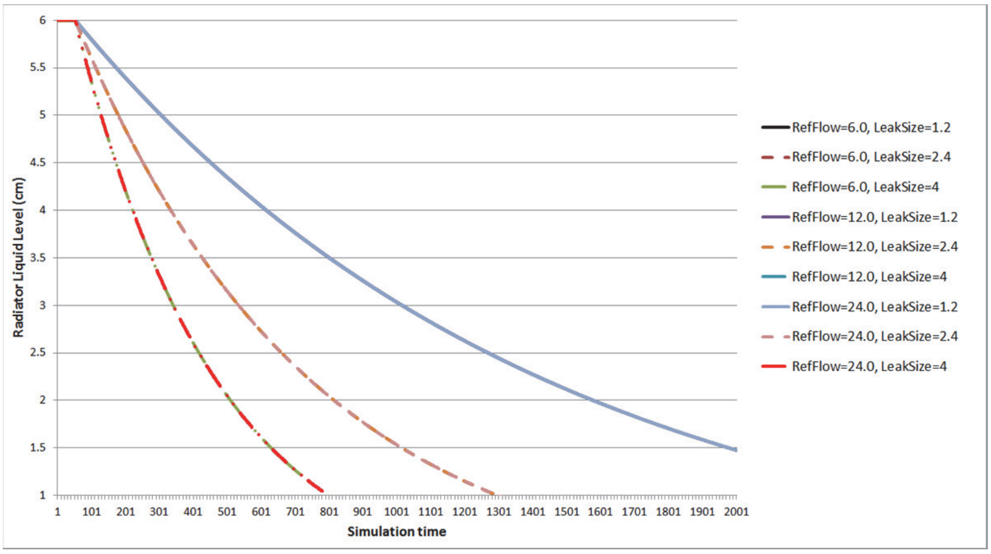


Figure 17, The radiator liquid level result of performing the procedure in Fig. 16 when parameter 1 is RefInputLiquidFlow and parameter 2 is leakSize. The level drop due to the leak flow does not depend on the reference flow through the radiator. It only depends on the liquid level in the radiator and the size of the leak.

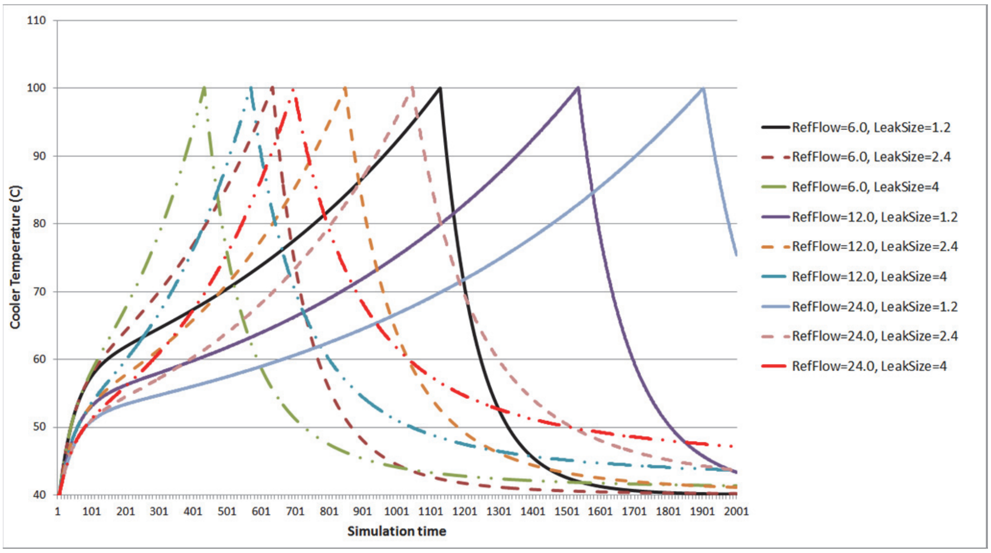
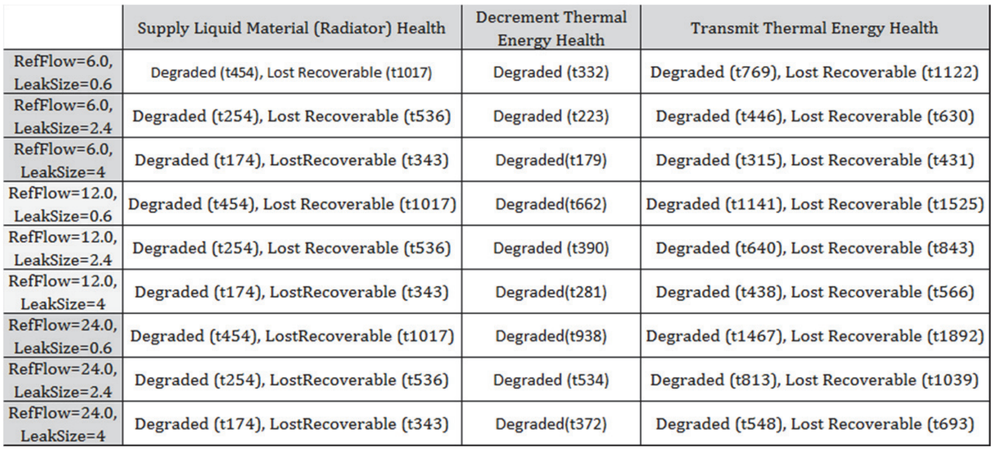


Figure 18, The cooler temperature result of performing the procedure in Fig. Flowchart_alt when parameter 1 is RefInputLiquidFlow and parameter 2 is leakSize. The worst behaviour is for the combination of the smallest reference flow and the most severe leak. The best three curves are the ones where the leak is the smallest.



	Supply Liquid Material (Radiator) Health	Decrement Thermal Energy Health	Transmit Thermal Energy Health
RefFlow=6.0, LeakSize=0.6	Degraded (t454), Lost Recoverable (t1017)	Degraded (t332)	Degraded (t769), Lost Recoverable (t1122)
RefFlow=6.0, LeakSize=2.4	Degraded (t254), Lost Recoverable (t536)	Degraded (t223)	Degraded (t446), Lost Recoverable (t630)
RefFlow=6.0, LeakSize=4	Degraded (t174), LostRecoverable (t343)	Degraded(t179)	Degraded (t315), Lost Recoverable (t431)
RefFlow=12.0, LeakSize=0.6	Degraded (t454), Lost Recoverable (t1017)	Degraded(t662)	Degraded (t1141), Lost Recoverable (t1525)
RefFlow=12.0, LeakSize=2.4	Degraded (t254), Lost Recoverable (t536)	Degraded (t390)	Degraded (t640), Lost Recoverable (t843)
RefFlow=12.0, LeakSize=4	Degraded (t174), LostRecoverable (t343)	Degraded(t281)	Degraded (t438), Lost Recoverable (t566)
RefFlow=24.0, LeakSize=0.6	Degraded (t454), LostRecoverable (t1017)	Degraded(t938)	Degraded (t1467), Lost Recoverable (t1892)
RefFlow=24.0, LeakSize=2.4	Degraded (t254), LostRecoverable (t536)	Degraded (t534)	Degraded (t813), Lost Recoverable (t1039)
RefFlow=24.0, LeakSize=4	Degraded (t174), LostRecoverable (t343)	Degraded(t372)	Degraded (t548), Lost Recoverable (t693)

Table 1, The output of Function Failure Logic for the simulation in Fig. 17 and Fig. 18. In parenthesis is the simulation time in which the function health changed status

The purpose of this simple example is not to demonstrate the FFIP as a simulation based risk assessment method; the more complex boiling water reactor case study is used for this purpose. This purpose of this section has been to describe how the results in later sections were obtained.

5.3 Choice of case studies

Since the objectives of the dissertation concern development of new methodology, case studies are used to demonstrate the methodology. The recirculation pump subsystem has been chosen to illustrate the early phases of the methodology (Fig. 2 steps 1-3), since this subsystem has many dependencies to physical systems as well as other automation systems. These dependencies make it a mechatronic application that is difficult to study using conventional risk analysis methods, so there is a need for simulation-based risk assessment, as advocated in this dissertation.

The early phases of the development process result in a feature model describing all combinations of design alternatives that passed the risk assessment. Thus, the source information of the later phases is such a feature model, and the technology used from this point onward is not concerned with how this source information is obtained. Therefore, it is considered acceptable to use another case from this point onward. The mobile elevating working platform is used to demonstrate the process of supporting PLC targets with SPL technology (Fig. 2 steps 4 and 5).

This mobile elevating work platform is representative of safety critical work machinery. The Boiling Water Reactor (BWR) is representative of safety critical industrial processes, since beyond the fission reaction at the core of the reactor vessel, the nuclear power plant has many similarities to conventional power plants.

Further research may evaluate the method with additional case studies and use a case study for the steps 1-5. At the minimum, the steps 1-3 and the steps 4-5 should all be carried out for the same case study. The interface between step 3 and step 4 consists of passing a feature model that specifies the range of safe product line variants. This is a well defined interface with a formal syntax and semantics, and the methodology presented here makes no assumptions on tacit knowledge being passed between steps 3 and 4. For this reason, it is possible to evaluate steps 1-3 and steps 4-5 with different case studies.

6 Results

This section contains the results of the application of the methodology for system design and risk assessment for safety critical control software product lines, presented in the objectives section of this dissertation (section 3).

6.1 Failure propagation analysis with the boiling water reactor

The FFIP risk assessment method, introduced in section 5.2, is applied to the BWR case study, presented in section 4.2. The BWR case study was selected as an example of the early design of a safety critical complex system.

The first step of our proposed methodology (Fig. 2) is to identify the desired functionality and express the processes inside the BWR as a set of functions. A functional model can be created by connecting these functions with flows of Energy, Material or Signal (EMS). The functional basis [102] standard provides a list of function and flow names that can be used in the functional model. The functional basis has been carefully formulated to avoid assumptions on design and implementation decisions, which would imply certain component choices. Several design alternatives will be presented in later sections to satisfy the functional model of the BWR case study (see Fig. 5) presented in (Fig. 19).

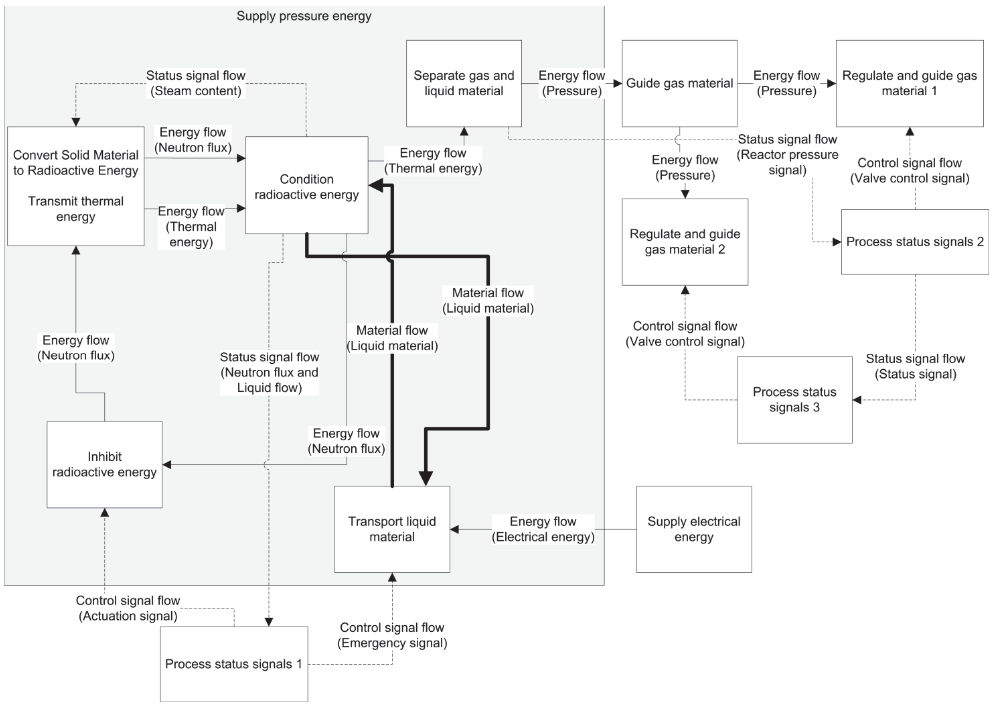

Supply pressure energy
Status signal flow (Steam content)
Separate gas and liquid material
Energy flow (Pressure)
Guide gas material
Energy flow (Pressure)
Regulate and guide gas material 1
Convert Solid Material to Radioactive Energy
Transmit thermal energy
Energy flow (Neutron flux)
Condition radioactive energy
Energy flow (Thermal energy)
Status signal flow (Reactor pressure signal)
Energy flow (Pressure)
Control signal flow (Valve control signal)
Energy flow (Thermal energy)
Regulate and guide gas material 2
Process status signals 2
Material flow (Liquid material)
Material flow (Liquid material)
Control signal flow (Valve control signal)
Energy flow (Neutron flux)
Status signal flow (Neutron flux and Liquid flow)
Status signal flow (Status signal)
Process status signals 3
Energy flow (Neutron flux)
Inhibit radioactive energy
Transport liquid material
Energy flow (Electrical energy)
Supply electrical energy
Control signal flow (Actuation signal)
Control signal flow (Emergency signal)
Process status signals 1

Figure 19, Functional model of the reactor core and steam outlets [51]

The second step of the methodology in Fig. 2 is to specify several alternatives for obtaining the required functionality. This step is omitted in this subsection, which illustrates the application of FFIP to a single design; the application to a family of designs is illustrated in section 6.3. The third step in Fig. 2 is to create the Configuration Flow Graph (CFG) that contains components that can be simulated.

One function of the functional model can be related to one or more components and a component of the CFG can be related to one or more functions. The CFG contains the components that implement the functions of the functional model. The components are connected with the same EMS flows as in the functional model. A CFG that implements the functional model of Fig. 19 is presented in Fig. 20, as an overview, and is refined and extended in Fig. 21 as a Simulink model. The Simulink model in the Matlab environment can be simulated and provides the results in sections 6.2 and 6.3. The CFG can have a hierarchical structure. As an example, Fig 22 shows the contents of the Reactor component in Fig. 21. The mapping between functions and components for this case study is presented in Table 2.

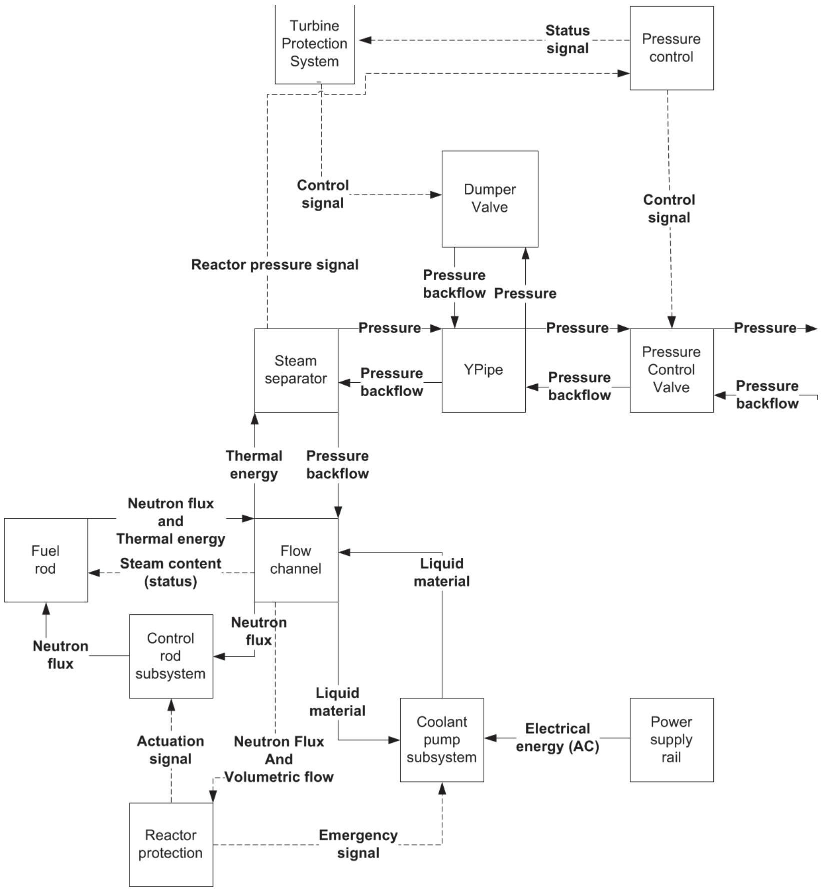


Figure 20, The Configuration Flow Graph model related to the functional model of the BWR in Fig. 19 [51]

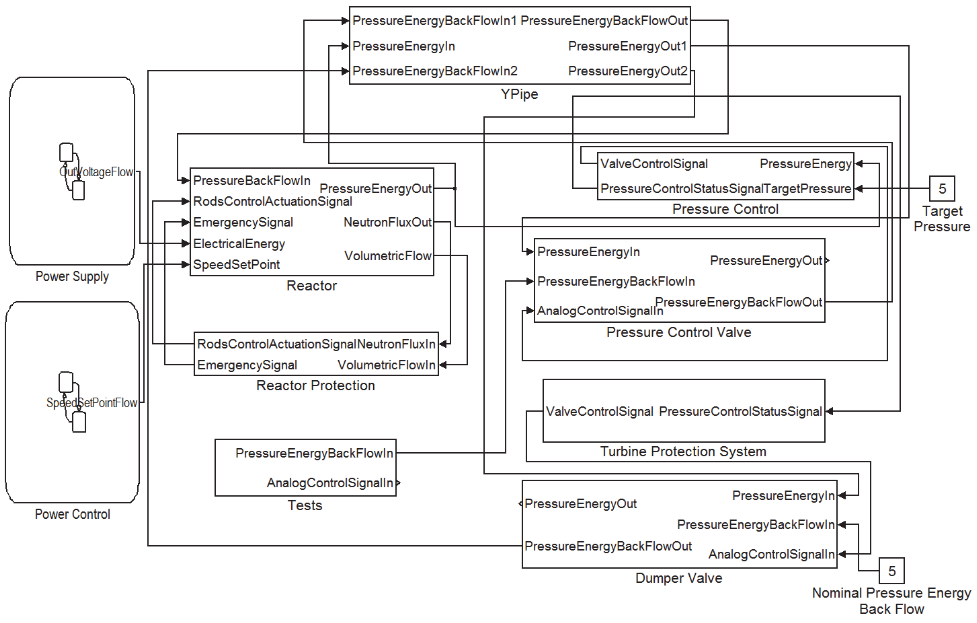


Figure 21, The Configuration Flow Graph top level model in Simulink, related to the functional model of the BWR in Fig. 19, adapted from [103]

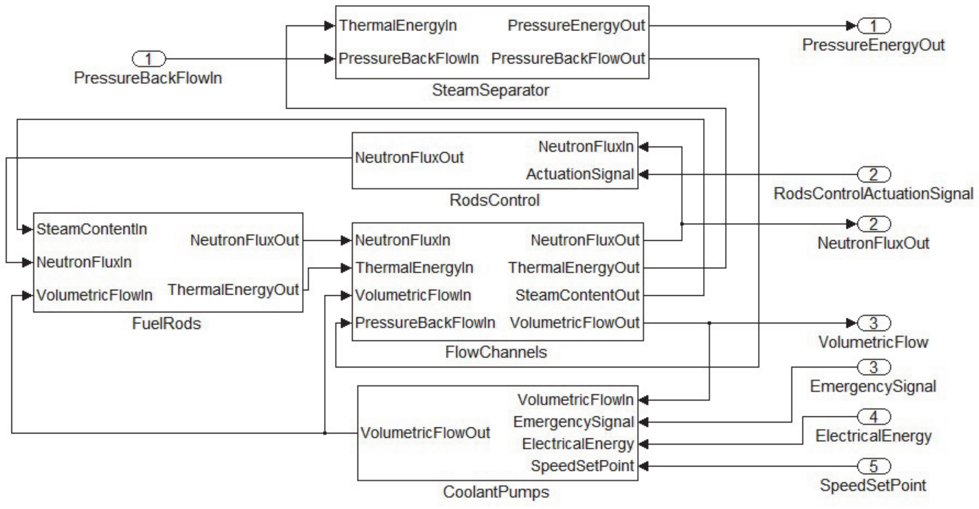


Figure 22, The internal of the Reactor component of the Configuration Flow Graph in Fig. 21 [103]

Function	Component
Supply pressure energy	Reactor core
Convert Solid Material to Radioactive Energy	Fuel rods
Transmit thermal energy	Fuel rods
Condition radioactive energy	Flow channels
Inhibit radioactive energy	Control rods
Transport liquid material	Coolant pumps
Process status signals 1	Reactor protection
Separate gas and liquid material	Steam separator
Supply electrical energy	Power supply rail
Guide gas material	Y pipe
Regulate and guide gas material 1	Pressure control valve
Regulate and guide gas material 2	Dumper valve
Process status signals 2	Pressure control
Process status signals 3	Turbine protection

Table 2, Mappings between functions of the functional model in Fig. 19 and components of the Configuration Flow Graph in Fig. 21 and Fig. 22 [51]

The components in the CFG contain behavioral code at different levels of detail, depending on the available information in this early design phase. In Fig. 23 the different modes of operation of the coolant pumps controller (see Fig. 22) are presented. The behavioral models of the components are expressed as statecharts using the Stateflow language in Simulink.

The Stateflow chart for the coolant pumps controller (Fig. 23) contains three parallel processes. One process controls the output voltage to the inverters that drive the pumps (OutVoltageControlState), one process keeps track of the emergency power (emergPower) and the third process keeps track of the scram state of the controller (ScramState). The state of the latter two are used to specify transition guards in the former. During normal operation of the plant, the coolant pump controller receives its setpoint from the Power control subsystem (see Fig. 21). As mentioned in section 4.2, the coolant pumps control the power production of the plant as well as the cooling of the reactor core. When there is external power available from the Power Supply subsystem, then the emergency power of the coolant pumps is charged. Emergency procedures are initiated if voltage sags or transients occur in the external power supply or if a SCRAM signal is received from the reactor protection subsystem. In such cases, the controller drives the pumps to ramp down to minimum rotations per minute (RPM) or to zero RPM, if the pumps rely on the emergency power supply. If the external power returns to the pumps while they are running at minimum RPM, and there is no SCRAM signal recorded, the controller resumes normal operation. If external power is restored before the pumps have stopped, the controller will maintain minimum RPM, if the SCRAM state is active, or resume normal operation otherwise. If the pumps reach zero RPM, or if the emergency power is lost while it is used, then the controller considers the pumps lost and their operation cannot be recovered without operator action (state DeadStopUnrecoverable). If the controller receives the emergency signal and enters the ScramState, it can return to the NoScram state if the emergency signal is not present anymore and the operator acknowledges the SCRAM after a delay has expired.

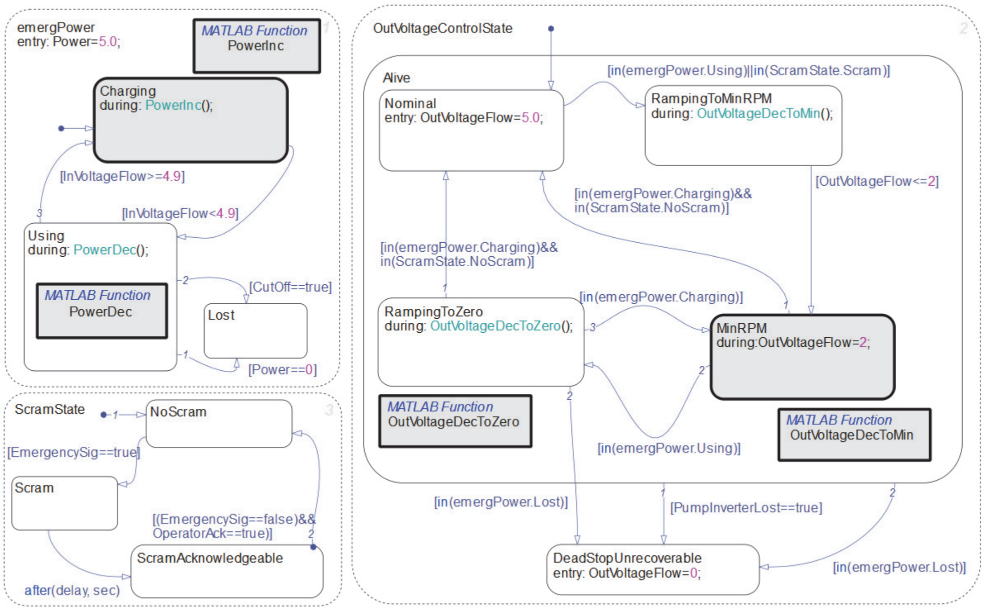


Figure 23, Behavioural model of the CoolantPumps component in Fig. 22 [51]

The health of the functions in Fig 19 is determined by Function Failure Logic (FFL) that monitors the simulation parameters and contains the logic that determines the health of a function. FFL may make decisions either by comparing several flows or component state variables in the simulation or by using thresholds for a single variable. The FFL for the health of the transmit thermal energy function (see Fig. 19), which is related to the fuel rods in Fig. 22, is presented in Fig. 24. The FFL for the Transport liquid material function, which is related to the coolant pumps in Fig. 22, is presented in Fig. 25.

The health of the transmit thermal energy function (Fig. 24) is determined only by the temperature of the fuel rods inside the reactor component (see Fig. 22). The FFL related to the transport liquid material function (Fig. 25) determines the health of that function by evaluating the input and output liquid flows and the emergency signal.

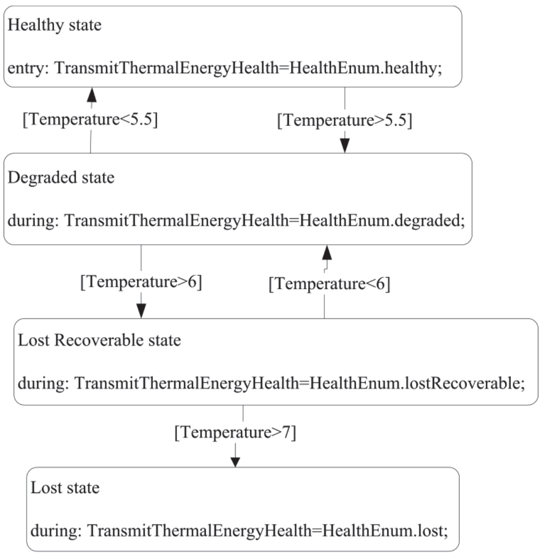

Healthy state
entry: TransmitThermalEnergyHealth=HealthEnum.healthy;
[Temperature<5.5]
[Temperature>5.5]
Degraded state
during: TransmitThermalEnergyHealth=HealthEnum.degraded;
[Temperature>6]
[Temperature<6]
Lost Recoverable state
during: TransmitThermalEnergyHealth=HealthEnum.lostRecoverable;
[Temperature>7]
Lost state
during: TransmitThermalEnergyHealth=HealthEnum.lost;

Figure 24, Functional Failure Logic that determines the health of the transmit thermal energy function (Fig. 19) related to the FuelRods component in Fig 22 [103]

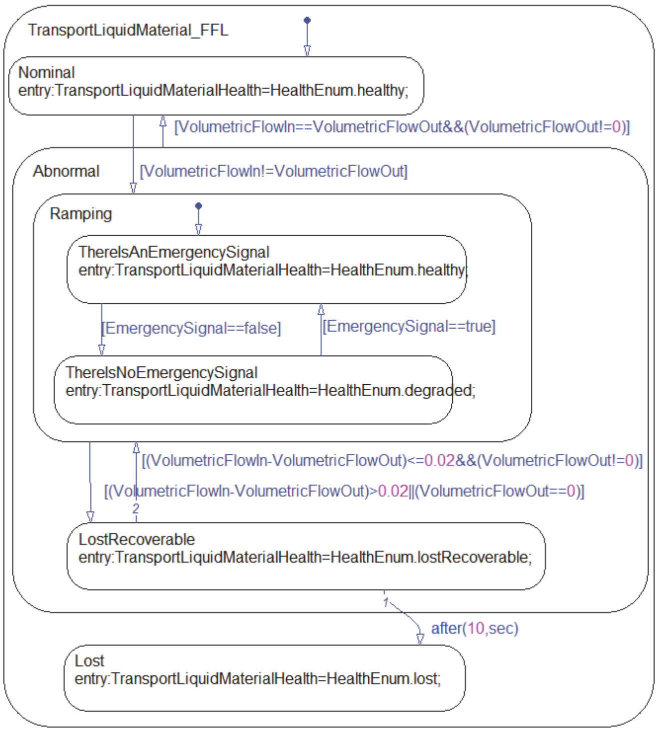

TransportLiquidMaterial_FFL
Nominal
entry:TransportLiquidMaterialHealth=HealthEnum.healthy;
[VolumetricFlowIn==VolumetricFlowOut&&(VolumetricFlowOut!=0)]
Abnormal
[VolumetricFlowIn!=VolumetricFlowOut]
Ramping
ThereIsAnEmergencySignal
entry:TransportLiquidMaterialHealth=HealthEnum.healthy;
[EmergencySignal==false]
[EmergencySignal==true]
ThereIsNoEmergencySignal
entry:TransportLiquidMaterialHealth=HealthEnum.degraded;
[(VolumetricFlowIn-VolumetricFlowOut)<=0.02&&(VolumetricFlowOut!=0)]
[(VolumetricFlowIn-VolumetricFlowOut)>0.02||(VolumetricFlowOut==0)]
2
LostRecoverable
entry:TransportLiquidMaterialHealth=HealthEnum.lostRecoverable;
1
after(10,sec)
Lost
entry:TransportLiquidMaterialHealth=HealthEnum.lost;

Figure 25, Functional Failure Logic that determines the health of the transport thermal energy function (Fig. 19) related to the CoolantPumps component in Fig 22 [51]

6.2 Analyzing critical event scenarios

6.2.1 Study of the basic critical event scenario

The effect of injecting two concurrent, independent failures was analyzed with the FFIP framework [51]. At simulation time t40 the coolant pumps loose the external power supply and the pressure control subsystem suffers a software malfunction and closes the pressure control valve without notifying the turbine protection system (see the CFG model of the case study in Fig. 20). This critical event scenario leads to the pressure control hazard described in section 4.2.

The output in Fig. 26 is generated by monitoring the most important flows from the reactor core CFG model, ready to be simulated as a Simulink model, presented in Fig. 22. Additionally the temperature of the fuel rods (“FuelRodsTemp” signal) was monitored from within the “FuelRods” component of the reactor core CFG Simulink model. The graph in Fig. 26 shows how the levels of flows change within the qualitative range [0…10] [103] during the concurrent failure of coolant power supply and pressure control system. The behavioral models are parameterized so that when all components are running at nominal mode, all flows have the nominal value of 5; this is why all curves in Fig. 26 diverge at the time when component failures are injected.

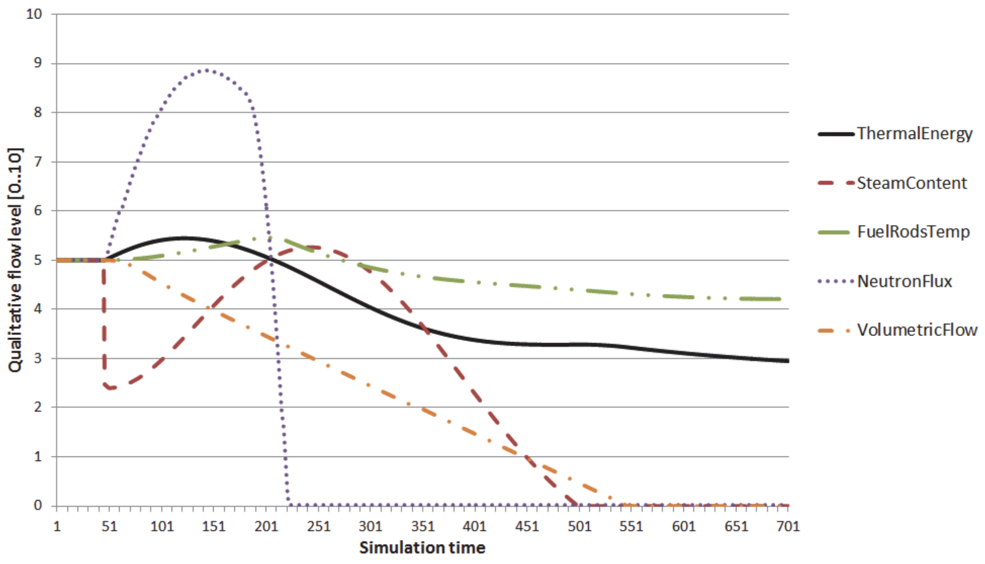


Figure 26, Behaviour at reactor core; both failures occur at t=40, coolant pumps can use their emergency power [51].

At t=40, a loss of power supply to coolant pumps occurs in conjunction with a software malfunction in the “Pressure control” component, which drives the pressure control valve shut. Since it is a software malfunction, the turbine protection system is not notified. The pressure transient resulting from sudden valve closure is propagated upstream to the YPipe (by the pressure backflow arrows in Fig. 21). Since the turbine protection system was not notified, it does not open the dumper valve, so the pressure shockwave is propagated to the reactor and causes a sharp drop in the steam content (see the “SteamContent” flow in Fig. 26). This improves moderation and causes neutron flux (see the “Neutronflux” flow) to exceed scram threshold (6), which triggers the ramp of reactor coolant pumps (see the “VolumetricFlow” flow). Since the power rail was lost due to an injected failure, the pumps use their emergency power supply.

The SCRAM causes control rods to be inserted, and these bring the neutron flux down after a delay. In the meanwhile, although there is an increased thermal energy flow from the fuel rods (see the “ThermalEnergy” flow), the combined effect of the coolant flow ramp (see the “VolumetricFlow” flow) and the Doppler effect keep the transmit thermal energy function healthy (fuel rods temperature > 5.5 is interpreted by the FFL reasoner as degraded (Fig. 24), see the “FuelRodsTemp” signal in Fig. 26).

6.2.2 Loss of emergency power supply to coolant pumps added to failure scenario

The following failure is added to the scenario presented in section 6.2.1. At simulation time t5, the coolant pumps’ emergency power supply is lost (internal state change in pump subsystem – no effect on the flow levels). The flow levels at the reactor core CFG Simulink model (Fig. 22) are shown in Fig. 27.

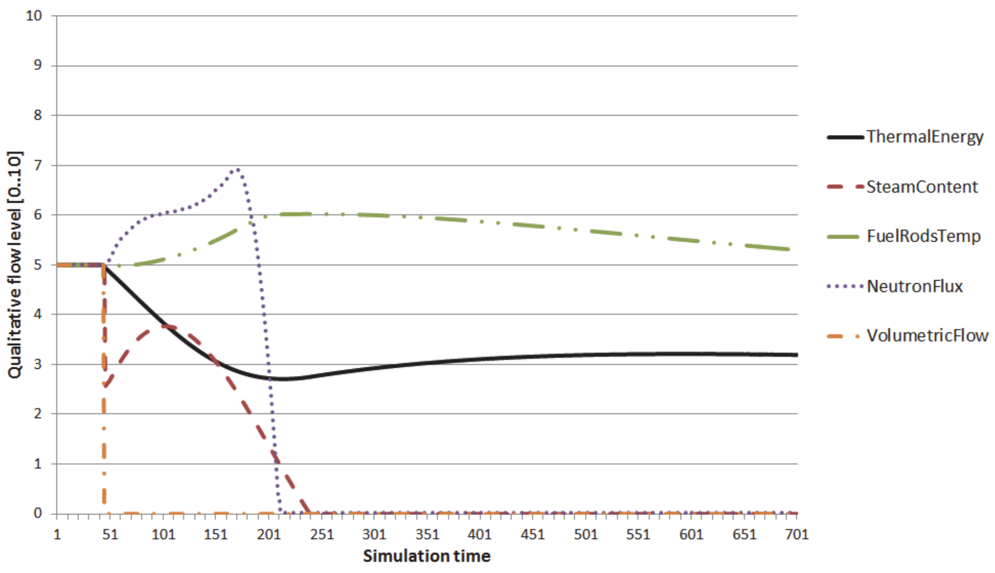


Figure 27, Behaviour at reactor core when power supply and pressure control malfunction occur at t=40. Volumetric flow drops sharply to zero since emergency power is not available due to injected failure [51].

An effect of this injected failure is that the coolant flow (see the “VolumetricFlow” flow in Fig. 22) stops immediately after the loss of the power to the pumps. The SCRAM is triggered by the coolant flow dropping under 3.0, which in this case occurs before the neutron flux scram threshold is exceeded (see the “Neutronflux” flow).

The loss of coolant flow causes steam buildup around the fuel rods (see the “SteamContent” flow), so the thermal energy transfer from rods to coolant is slower (see the “ThermalEnergy” flow). The fuel rod temperature exceeds the level 5.5 (transmit thermal energy function’s health becomes degraded) and later the level 6.0 (transmit thermal energy function’s health becomes lostRecoverable), see the “FuelRodsTemp” signal.

The transmit thermal energy function’s health recovers to normal as the fuel rod’s temperature drops after the control rods reduce the neutron flux (see the “Neutronflux” flow and “FuelRodsTemp” signal). The implication of the recovery for risk estimation is that the hazard can be given a reduced preliminary severity ranking, which naturally must be confirmed by probabilistic methods after detailed design information is available.

6.2.3 Propagation paths of function failures

Fig . 27 does not yet show the propagation of functional failures, but the FFL reasoner observing these flow levels gives the following output regarding functional health changes. At simulation time t5, the failure related to the loss of the emergency power supply of the coolant pumps is injected; this does not affect the health of the transport liquid material function as long as the supply electrical energy function is healthy. At simulation time t40, the failures of the power rail (which affects the health of the supply electrical energy function) and the software failure of the pressure control component (which affects the health of the process status signals function) are injected. Soon after, the transport liquid material function’s health changes to lostRecoverable, since it has no main or emergency power supply. Also, the regulate and guide gas material function’s health falls to degraded and then to lostRecoverable because of the pressure transient that it is produced by the sudden closure of the pressure control valve. The combined effects of the pressure transient and the loss of the transport liquid material function affect the health of the condition radioactive energy function which falls to degraded and then lostRecoverable health status. This condition affects the health of the transmit thermal energy function, which drops to degraded at time 152 and then to lostRecoverable at time 204. This information has been added to the functional model in Fig. 28.

These results do not give real-time information, but identify possible system wide effects of component failures. This identification must be done by human judgment when using established methods such as FMECA.

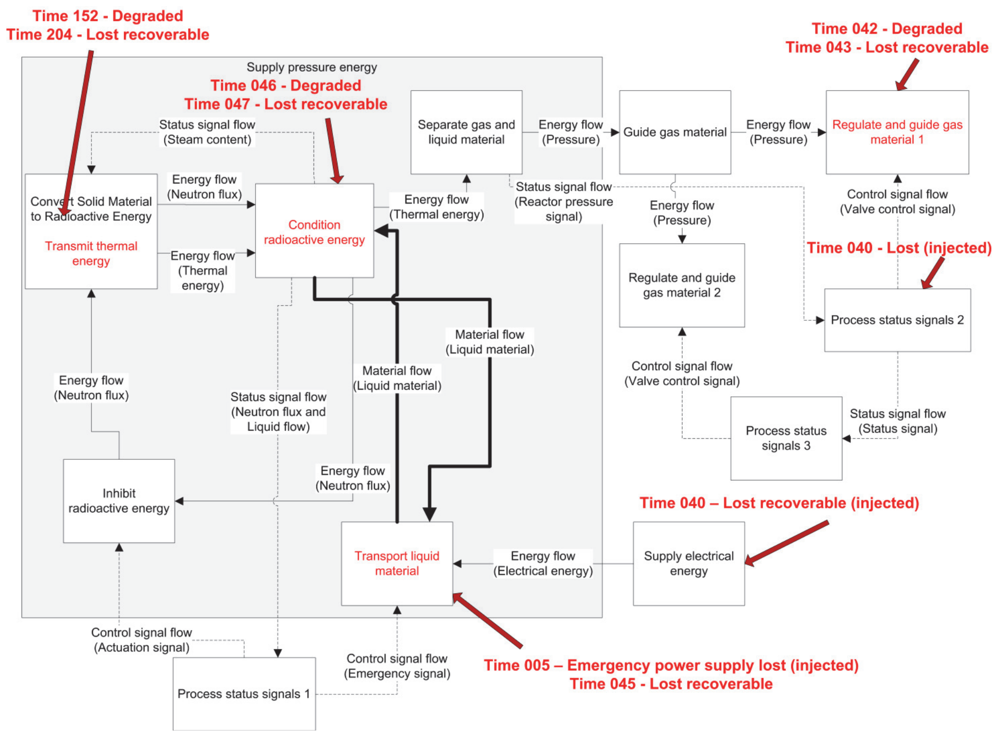

Time 152 - Degraded
Time 204 - Lost recoverable
Time 042 - Degraded
Time 043 - Lost recoverable
Supply pressure energy
Time 046 - Degraded
Time 047 - Lost recoverable
Status signal flow (Steam content)
Separate gas and liquid material
Energy flow (Pressure)
Guide gas material
Energy flow (Pressure)
Regulate and guide gas material 1
Convert Solid Material to Radioactive Energy
Transmit thermal energy
Energy flow (Neutron flux)
Condition radioactive energy
Energy flow (Thermal energy)
Status signal flow (Reactor pressure signal)
Energy flow (Pressure)
Control signal flow (Valve control signal)
Energy flow (Thermal energy)
Time 040 - Lost (injected)
Regulate and guide gas material 2
Process status signals 2
Material flow (Liquid material)
Material flow (Liquid material)
Control signal flow (Valve control signal)
Energy flow (Neutron flux)
Status signal flow (Neutron flux and Liquid flow)
Process status signals 3
Status signal flow (Status signal)
Energy flow (Neutron flux)
Inhibit radioactive energy
Time 040 – Lost recoverable (injected)
Transport liquid material
Energy flow (Electrical energy)
Supply electrical energy
Control signal flow (Actuation signal)
Control signal flow (Emergency signal)
Time 005 – Emergency power supply lost (injected)
Time 045 - Lost recoverable
Process status signals 1

Figure 28, Propagation of functional failures displayed on the functional model of the BWR (Fig. 19), displaying the results of the Functional Failure Logic reasoners, adapted from [51]

6.2.4 Multiple simulation runs with different scenario parameter values

In sections 6.2.2 – 6.2.3, one scenario was presented with specific timing of injected events and model parameters. It is possible to run a series of simulation runs having different sequence and spacing of events. The behavioral model's input and output flow relationships are parameterized according to expert judgment, so it is possible to perform several FFIP runs to cover a range of values for a parameter. The value of doing these multiple simulation runs at the concept phase is that safety constraints of certain parameters of the mechatronic design are discovered much earlier, making it possible to integrate safety considerations to the task of finding the optimal mechatronic design by integration of technical design subdisciplines.

To demonstrate this aspect of the framework, the time between the injected failure of the power rail and the injected software malfunction of the pressure control was treated as a simulation parameter. A Matlab script was used to run the simulation 8 times with the relative time between failures, measured in simulation steps, taking the values from -30 to 40 with a step of 10 (negative values mean that the pressure control malfunction happens before the loss of the power rail), as shown in Fig. 29.

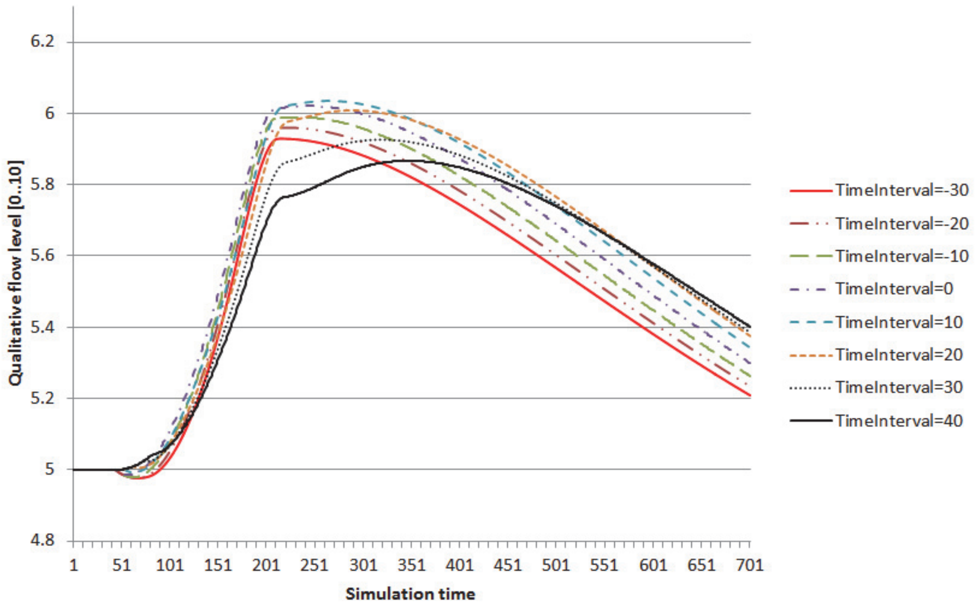


Figure 29, Fuel rods temperature in 8 simulation runs with different sequencing of injected failures. TimeInterval = TPressureControlMalfunction – TpowerRailFailure [51]

In three of the runs the temperature exceeds the value of 6.0 which the FFL interprets as the limit for the transition of the Transmit thermal energy function's health from degraded to lostRecoverable. There is a trend for the temperature to have a lower peak value when the power rail lost failure is injected later than the pressure control software failure, and this is due to the pumps being able to ramp down partially before losing power. While absolute numerical values are not meaningful, the observed trend can be taken into account by designers. If it is possible to identify a common-cause failure behind these two injected failures, the design should prevent their simultaneous occurrence. For example, an electric spike could cause loss of the power rail and hardware damage to the pressure controller, resulting in the control output being stuck low. Designers can guard against this by designing the emergency power in a way that cannot be affected by the spike; if this is not possible, reactor shutdown might need to be initiated if emergency power is lost. The value of FFIP over established risk assessment methods is that mechatronic system engineers are aware of such design constraints before beginning detailed design.

6.2.5 Multiple simulation runs with different component parameter values

Apart from assessing the system design for different critical event scenarios (in section 6.2.4), it is equally important to be able to run the FFIP method for different design parameters. Trends that emerge in the system's behaviour that are linked to a specific design parameter can be identified and help the designer improve the system's design. In this section different values for a component parameter, related to the Heat Transfer Coefficient (HTC) between the fuel rods and the coolant, were used to run a number of simulations for the critical event scenario presented in Table 3, which is similar to the scenario presented in section 6.2.2 [103].

Time	Scenario events
t043	Emergency power supply of coolant pumps is lost
t077	Software failure injected in pressure control valve. It starts oscillating between nominal and half closed
t078	The external power supply is lost

Table 3, Critical event scenario used in the simulations in Fig. 31 [103]

With the value of 5 as our baseline for the HTC, the simulation started with the value of 2 (smaller HTC than the baseline) and with increments of 1 it continued until the value of 8 (bigger HTC than the baseline). The comparison of the fuel rod temperature results reveals a trend of the improved fuel rod cooling in emergency situations as the HTC is increased, as shown in Fig. 31.

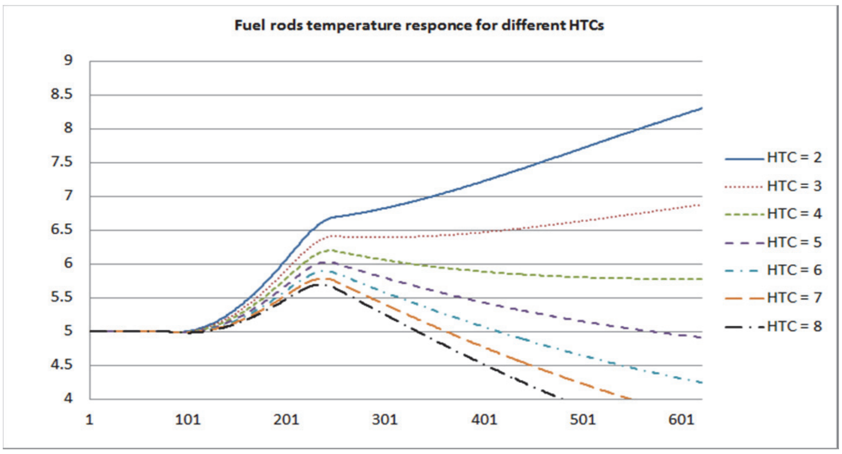


Figure 31, Fuel rod temperature results when different values for the Heat Transfer Coefficient (HTC) design parameter are used. Temperature values above 5.5 are considered not healthy by the Functional Failure Logic (Fig. 24) related to the transmit thermal energy function (Fig. 19) [103]

6.2.6 Multiple simulation runs with combinations of simulation parameter values

Previously, the effect of varying a single parameter has been studied. In section 6.2.4, the timing parameters of the critical event scenario were changed, and the effect on occurrence of functional failures was explored (Fig. 29). In section 6.2.5, a single component parameter was used as a variable in order to establish the effect of that parameter to the system behavior (Fig. 31). In this section, a methodology for systematically studying several parameters is presented.

This section presents an algorithm for studying combinations of parameter values based on the following inputs: a list of parameter names, their initial and final values and the value increment steps. The flow chart of that algorithm is presented in Fig. 32. This algorithm refines the one presented in Fig. 16 and it is closer to the implementation in a programming language. Recursion is used as a programming technique because the algorithm does not need to know the length of the parameter list and to improve readability and compactness of code [104, 105]. A tool was developed to read this input and to produce a Matlab script that can run the simulation for all possible combinations and to export the results to a spreadsheet.

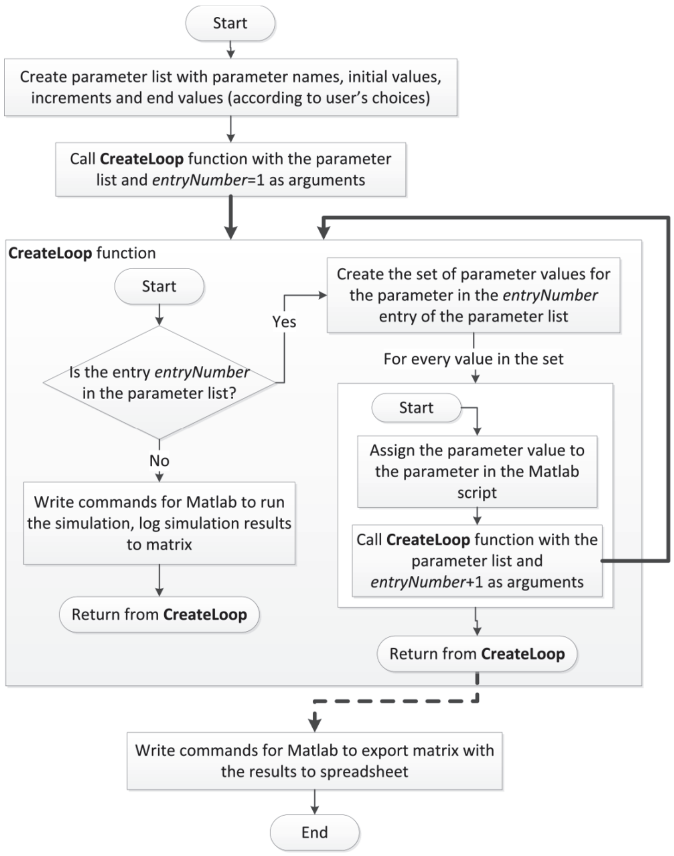

Start
Create parameter list with parameter names, initial values, increments and end values (according to user's choices)
Call **CreateLoop** function with the parameter list and *entryNumber*=1 as arguments
CreateLoop function
Start
Is the entry *entryNumber* in the parameter list?
Yes
No
Create the set of parameter values for the parameter in the *entryNumber* entry of the parameter list
For every value in the set
Start
Assign the parameter value to the parameter in the Matlab script
Call **CreateLoop** function with the parameter list and *entryNumber*+1 as arguments
Write commands for Matlab to run the simulation, log simulation results to matrix
Return from **CreateLoop**
Return from **CreateLoop**
Write commands for Matlab to export matrix with the results to spreadsheet
End

Figure 32, Algorithm for generating the parameter combinations for executing FFIP simulation runs, a refinement of the algorithm in Fig. 16.

In the initial critical event scenario (see Table 3), the emergency power for the coolant pumps was cut off before the pressure shockwave occurred. To demonstrate the algorithm in Fig. 32, three different values are selected for the HTC parameter (see section 6.2.5), the slope of the ramp for driving the coolant pumps to minimum rotations per minute (RampSlope) and the time delay between the unintentional pressure control valve malfunction and power rail failure (RailFailDelay). The HTC will be given values of [1, 3, 5] in separate simulations, the RailFailDelay will be one of the [0, 40, 80] simulation time periods and the RampSlope is one of: [-0.01, -0.015, -0.02]. The medium value is the designer's best estimation of a realistic design parameter, while the low and high limits are given the most extreme values that are estimated to be feasible. This will result in 27 fuel rod temperature curves, one for each combination of parameter values. The selected function of interest is thermal energy transfer from fuel rods, so the Excel output for each simulation run includes the lowest functional health status of this function during the simulation. The curves for fuel rod temperature can be grouped by the health status of the transmit thermal energy function. Temperature values above 5.5 lead to degraded health of the transmit thermal energy function (see the FFL logic in Fig. 24). The results of the simulation runs, during which the transmit thermal energy function's health was healthy throughout the simulation, are presented in Fig. 33.

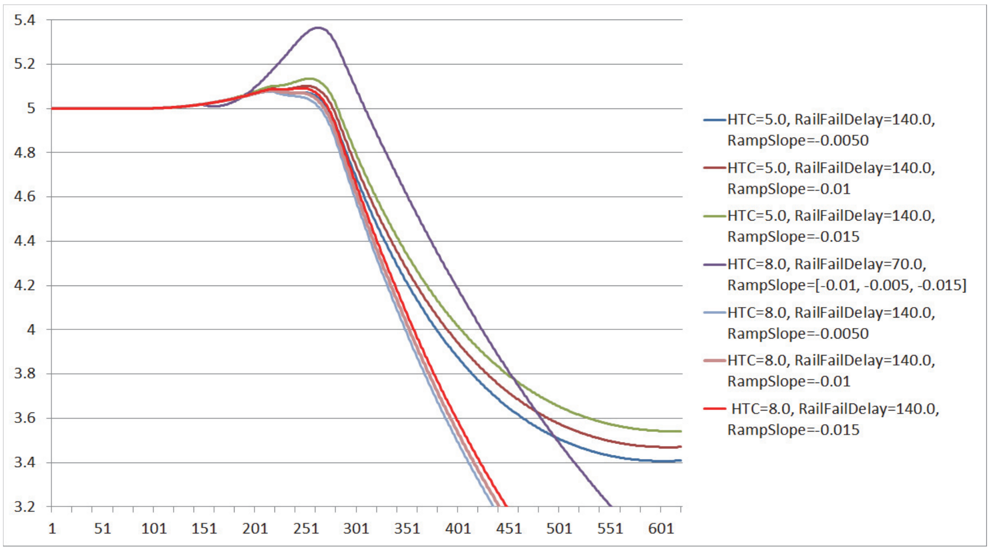


Figure 33, Temperature of fuel rods in simulations resulting in healthy FFL verdicts. The three curves for HTC=8 and RailFailDelay=70 (for the three values of the RampSlope parameter) were overlapping and visually indistinguishable on the graph, so the parameter value combinations corresponding to them are shown in square brackets in the legend.

All the simulation runs in which the time delay between the pressure control valve malfunction and power rail failure is zero (RailFailDelay=0) resulted in unhealthy verdicts for the transmit thermal energy function. No emergency power is available for the coolant pumps and the external power rail is lost, so the coolant pumps cannot follow the ramp to minimum Rotations Per Minute (RPM). In this case, the ramp slope parameter is not relevant, and there is no HTC parameter value with which the fuel rod’s temperature stays within the healthy limit.

The results in Fig. 33 provide the following constraints to the designer. If the time delay between the valve malfunction and power rail failure is 70 (RailFailDelay=70), then only the combinations that include the maximum HTC (HTC=8) lead to fuel rod temperatures that are safe regardless of the ramp slope. If the time delay between the valve malfunction and power rail failure is 140 (RailFailDelay=140), all combinations of parameter values result to a healthy transmit thermal energy function, with the HTC parameter being more influential than the slope of the ramp.

6.3 Evaluating the safety of design alternatives by failure propagation analyses

Section 6.2 presented the simulation results for combinations of values for component parameters and critical event timing parameters. That approach is restricted to configuration alternatives that can be expressed by varying values of parameters, so it cannot cover more fundamental design differences such as different types of control algorithms or the existence of sensors or connections [45, 106].

In this section the FFIP risk assessment method is applied to a set of design alternatives of the BWR case (see section 4.2). By following the flow chart of Fig. 2, the next step after the creation of the functional model of the system under study (this step has been presented in section 6.1) is the development of the feature model (see section 5.1 for more information about creating a feature model) that contains the features that define the design alternatives (Fig. 2, step 3).

For the BWR case study the option of providing pressure information to the turbine protection system is added, as it is described in section 4.2 (see Fig 6). Additionally different variations for the algorithm for dropping the coolant pumps' RPM to a minimal level in case of emergency are introduced as design alternatives. Apart from ramping to minimum RPM, the alternatives introduced are stepping to minimum RPM and reaching minimum RPM using a decay function for the pump motors. The feature model is presented in Fig. 34. The coolant pump control and the turbine protection subsystems are mandatory features for all possible configurations. While the dropping to minimum RPM is a mandatory feature, there are three alternative algorithms to perform that action, the choices are a step, a ramp or a decay function. The connection to a pressure sensor is optional and there are two alternative designs, the pressure information can be read from the sensor in the reactor vessel or a sensor can be added at the pipeline before the pressure control valve (see Fig. 6). At this phase there are no constraints in the feature model.

All the possible alternative designs of the BWR case study are evaluated using the FFIP method against the critical event scenario presented in section 6.2.1 The results of the simulation determine the acceptable configurations and the feature model is updated so that it represents only these configurations. After this step the feature model can continue to the latter phases of the design. The workflow for this methodology is presented in Fig. 35.

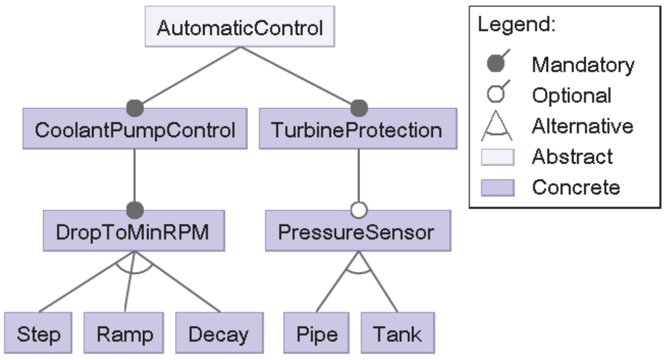


Figure 34, Feature model describing the design alternatives of the Boiling Water Reactor case [106]

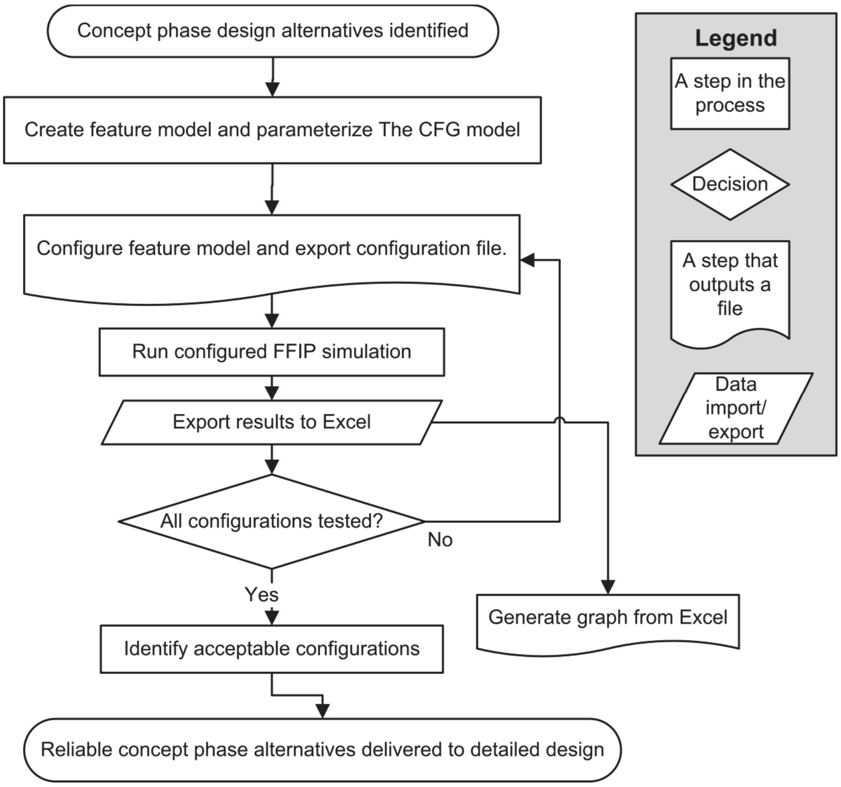

Concept phase design alternatives identified
Create feature model and parameterize The CFG model
Configure feature model and export configuration file.
Run configured FFIP simulation
Export results to Excel
All configurations tested?
No
Yes
Identify acceptable configurations
Generate graph from Excel
Reliable concept phase alternatives delivered to detailed design
Legend
A step in the process
Decision
A step that outputs a file
Data import/ export

Figure 35, Flowchart describing the identification of acceptable design alternatives, adapted from [106]

The CFG model includes the different behavior models that support the alternative features of the feature model. The data flow during the simulation is controlled by configuration signals. The coolant pump behavior models that correspond to the alternative algorithms for dropping to minimum RPM are presented in Fig. 36. The boolean signals RampConf, StepConf and DecayConf determine which behavior is going to be included in the simulation.

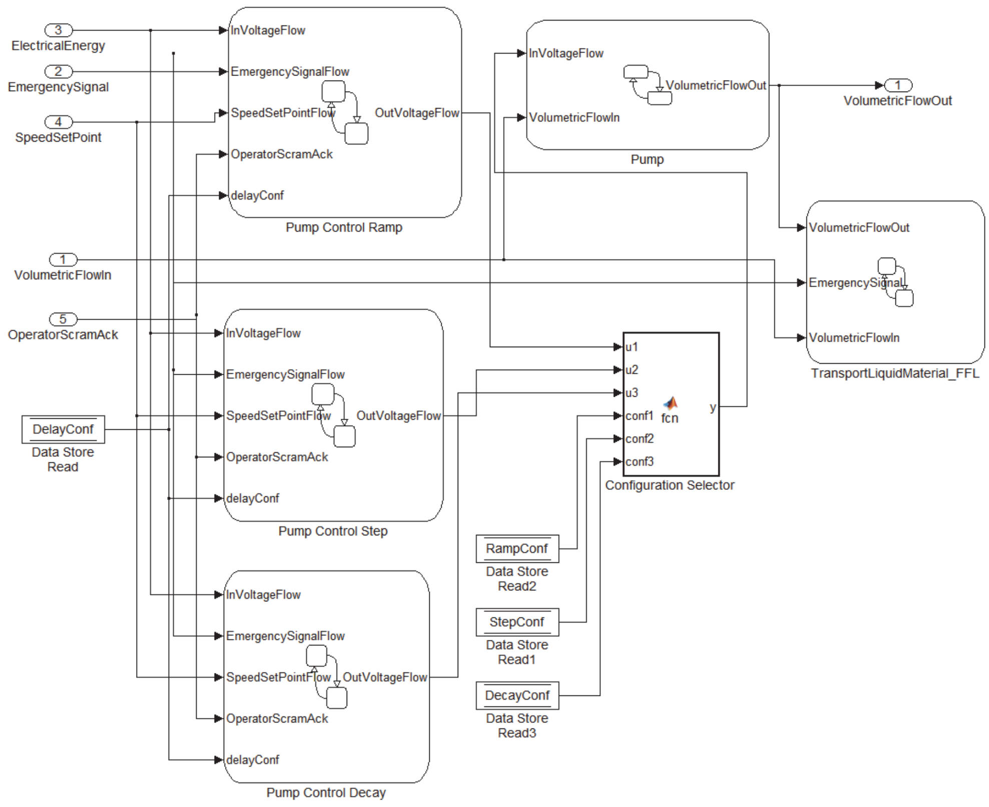
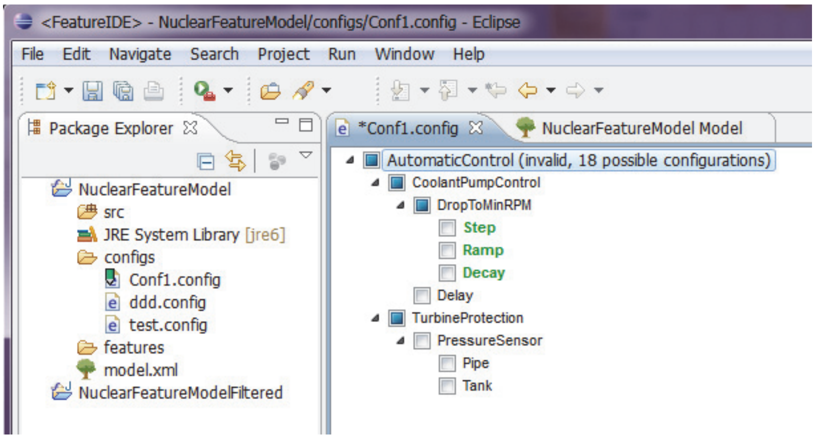


Figure 36, The internals of the CoolantPumps component of Fig. 22. Alternative behavioural models have been added to support the different emergency behaviour algorithms presented in Fig. 35 [106]

The feature model in Fig. 34 was created using the FeatureIDE tool [69]. This tool contains a user-friendly configurator (Fig. 37) that helps the user configure a product instance. Since the FeatureIDE is an open-source software, it was possible to extend the configurator (see Fig. 38) in order to create a Matlab script (see Fig. 39) that configures the CFG model from a valid product configuration. Every possible valid configuration was created and was assessed with the FFIP method.

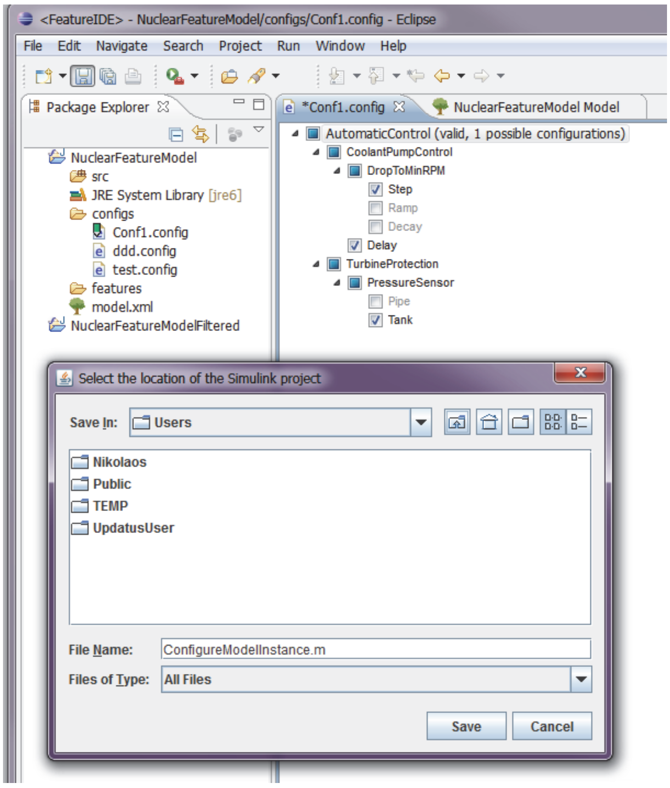

<FeatureIDE> - NuclearFeatureModel/configs/Conf1.config - Eclipse
File Edit Navigate Search Project Run Window Help
Package Explorer
*Conf1.config
NuclearFeatureModel Model

NuclearFeatureModel
- src
- JRE System Library [jre6]
- configs
  - Conf1.config
  - ddd.config
  - test.config
- features
- model.xml

NuclearFeatureModelFiltered

AutomaticControl (invalid, 18 possible configurations)
- CoolantPumpControl
  - DropToMinRPM
    - Step
    - Ramp
    - Decay
  - Delay
- TurbineProtection
  - PressureSensor
    - Pipe
    - Tank

Figure 37, Configuring a valid product instance using FeatureIDE's configurator


<FeatureIDE> - NuclearFeatureModel/configs/Conf1.config - Eclipse
File Edit Navigate Search Project Run Window Help
Package Explorer
*Conf1.config
NuclearFeatureModel Model
Select the location of the Simulink project
Save In:
Users
Save
Cancel

NuclearFeatureModel
- src
- JRE System Library [jre6]
- configs
  - Conf1.config
  - ddd.config
  - test.config
- features
- model.xml

NuclearFeatureModelFiltered

AutomaticControl (valid, 1 possible configurations)
- CoolantPumpControl
  - DropToMinRPM
    - ✓ Step
    - Ramp
    - Decay
  - ✓ Delay
- TurbineProtection
  - PressureSensor
    - Pipe
    - ✓ Tank

Nikolaos
Public
TEMP
UpdatusUser

File Name: ConfigureModelInstance.m
Files of Type: All Files

Figure 38, Extension developed for FeatureIDE's configurator for the exports of a Matlab script (Fig, 39) that can configure the Configuration Flow Graph model in Simulink (Fig. 36) [106]

```
% Initializing configuration signals

StepConf.InitialValue='true';
DelayConf.InitialValue='true';
PressureSensorConf.InitialValue='true';
TankConf.InitialValue='true';

% Run the simulation

gatherDataSPL
```

Figure 39, Matlab script that configures the Configuration Flow Graph model in Simulink (Fig. 36) [106]

The Function Failure Logic (FFL) indicates that for some configurations of design alternatives, there is health degradation of the function "transmit thermal energy from fuel rods to coolant" (see the functional model of the BWR in Fig. 19). The FFIP risk assessment of each individual configuration is done in a similar process as in section 6.2.1. Fig. 40 shows the temperature curves for each valid configuration of the feature model in Fig. 34; as there are 9 valid configurations, there are nine curves.

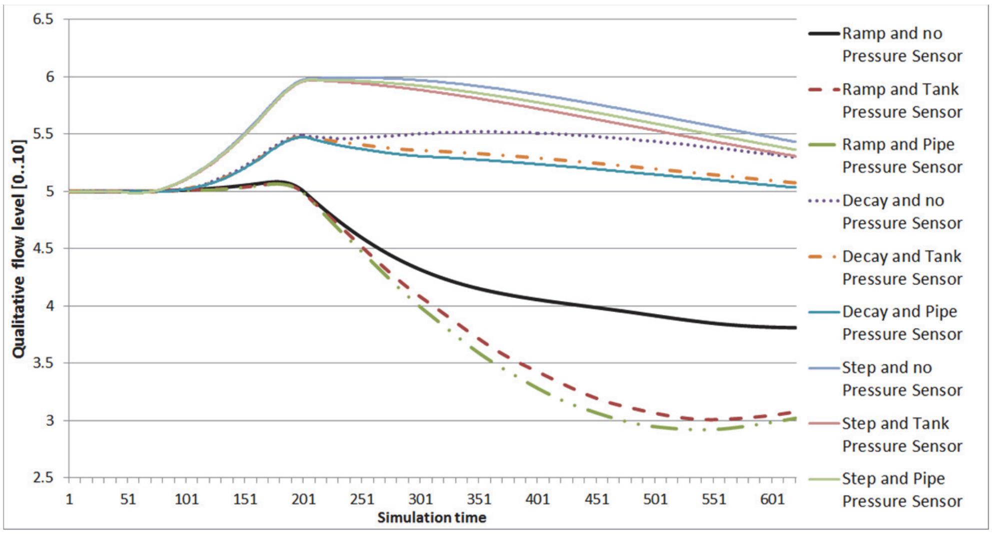


Figure 40, Fuel rod temperature results for different alternative design configurations. Temperature values above 5.5 are considered not healthy by the Functional Failure Logic (Fig. 24) related to the transmit thermal energy function (Fig. 19) [106]

A summary of the simulation runs for the entire feature model is obtained automatically and presented in Table 4. If the function status remains healthy throughout the simulation, the configuration has passed, since the functional requirements are satisfied. If it is degraded or lost at any time, the configuration has failed and the number of simulation time steps spent in a status other than healthy is indicated in parentheses.

	No Pressure Sensor (PS)	PS on Tank	PS on Pipe
Ramp	pass	pass	pass
Decay	fail (120)	pass	pass
Step	fail (435)	fail (364)	fail (395)

Table 4, Results of the Functional Failure Logic (Fig. 24) for the results presented in Fig, 40. In parenthesis is the total simulation time that the transmit thermal energy function's health status is "degraded" [106]

Fig. 41 repeats the temperature curve from Fig. 40 for the configuration decay and no pressure sensor connected to the turbine protection system. The decay feature involves an algorithm that reduces the control signal to the pumps by a logarithmic decay, and the volumetric flow of the coolant in Fig. 41 follows this pattern. The steam content drops abruptly as a result of the shockwave and then rises as coolant flow decreases and heat production due to increased neutron flux increases; insertion of the control rods eventually brings the neutron flux down to a state. The temperature exceeds acceptable level, so the verdict of this simulation run is failed.

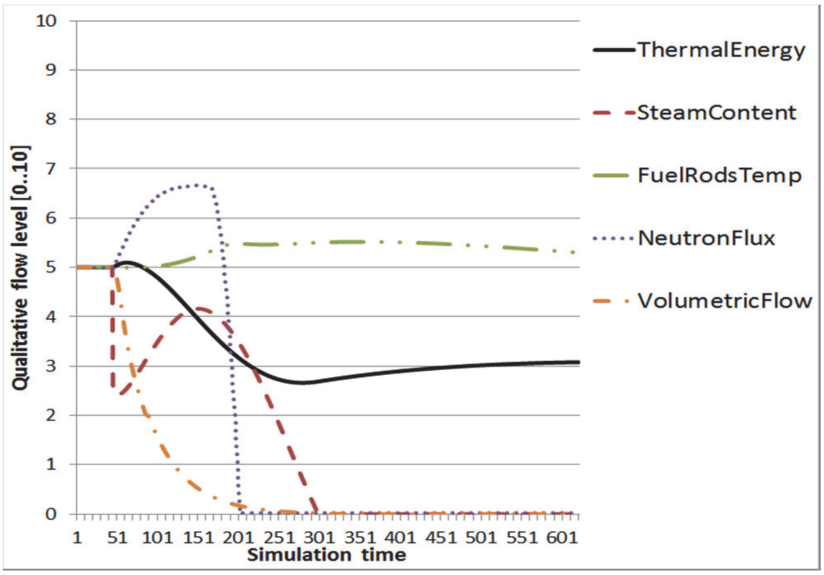


Figure 41, The decay algorithm for the coolant pumps behaviour is rejected if pressure sensor information is not available for the turbine protection system [106]

Fig. 42 shows a similar scenario, but in this case the pressure sensor inside the reactor tank is connected to the Turbine Protection System (TPS). As soon as pressure exceeds an alarm threshold, the TPS opens the dumper valve which relieves the pressure inside the tank, resulting in a sharp rise of steam content at the core. After opening the dumper valve, the software in the TPS waits for a delay to expire and then closes the valve as soon as pressure is nominal or lower. The purpose of the delay is to prevent very frequent opening and closing of the dumper valve, which would otherwise occur with basic binary control. The three spikes in the steam content curve in Fig. 42 are due to the dumper valve being opened 3 times in this scenario.

In Fig. 42, the temperature stays within acceptable levels in this case, and the test is passed. As shown in Fig. 40, the temperature decrease is even slightly better if the additional sensor in the pipeline is used, since this permits the earlier opening of the dumper valve. In summary, the decay feature gave acceptable performance if the pressure sensor option was present, regardless of which alternative (pipe or tank) was selected. However, performance was evaluated unreliable if no pressure sensor was present. This can be summarized as a constraint: the decay feature requires the pressure sensor feature to be also selected. In the feature model syntax, this is written as follows:

Decay => PressureSensor

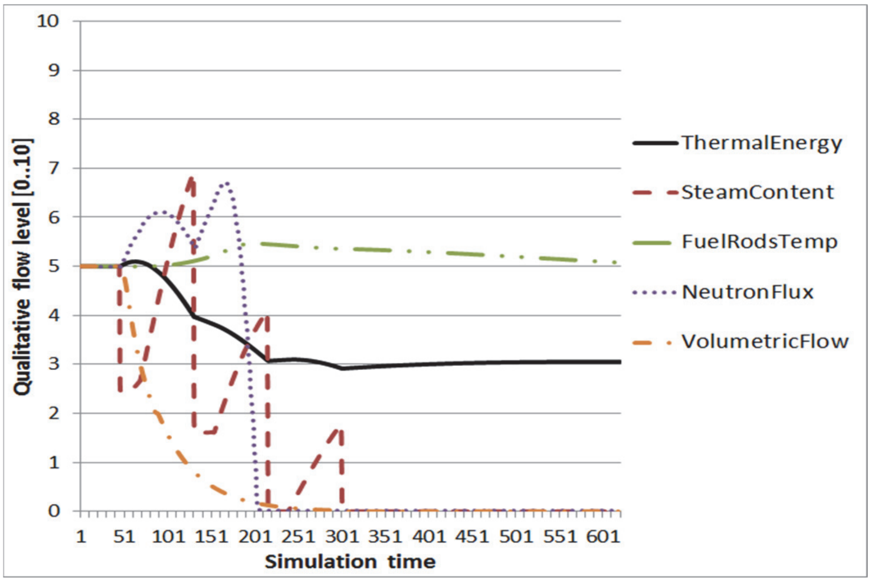


Figure 42, The presence of pressure sensor information leads to acceptable system response to the critical event scenario (see the functional health results in Table 4) for the configurations including the Decay feature. These are the results when the pressure information comes from the reactor vessel. If there is a pressure sensor on the pipeline before the pressure control valve, the results are even better (see Table 4 and Fig. 40) [106]

Fig. 43 shows the results for the step algorithm: the pumps are stopped immediately in response to a scram. In this case, the impaired moderation due to faster steam buildup causes neutron flux to rise significantly slower than when the decay algorithm was used (Fig. 41). However, since water acts as coolant as well as moderator, the loss of volumetric flow of coolant also results in a lower thermal energy curve (i.e., thermal energy output flow from fuel rods to coolant). The combined effect of these conflicting forces is that fuel rod temperature rises to more dangerous levels with the step algorithm rather than the ramp algorithm. Even the use of a pressure sensor by the turbine protection system for opening the dumper valve does not help to reduce the temperature to within healthy limits in this failure scenario (Fig. 44). For these reasons, the step alternative will be entirely removed from the feature model that is passed to detailed design from the FFIP risk assessment. Determining the net effect of these conflicting forces is problematic if traditional risk analysis methods relying only on human judgment are used; this case study strengthens the argument for simulation based methods for the risk assessment of complex cyber-physical systems.

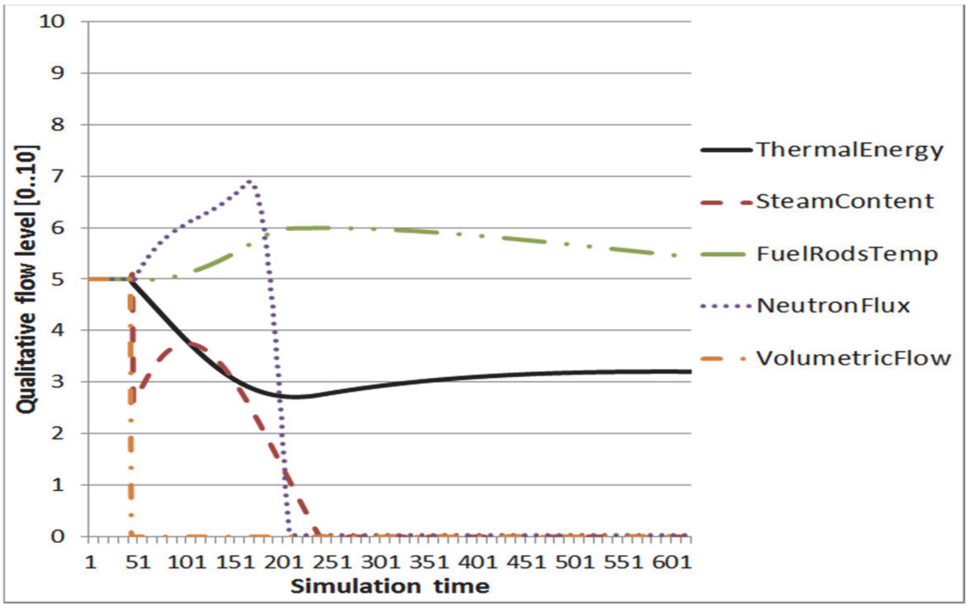


Figure 43, The Step algorithm for the coolant pumps behaviour fails if there is no pressure sensor information [106]

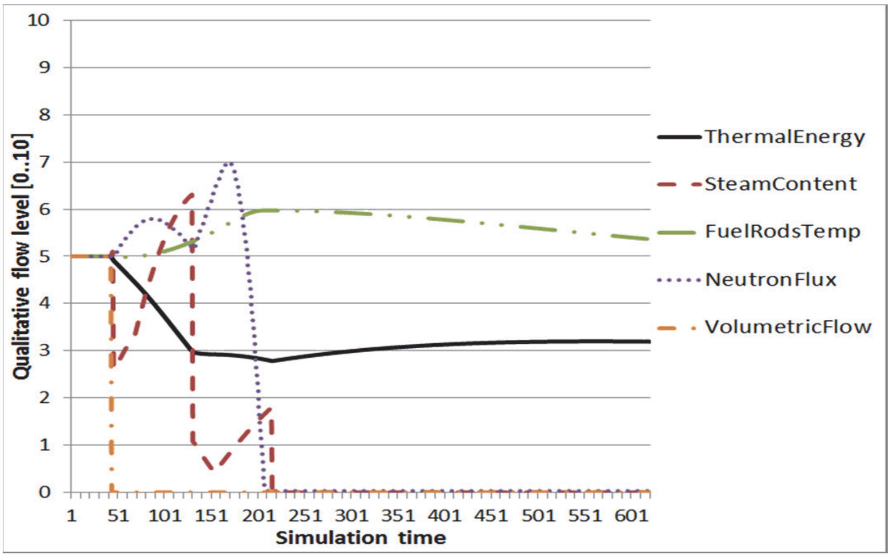


Figure 44, The inclusion of pressure sensor information cannot lead to acceptable system response to the critical event scenario. This leads to the removal of the Step feature from the design alternatives in Fig. 34 [106]

Finally, the ramp algorithm employs a linear ramp to reduce the coolant pump rotations per minute. This alternative gave healthy results for the “thermal energy transmission from fuel rods” function regardless of what options were selected for the pressure sensor (see Fig. 40 and Table 4). The restricted feature model is shown in Fig. 45; the feature combinations that were found unreliable in this risk assessment phase have been removed. The constraint “Decay => PressureSensor” has been specified using the FeatureIDE tool and can be used by the FeatureIDE configurator (Fig. 38) to determine valid configurations.

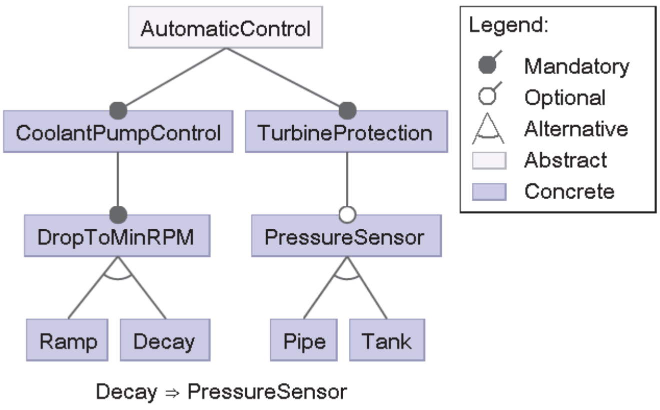


Figure 45, Feature model after the risk assessment of the design alternatives. Compared to Fig. 34 the Step feature is removed and the constraint that the Decay feature requires the presence of pressure information from either the reactor vessel or the pipeline before the pressure control valve [106]

6.4 Configuring machine control software based on a feature model of safe design alternatives

The methodology up to the previous subsection delivers a feature model of the system, from which unsafe features and feature combinations have been removed (step 3 in Fig. 2). This is given as input to the SPL development process, which first strips away the features that do not involve software (step 4 in Fig. 2). The resulting feature model is processed in step 5, in order to configure products. As mentioned in section 5.3, the case study changes at this point. This is considered acceptable, since the interface between steps 3, 4 and 5 are only feature model deliverables, without additional dependencies to other artifacts produced in steps 1-3.

In this section the configuration of software instances for the MEWP product family case study (see section 4.1) will be presented. Valid software product instances obtained by configuring the feature model of the MEWP family from section 5.1 are used to configure PLC software projects. Object Oriented (OO) extensions of IEC 61131 are used to realize the optional and alternative constructs as described in [55, 84].

6.4.1 Feature model to UML mapping and implementation using Object Oriented IEC 61131 extensions

The constructs provided by the UML class diagram [107] are used to specify the mapping from the feature model to an object oriented language [55, 84]. These mappings (Fig. 46) exploit composition and inheritance, and have been defined to enable simple and automatic configuration of product code after a user has selected the features for a product.

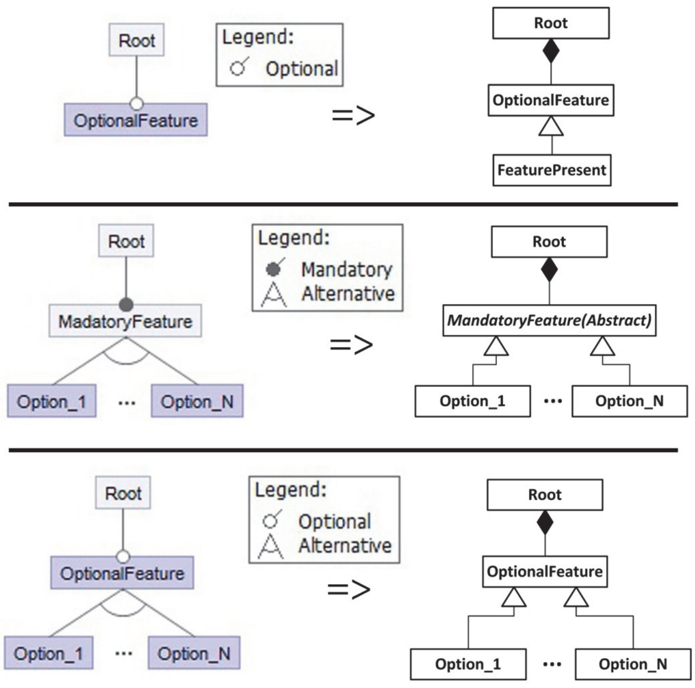


Figure 46, Mappings from feature model notation to UML [55]

By following the mapping rules in Fig. 46, the feature model in Fig. 7 will be converted into the object oriented structure in fig. 47. After this UML model is created, it is possible to develop the generic control application, without directly specifying optional or alternative features, in an Integrated Development Environment (IDE) that supports the object oriented extensions of IEC 61131-3. In this case study, CoDeSys 3.4 and the Structured Text (ST) language was used.

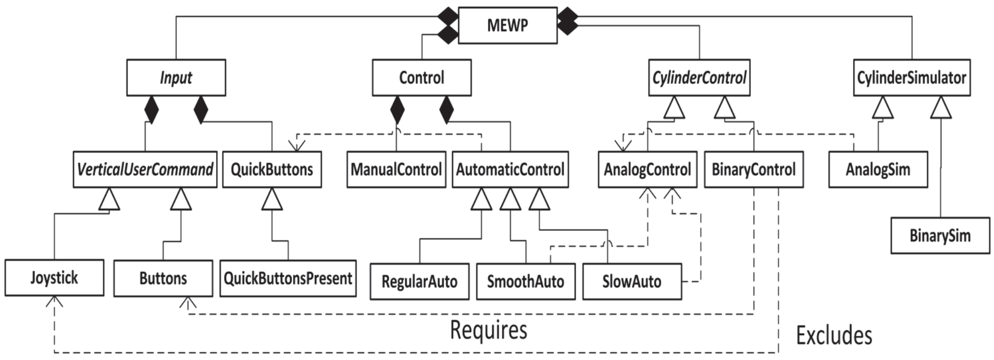
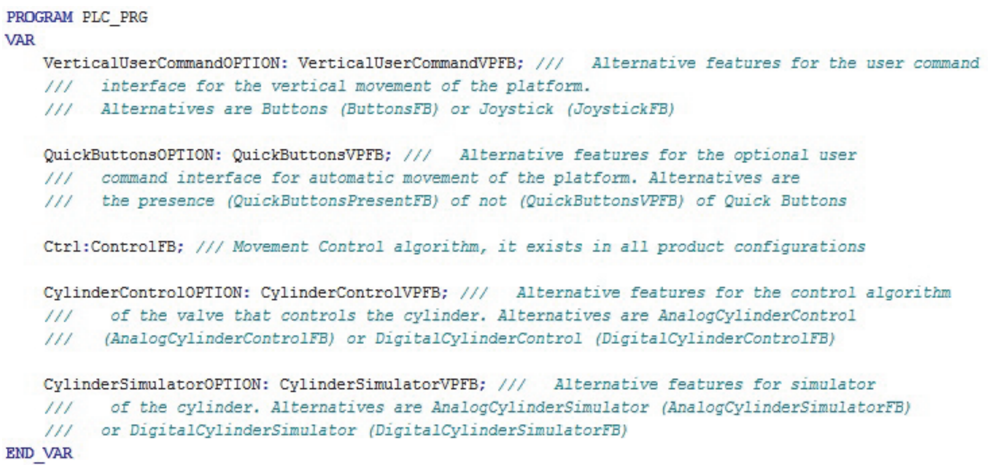


Figure 47, UML model of the software product family, after applying the mappings in Fig. 46 to the feature model in Fig. 7 [55]

The technique to create the PLC software relies on using superclasses as placeholders for optional features and the parents of alternative features, as presented in the ST code presented in Fig 48.



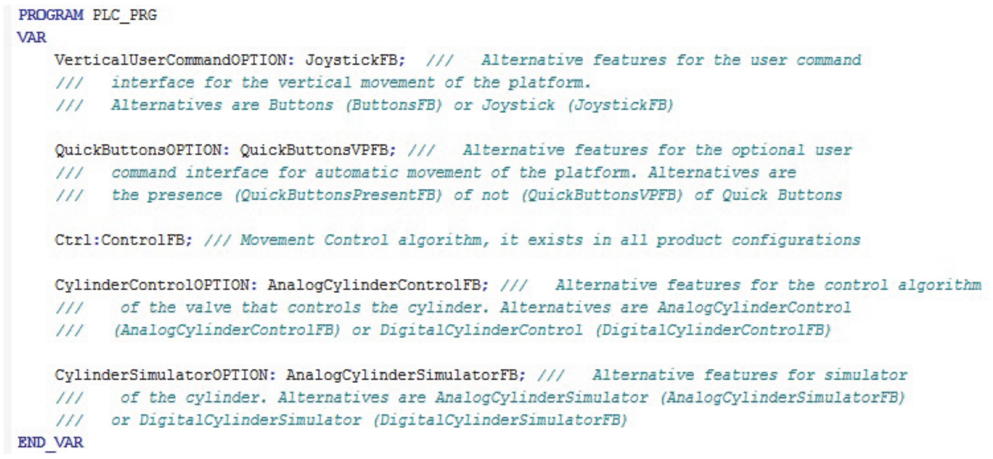
```
PROGRAM PLC_PRG
VAR
    VerticalUserCommandOPTION: VerticalUserCommandVPFB; ///   Alternative features for the user command
    ///   interface for the vertical movement of the platform.
    ///   Alternatives are Buttons (ButtonsFB) or Joystick (JoystickFB)

    QuickButtonsOPTION: QuickButtonsVPFB; ///   Alternative features for the optional user
    ///   command interface for automatic movement of the platform. Alternatives are
    ///   the presence (QuickButtonsPresentFB) of not (QuickButtonsVPFB) of Quick Buttons

    Ctrl:ControlFB; /// Movement Control algorithm, it exists in all product configurations

    CylinderControlOPTION: CylinderControlVPFB; ///   Alternative features for the control algorithm
    ///    of the valve that controls the cylinder. Alternatives are AnalogCylinderControl
    ///   (AnalogCylinderControlFB) or DigitalCylinderControl (DigitalCylinderControlFB)

    CylinderSimulatorOPTION: CylinderSimulatorVPFB; ///   Alternative features for simulator
    ///    of the cylinder. Alternatives are AnalogCylinderSimulator (AnalogCylinderSimulatorFB)
    ///   or DigitalCylinderSimulator (DigitalCylinderSimulatorFB)
END_VAR
```

Figure 48, Unconfigured Structured Text code in CoDeSys for the main Program Organization Unit of the control application of the Mobile Elevating Work Platform case

During product configuration, the user specifies alternatives; for example for VerticalUserCommand, the Joystick alternative is selected. The product configurator tool, which will be described in section 6.4.3, is able to automatically replace the type of the corresponding FB instance from VerticalUserCommand to Joystick. Since the latter is a derived class of the former, no changes are needed to the rest of the program.

For optional features such as QuickButtons, a superclass function block that is neutral to the program (either has no logic or it copies its inputs to its outputs) is created and then it is extended by a function block(s) that implement(s) the optional feature (QuickButtonsPresent in our example). If during product configuration this feature is selected, the configurator tool will replace the type definition from QuickButtons to QuickButtonsPresent.

The product configurator tool will configure the code in Fig 48 into the code in Fig. 49, if the Joystick, AnalogControl and AnalogSim features are selected.



```
PROGRAM PLC_PRG
VAR
    VerticalUserCommandOPTION: JoystickFB;  ///   Alternative features for the user command
    ///   interface for the vertical movement of the platform.
    ///   Alternatives are Buttons (ButtonsFB) or Joystick (JoystickFB)

    QuickButtonsOPTION: QuickButtonsVPFB; ///   Alternative features for the optional user
    ///   command interface for automatic movement of the platform. Alternatives are
    ///   the presence (QuickButtonsPresentFB) of not (QuickButtonsVPFB) of Quick Buttons

    Ctrl:ControlFB; /// Movement Control algorithm, it exists in all product configurations

    CylinderControlOPTION: AnalogCylinderControlFB; ///   Alternative features for the control algorithm
    ///    of the valve that controls the cylinder. Alternatives are AnalogCylinderControl
    ///   (AnalogCylinderControlFB) or DigitalCylinderControl (DigitalCylinderControlFB)

    CylinderSimulatorOPTION: AnalogCylinderSimulatorFB; ///   Alternative features for simulator
    ///    of the cylinder. Alternatives are AnalogCylinderSimulator (AnalogCylinderSimulatorFB)
    ///   or DigitalCylinderSimulator (DigitalCylinderSimulatorFB)
END_VAR
```

Figure 49, The code of Fig. 49 after it has been configured to a specific, valid, software product instance

According to Fig. 46, if a mandatory feature (e.g. VerticalUserCommand) has alternative subfeatures, the mandatory feature is mapped to an abstract class. This is because one of the subfeatures must be selected. If an optional feature has alternative subfeatures (e.g. AutomaticControl) it is not abstract but has the above mentioned neutral implementation, which will be included into the final product in case the option was not selected.

In summary, all modifications that need to be made by the product configurator tool will be limited to the declaration part of a POU (Program Organization Unit); the code is not changed.

6.4.2 Extension to the PLCopen XML standard

The PLCopen XML standard for storing IEC 61131 applications has the flexibility to be extended so to include feature modeling information [55]. The <Constrains> element, a child of the <data> element (which is a child of the <addData> element), is proposed as a container to store this information. An attribute of the <Constrains> element ("isOptional") stores a Boolean of whether a feature is optional. The information of whether other features are required or excluded if that feature is selected can be stored by using the <requires> and <excludes> child elements of the <Constrains> element. The schema of the <Constrains> element can be the following:

```
<xs:element name="Constrains">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Requires" type="xs:string"/>
      <xs:element name="Excludes" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="isOptional" type="xs:boolean"/>
  </xs:complexType>
</xs:element>
```

The schema is not used to specify alternative features, because the class inheritance mechanism fits these semantics: it is not possible to select more than one derived class, since object-oriented extensions of IEC 61131-3 do not permit multiple inheritance [6].

The <Constrains> element stores rules to prevent invalid feature combinations. Consider the optional "AutomaticControl" feature, which requires the "QuickButtons" feature. This information is expressed according to the schema as follows:

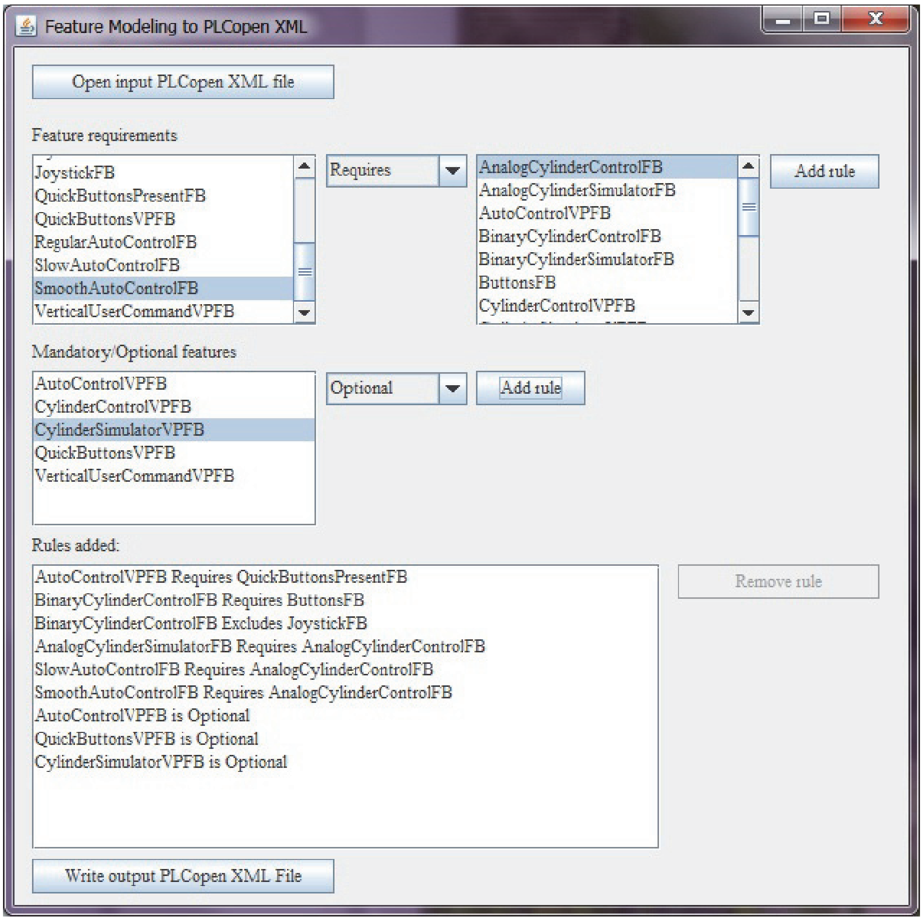
```
<pou name="AutoControlVPFB" pouType="functionBlock">
 <interface>
  ...
  <addData>
   <data name="Feature Modeling data">
    <Constrains isOptional="true">
     <Requires>QuickButtonsPresentFB</Requires>
    </Constrains>
   </data>
  </addData>
  ...
 </interface>
 ...
</pou>
```

The <data> element under the <addData> is already used by CoDeSys to support object oriented extensions, and with our additions the PLCopen file can still be imported. However, standardization will be required for portability if several players will support feature modeling with IEC 61131-3.

6.4.3 Tools to support the IEC61131 PLCopen-based SPL

6.4.3.1 Embedding feature constrains in PLCopen XML

As a proof-of-concept, a tool was developed to guide the creation of constraints for a PLCopen XML file describing a product line with mandatory, optional and alternative features (Fig. 50) [55]. The tool reads a PLCopen xml file with the function block library of the project, and after analyzing the inheritance keywords, it presents the alternative and optional features. After this, it is possible to specify "requires" and "excludes"rules between any two features, which are stored in an "enhanced" PLCopen XML file, according to the schema presented in section 6.4.2.


Feature Modeling to PLCopen XML
Open input PLCopen XML file
Feature requirements
Requires
Add rule
Mandatory/Optional features
Optional
Add rule
Rules added:
Remove rule
Write output PLCopen XML File

JoystickFB
QuickButtonsPresentFB
QuickButtonsVPFB
RegularAutoControlFB
SlowAutoControlFB
SmoothAutoControlFB
VerticalUserCommandVPFB

AnalogCylinderControlFB
AnalogCylinderSimulatorFB
AutoControlVPFB
BinaryCylinderControlFB
BinaryCylinderSimulatorFB
ButtonsFB
CylinderControlVPFB

AutoControlVPFB
CylinderControlVPFB
CylinderSimulatorVPFB
QuickButtonsVPFB
VerticalUserCommandVPFB

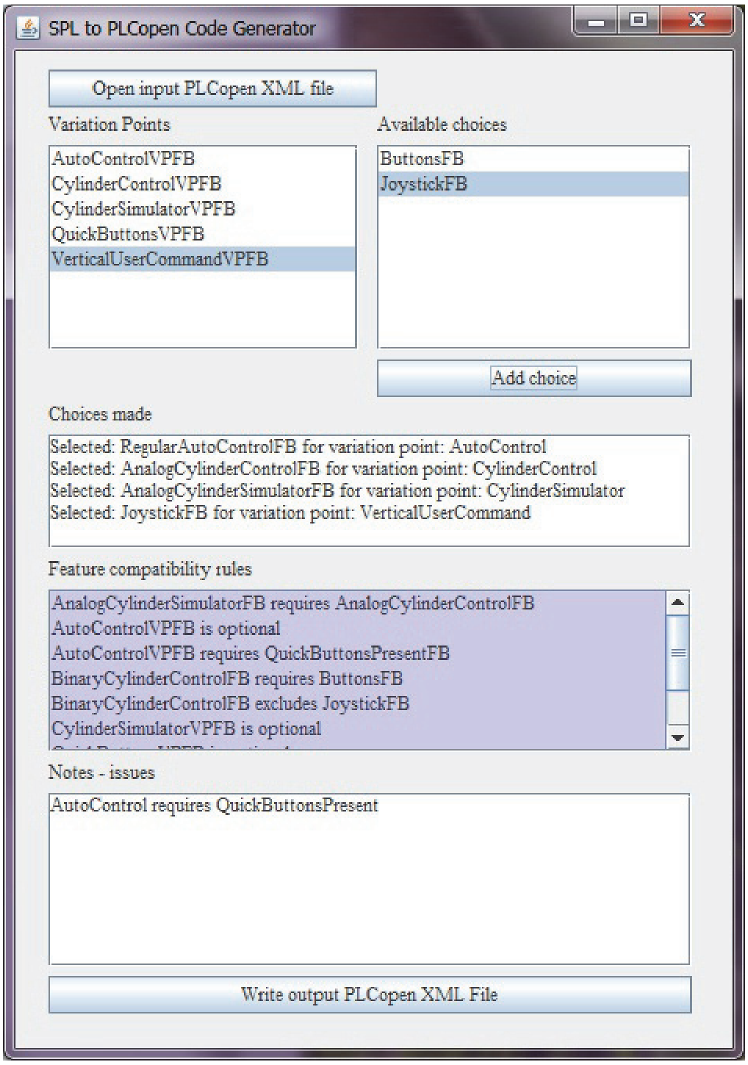
AutoControlVPFB Requires QuickButtonsPresentFB
BinaryCylinderControlFB Requires ButtonsFB
BinaryCylinderControlFB Excludes JoystickFB
AnalogCylinderSimulatorFB Requires AnalogCylinderControlFB
SlowAutoControlFB Requires AnalogCylinderControlFB
SmoothAutoControlFB Requires AnalogCylinderControlFB
AutoControlVPFB is Optional
QuickButtonsVPFB is Optional
CylinderSimulatorVPFB is Optional

Figure 50, A tool that was developed to add the constraints information of the feature model in Fig. 7 into an "enhanced" PLCopen XML file [55]

6.4.3.2 Configuring and validating products

A second tool was developed to help the product configuration, perform validation of the choices and then produce the configured PLCopen xml file with the final product code [55].

The tool accepts as input an enhanced PLCopen xml file containing the generic IEC 61131 application and the constraints of the feature model (see section 6.4.2). The user can select which optional features are included, and for each alternative feature, one choice from a list of alternatives must be made (Fig. 51).


SPL to PLCopen Code Generator

Variation Points

AutoControlVPFB
CylinderControlVPFB
CylinderSimulatorVPFB
QuickButtonsVPFB
VerticalUserCommandVPFB

Available choices

ButtonsFB
JoystickFB

Choices made

Selected: RegularAutoControlFB for variation point: AutoControl
Selected: AnalogCylinderControlFB for variation point: CylinderControl
Selected: AnalogCylinderSimulatorFB for variation point: CylinderSimulator
Selected: JoystickFB for variation point: VerticalUserCommand

Feature compatibility rules

AnalogCylinderSimulatorFB requires AnalogCylinderControlFB
AutoControlVPFB is optional
AutoControlVPFB requires QuickButtonsPresentFB
BinaryCylinderControlFB requires ButtonsFB
BinaryCylinderControlFB excludes JoystickFB
CylinderSimulatorVPFB is optional

Notes - issues

AutoControl requires QuickButtonsPresent

Figure 51, A tool that was developed to help the user to configure a software product instance, validate it against the feature model, and export a configured PLCopen XML project file [55]

After all the choices are made, the tool runs the validation algorithm in Fig. 52 to verify that no rules are broken by the configuration and then it either produces the PLCopen xml file with the configured product, or it presents feedback to the user about which rules were broken by the selected configuration.

After the configured PLCopen XML file is produced, it is ready to be imported in a PLCopen XML IEC 61131 compliant tool such as CoDeSys 3.4. It only needs to be built in order to obtain the executable application.

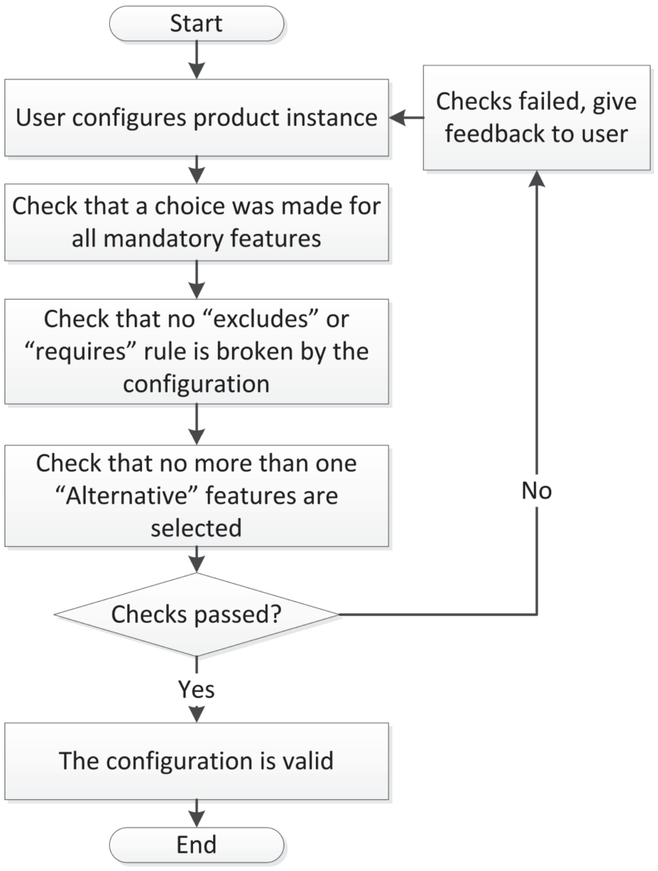

Start
User configures product instance
Check that a choice was made for all mandatory features
Check that no “excludes” or “requires” rule is broken by the configuration
Check that no more than one “Alternative” features are selected
Checks passed?
No
Checks failed, give feedback to user
Yes
The configuration is valid
End

Figure 52, Flowchart of the algorithm used in the tool in Fig. 51, that validates the user choices against the feature model of the Software Product Line, adapted from [55]

7 Discussion

7.1 Discussion related to objective 1

According to section 3, objective 1 of this dissertation is that "*the modeling approach should support filtering designs based on results of risk assessment".*

The output of the method presented in the results section lists combinations of features that need to be eliminated. Feature modeling with constraints provides the appropriate level of granularity to eliminate these combinations and no other combinations. Currently this filtering process is done manually. Further research would involve automating the filtering process by

1. reading the design alternatives subjected to risk assessment as a feature model
2. obtain FFIP output for unsafe feature combinations
3. define an algorithm that prunes the feature model (a tree data structure) created in step 1 to eliminate the combinations identified in step 2

The scalability of the method requires automation of step 2 to systematically evaluate every feature combination. In this dissertation, step 2 was done manually, and automating it is further research.

7.2 Discussion related to objective 2

According to section 3, objective 2 of this dissertation is that *"The behavioral simulation in previous work on FFIP does not support any kind of study of several positive or negative feedback loops affecting the same process variable. The discrete qualitative enumeration for flow values [zero, low, nominal, high] is insufficient to capture, even in a qualitative way, how several feedback mechanisms reinforce or oppose each other. An objective of this work is to propose a continuous range of values to qualitatively describe flow values, so that simulation models are able to capture how several feedback mechanisms reinforce or oppose each other."*

The results section shows that the definition of qualitative continuous values for flow levels is able to capture the aggregate effect of key phenomena in scram situations; examples of phenomena that are included were:

- improved moderation due to lower void fraction after pressure shockwave
- Doppler effect
- decreased power output due to decreased coolant flow
- decreased heat transfer from fuel rods to coolant due to decreased coolant flow

A weakness is that an absolute value needed to be used for identifying flows that exceed safe thresholds; for example the value 5.5. in Fig. 31. The significance of functional degradation identified according to such criteria is that the hazard mechanism has been identified and that it is a requirement for the detailed design phase to parameterize the design in such a way that the hazard is avoided or that its probability is within acceptable limits. The modeling approach that was used is not based on first principles and cannot be used for purposes of evaluating a single design. However, it can be used for comparative evaluation of alternative designs as discussed under objective 3.

It is possible to use first-principles simulation models based on SI units even in the early phase using generic components, and one area of further research is to transfer the simulation framework that was used in this dissertation to such an environment. The use of a simulation environment that provides a library of generic components, commonly used in the engineering domain of the system under design, would result in quantitative results with physical meaning. In an early design phase, the detailed configuration of components is not yet possible, and for the purpose of reducing the complexity of the model only the most important components should be included.

The methodology presented in this dissertation can be adapted to a more advanced simulation environment; the most important implementation task would be to port the Functional Failure Logic code into the new environment. The qualitative thresholds for the flows, that are the core of the Functional Failure Logic in this dissertation, would have to be updated to realistic threshold values such as temperature, pressure and flow volume thresholds. If the results of the simulation have physical meaning, the risk assessment outcomes would be easier to understand by an expert of the domain, and redesign requirements could be expressed in terms of SI units.

The simulation environment used in this dissertation provides the flexibility of using state machines to define the behavioral model of a component, with one state for each nominal and failure mode of the component. The behavior within a state is defined by a system of linear difference equations. Since difference equations were used, the solver of the simulation environment was able to handle the transition between states with different sets of equations without convergence failures.

The response of physical process components is always continuous over time, and advanced simulators model this behavior. For example, if the control software instructs a closed valve to open, the flow through the valve doesn’t increase instantly by a step function, but follows a curve over time as the valve transitions from allowing no flow to full flow. In a simulator that models separately the control algorithms and the behavior of the process, the software can be described using state machines with instant transitions, while the behavior of the controlled components of the process contains no discontinuities; for example, the opening of a binary valve can be handled in the process simulator by increasing the cross-sectional area of the aperture by a linear ramp.

7.3 Discussion related to objective 3

According to section 3, objective 3 of this dissertation is that *“the modeling approach used in behavioral simulation should support comparative evaluation of the safety of alternative designs or different parameter values for the same design. As the number of alternatives increases, the scalability of the method should be evaluated”.*

The approach based on qualitative flow levels was able to capture how feedback loops opposed or reinforced each other and how alternative designs or parameter value changes caused some configurations to fail according to the criteria of the function failure logic. Further work for obtaining verdicts from functional failure logic defined in terms of SI units was discussed in response to objective 2.

The method and tool presented in this dissertation supports the scalability of the approach for analyzing variations in parameter values. Computation resources form a practical limit to scalability. Resource consumption is increased by increasing the number of parameters being varied, the number of different values used for each parameter and the level of detail of the simulation model. The user of the method is responsible for controlling the factors that affect resource consumption, and the method proposed in this dissertation does not attempt to constrain these decisions. After the simulation model is developed to the desired level of detail, the user interface of the supporting tool provides the user full control of the factors that impact resource consumption.

As the number of valid configurations in a feature model increases, the manual and computational load of performing the method for the product line impacts the scalability of the method. In this dissertation, each feature requires a single modification to the simulation model. However, the number of manual steps in performing the analysis is equal to the number of valid configurations, which depends on the feature model tree. For example, an application with n mandatory features under the root, each of which has m alternatives, has m^n valid configurations if no constraints are present. The number of manual steps in performing the analysis can be reduced or even eliminated by further work on an automatic algorithm that traverses the feature model tree and outputs a list of valid configurations as scripts that configure and execute the simulation model; the functional failure logic results identify the configurations that did not satisfy the functional requirements.

Even if further work for reducing the manual workload is performed, another limitation to scalability is computation time. Many combinations in an unconstrained feature model can be identified as irrelevant or infeasible by designers, so they should be eliminated by defining constraints in the feature model before applying the method in this dissertation. Constraints should also be used to eliminate feature combinations for which the simulation model is unsolvable, but if designers fail to identify such a configuration, the method will simply output that the model was unsolvable. After these constraints are in place, computation resources may still be insufficient. As simulation runs for each configuration are independent, several machines can be used, and automatic distribution of the runs onto different machines is one possibility for further work. Regardless of whether one or more machines are used, computational resources may remain an issue. The advantage of FFIP is that it does not require a specific level of detail in the simulation model and that it is applicable to lightweight conceptual models such as those presented in this dissertation. There is a tradeoff between the level of detail in the simulation model and the computational resources. The proposed method does not constrain this tradeoff, so the user is expected to make the tradeoff in the context of the application.

8 Conclusions

The risk assessment method presented in this dissertation is applicable at the mechatronic system design stage, before the development process branches into sub-domains such as software and electrical systems. Some risks can only be identified when studying the interactions of software, electrical and mechanical elements of the design. Further, if the design needs to be changed in response to an identified risk, it is still possible to select the best design as the allocation of functionality to software, electrical and mechanical subsystems has not yet been fixed. If risks are only identified in the later design phases, the options involve compromises to the product quality or significant additional costs:

a) The mechatronic design may be changed in order to avoid the risk. This implies extensive re-work for all the design phases that followed the outdated design, which in turn results in financial cost and delays.
b) The system's behavior may be improved by working around the problem in the software, electrical or mechanical aspect of the detailed design. Although this may be less costly than option (a), the other properties of the mechatronic design, such as performance, may be compromised.
c) In some cases it is possible to accept the liability arising from the risk, if the cost of redesigning the system to be safer is high [108]. Although this is not an option in a tightly regulated industry, such as nuclear power production, it may be an option for other mechatronic applications.

Software safety is not addressed in this dissertation, as existing methods are applicable after the functional requirements of the mechatronic system have been allocated to SPL features. For control software applications, PLC targets need to be supported for configured instances of the SPL, and the open standard PLCopen XML is used for that purpose. At this point, the methodology and toolchain presented in this dissertation end: a feature model for the SPL has been obtained and the problem of configuring applications on IEC 61131-3 targets has been addressed. From this point on, the implementation, verification and validation in safety critical PLC environments such as PLCopen XML Safety Function Blocks can be handled by existing methods [47].

The flows within the simulated configuration flow graph model are qualitative, since the component implementation details are not yet determined when the system design is at an early phase. The FFIP framework presented here could be applied to a more advanced simulation model, since the Functional Failure Logic would continue to observe energy, material and signal flows in order to determine health degradation based on abnormal flow values.

References

[1] J.D. McGregor, L.M. Northrop, S. Jarrad, K. Pohl, Initiating software product lines, IEEE Software, 19 (2002) 24.
[2] S.E.I. Carnegie Mellon, Software Product Lines - Overview, in: http://www.sei.cmu.edu/productlines/.
[3] N. Papakonstantinou, S. Sierla, J. Alanen, K. Koskinen, Reducing redesign of safety critical control systems by early risk assessment, in: 8th IEEE International Conference on Industrial Informatics Osaka, Japan, 2010.
[4] K. Czarnecki, U. Eisenecker, Generative Programming: Methods, Tools, and Applications, Addison-Wesley, 2000.
[5] E. Althammer, E. Schoitsch, H. Eriksson, J. Vinter, A. Pataricza, G. Csertan, An open system for dependable system validation and verification support - The DECOS generic test bench, in: IEEE INDIN, Vienna, Austria, 2007.
[6] B. Werner, Object-oriented extensions for iec 61131-3, Industrial Electronics Magazine, IEEE 3(2009) 36.
[7] T. Kurtoglu, I.Y. Tumer, A Graph-Based Fault Identification and Propagation Framework for Functional Design of Complex Systems, Journal of Mechanical Design, 130 (2008).
[8] J.L. Rouvroye, E.G.v.d. Bliek, Comparing safety analysis techniques, Reliability Engineering & System Safety, 75 (2002) 289–294.
[9] J. Ren, I. Jenkinson, J. Wang, D.L. Xu, J.B. Yang, A methodology to model causal relationships on offshore safety assessment focusing on human and organizational factors, Journal of Safety Research, 39 (2008) 87-100.
[10] J. Wang, Offshore safety case approach and formal safety assessment of ships, Journal of Safety Research, 13 (2002) 81-115.
[11] B.A. Gran, R. Fredriksen, A.P.-J. Thunem, Addressing dependability by applying an approach for model-based risk assessment, Reliability Engineering & System Safety, 92 (2007) 1492-1502.
[12] A. Bobbio, E. Ciancamerla, G. Franceschinis, R. Gaeta, M. Minichino, L. Portinale, Sequential application of heterogeneous models for the safetyanalysis of a control system: a case study, Reliability Engineering & System Safety, 81 (2003) 269-280.
[13] B.G. Moffat, E. Abraham, M.P.Y. Desmulliez, D. Koltsov, A. Richardson, Failure mechanisms of legacy aircraft wiring and interconnects, IEEE Transactions on Dielectrics and Electrical Insulation, 15 (2008) 808 - 822
[14] Y. Papadopoulos, D. Parker, C. Grante, Automating the failure modes and effects analysis of safety critical systems, in: High Assurance Systems Engineering, Tampa, FL, USA, 2004.
[15] Y. Chen, L. Du, Y.-F. Li, H.-Z. Huang, X. Li FMECA for aircraft electric system, in: International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (ICQR2MSE), Xi'an, China, 2011.
[16] C.J. Price, N.S. Taylor, FMEA for multiple failures, in: Reliability and Maintainability Symposium, Anaheim, CA , USA, 1998.
[17] T. Hu, J. Yu, S. Wang, Research on complex system FMEA method based on functional modeling, in: Reliability, Maintainability and Safety, Chengdu, 2009.
[18] G. Mauri, J.A. McDermid, Y. Papadopoulos, Extension of hazard and safety analysis techniques to address problems of hierarchical scale, in: IEE Colloquium on Systems Engineering of Aerospace Projects (Digest No. 1998/249), 1998.
[19] T. Pasquale, E. Rosaria, M. Pietro, O. Antonio, A. Segnalamento Ferroviario, Hazard analysis of complex distributed railway systems, in: Reliable Distributed Systems, Florence, Italy, 2003.
[20] S. Schreiber, T. Schmidberger, A. Fay, J. May, J. Drewes, E. Schnieder, UML-based safety analysis of distributed automation systems, in: Emerging Technologies and Factory Automation, Patras, Greece, 2007.
[21] W.E. Vesely, F.F. Goldberg, N.H. Roberts, D.F. Haasi, The Fault Tree Handbook, US Nuclear Regulatory Commission, (1981).

[22] M. Stamatelatos, G. Apostolakis, Probabilistic Risk Assessment Procedures Guide for NASA Managers and Practitioners, NASA, Safety and Mission Assurance, 2002.
[23] D. Hofmann, M. Kopp, B. Bertsche, Development in Mechatronics — Enhancing reliability by means of a sustainable use of information, in: 2010 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), , Montreal, Canada, 2010
[24] Y. Kong, G. Pausch, K. Roemer, M. Neuer, C. Plettner, R. Lentering, J. Stein, Towards design and optimization of scintillation-detector systems: A Monte-Carlo simulation framework, in: IEEE Nuclear Science Symposium Conference Record (NSS/MIC), Knoxville, Tennessee, USA, 2010.
[25] T.W. Simpson, J.D. Poplinski, P.N. Koch, J.K. Allen, Metamodels for computer-based engineering design: Survey and recommendations, Engineering with Computers, 17 (2001) 129-150.
[26] G. Wang, S. Shan, Review of metamodeling techniques in support of engineering design optimization, Journal of Mechanical Design, 129 (2007) 370-381.
[27] J. Guo, X. Du, Reliability analysis for multidisciplinary systems with random and interval variables, AIAA Journal, 48 (2010) 82-91.
[28] C. Zang, M.I. Friswell, J.E. Mottershead, A review of robust optimal design and its application in dynamics, Computers and Structures, 83 (2005) 315-326.
[29] S. Nebel, A. Dieter, P. Muller, B. Bertsche, Application of ECSPN to RAMS modeling and analysis of hybrid drive systems in: Annual Reliability and Maintainability Symposium (RAMS), San Jose, California, 2010.
[30] S. Deb, K.R. Pattipati, V. Raghavan, M. Shakeri, R. Shrestha, Multisignal flow graphs: a novel approach for system testability analysis and fault diagnosis, in: IEEE Aerospace and Electronics Systems Magazine, 1995, pp. 14-25.
[31] T. Kurtoglu, I.Y. Tumer, D. Jensen, A functional failure reasoning methodology for evaluation of conceptual system architectures, Research in Engineering Design, 21 (2010) 209-234.
[32] IEC, Functional safety of electrical/electronic/programmable electronic safety-related systems, in: Part 4: Definitions and abbreviations, 1998.
[33] IEC, Safety of Machinery — Electrotechnical Aspects — Functional Safety of Electrical, Electronic and Programmable Control Systems, in, 2005.
[34] M. de Sousa, Restricting IEC 61131-3 programming languages for use on high integrity applications, in: IEEE ETFA, Hamburg, Germany, 2008.
[35] C. Wilwert, T. Clement, Evaluating quality of service and behavioral reliability of steer-by-wire systems, in: IEEE ETFA, Lisbon, Portugal, 2003.
[36] C. Fantuzzi, S. Marzani, C. Secchi, M. Ruggeri, A distributed embedded control system for agricultural machines, in: IEEE INDIN, Singapore, 2006.
[37] P. Wratil, Technology of safe drives, in: IEEE INDIN, Vienna, Austria, 2007.
[38] T. Tamandl, P. Preininger, Online self tests for microcontrollers in safety related systems, in: IEEE INDIN, Vienna, Austria, 2007.
[39] T. Novak, T. Tamandl, Architecture of a safe node for a fieldbus system, in: IEEE INDIN, Vienna, Austria, 2007.
[40] P. Fischer, M. Holz, M. Menzel, Network management for a safe communication in an unsafe environment, in: IEEE INDIN, Vienna, Austria, 2007.
[41] S. Mohan, K. Elango, S. Sivakumar, Evaluation of risk in canal irrigation systems due to non-maintenance using fuzzy fault tree approach, in: IEEE INDIN, Banf, Alberta, Canada, 2003.
[42] E. Althammer, E. Schoitsch, G. Sonneck, H. Eriksson, J. Vinter, Modular certification support - the DECOS concept of generic safety cases, in: IEEE INDIN, Daejeon, Korea, 2008.
[43] T. Novak, A. Treytl, Functional safety and system security in automation systems - a life cycle model, in: IEEE ETFA, Hamburg, Germany, 2008.
[44] T. Novak, A. Treytl, P. Palensky, Common approach to functional safety and system security in building automation and control systems, in: IEEE ETFA, Patras, Greece, 2007.
[45] N. Papakonstantinou, S. Sierla, Early Phase Fault Propagation Analysis of Safety Critical Factory Automation Systems, in: IEEE INDustrial INformatics (INDIN), Beijing, China, 2012.

[46] G. Cai, B.M. Chen, T.H. Lee, M. Dong, Design and implementation of a hardware-in-the-loop simulation system for small-scale UAV helicopters, Mechatronics, 19 (2009) 1057-1066.
[47] D. Soliman, G. Frey, Verification and validation of safety applications based on PLCopen safety function blocks, Control Engineering Practice, 19 (2011) 324-335.
[48] T. Bochot, P. Virelizier, H. Waeselynck, V. Wiels, Paths to Property Violation: A Structural Approach for Analyzing Counter-Examples, in: 12th IEEE International Symposium on High-Assurance Systems Engineering (HASE), 2010.
[49] E. Villani, P.E. Miyagi, R. Valette, Landing system verification based on petri nets and a hybrid approach, IEEE Transactions on Aerospace and Electronic Systems, 42 (2006) 1420-1436.
[50] C. Sonntag, S. Lohmann, A. Völker, S. Engell, Analyzing safety properties of hybrid processing systems: A case study on an industrial evaporator, Journal of Process Control, 18 (2008).
[51] S. Sierla, I. Tumer, N. Papakonstantinou, K. Koskinen, D. Jensen, Early integration of safety to the mechatronic system design process by the functional failure identification and propagation framework, Mechatronics, 22 (2012) 137-151.
[52] L.M. Northrop, Software product lines: reuse that makes business sense, in: Software Engineering Conference, Sydney, NSW, 2006.
[53] H.P. Breivold, S. Larsson, R. Land, Migrating Industrial Systems towards Software Product Lines: Experiences and Observations through Case Studies, in: Software Engineering and Advanced Applications, Parma, 2008.
[54] Y. Xue, Z. Xing, S. Jarzabek, Understanding Feature Evolution in a Family of Product Variants, in: 17th Working Conference on Reverse Engineering, Beverly, MA, 2010.
[55] N. Papakonstantinou, S. Sierla, K. Koskinen, Generating and validating product instances in IEC 61131-3 from feature models, in: IEEE ETFA, Toulouse, France, 2011.
[56] A. Fay, R. Zurawski, Progress in industrial automation programming and design: from a primitive to a simple solution [Guest Editorial], IEEE Industrial Electronics Magazine, 3 (2009).
[57] K. Czarnecki, A. Wasowski, Feature Diagrams and Logics: There and Back Again, in: Software Product Line Conference, 2007, pp. 23 – 34.
[58] S. Thiel, A. Hein, Modeling and using product line variability in automotive systems, IEEE Software, 19 (2002) 66 - 72.
[59] J. Sun, H. Zhang, Y.F. Li, H. Wang, Formal semantics and verification for feature modeling, in: 10th IEEE International Conference on Engineering of Complex Computer Systems, Shanghai, China, 2005.
[60] D. Benavides, S. Segura, P. Trinidad, A. Ruiz-Cortés, FAMA: Tooling a Framework for the Automated Analysis of Feature Models, in: Proceeding of the First International Workshop on Variability Modelling of Software-intensive Systems, Limerick, Ireland, 2007.
[61] K. Czarnecki, M. Antkiewicz, C.H.P. Kim, S.Q. Lau, K. Pietroszek, fmp and fmp2rsm: eclipse plug-ins for modeling features using model templates, in: OOPSLA '05 Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, San Diego, CA, USA, 2005.
[62] D. Streitferdt, M. Riebisch, I. Philippow., Details of formalized relations in feature models using OCL, in: 10th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, Huntsville, AL, USA, 2003
[63] H. Mei, W. Zhang, F. Gu, A feature oriented approach to modeling and reusing requirements of software product lines, in: 27th Annual International Computer Software and Applications Conference, Hong Kong, 2003.
[64] M. Kim, H. Yang, S. Park, A domain analysis method for software product lines based on scenarios, goals and features, in: 10th Asia-Pacific Software Engineering Conference, Chiangmai, Thailand, 2003.
[65] M.-O. Reiser, M. Weber, Managing Highly Complex Product Families with Multi-Level Feature Trees, in: 14th IEEE International Conference on Requirements Engineering, Minneapolis/St. Paul, MN 2006.
[66] D. Perovich, P.O. Rossel, M.C. Bastarrica, Feature model to product architectures: Applying MDE to Software Product Lines, in: Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture, Cambridge 2009

[67] V.T. Sarinho, A.L. Apolinario, Combining feature modeling and Object Oriented concepts to manage the software variability, in: IEEE International Conference on Information Reuse and Integration, Las Vegas, NV, 2010.
[68] G. Botterweck, S. Thiel, D. Nestor, S. bin Abid, C. Cawley, Visual Tool Support for Configuring and Understanding Software Product Lines, in: 12th International Software Product Line Conference, Limerick, 2008.
[69] C. Kästner, T. Thüm, G. Saake, J. Feigenspan, T. Leich, F. Wielgorz, S. Apel, FeatureIDE: Tool Framework for Feature-Oriented Software Development, in: 31th IEEE International Conference on Software Engineering, Vancouver, Canada, 2009.
[70] J. Bartholdt, M. Medak, R. Oberhauser, Integrating Quality Modeling with Feature Modeling in Software Product Lines, in: Fourth International Conference on Software Engineering Advances, Porto 2009.
[71] J. White, D.C. Schmidt, D. Benavides, P. Trinidad, Automated Diagnosis of Product-Line Configuration Errors in Feature Models, in: 12th International Software Product Line Conference, Limerick, 2008.
[72] G.G. Ge, E.J. Whitehead, Rhizome: A Feature Modeling and Generation Platform, in: 23rd IEEE/ACM International Conference on Automated Software Engineering, L'Aquila, 2008.
[73] BigLever, BigLever Software Gears website, in: http://www.biglever.com/solution/product.html.
[74] Pure-systems, pure::variants Technical White Paper, in: http://www.pure-systems.com/fileadmin/downloads/pv-whitepaper-en-04.pdf.
[75] L. Kwanwoo, G. Botterweck, S. Thiel, Feature-Modeling and Aspect-Oriented Programming: Integration and Automation, in: 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing, Catholic University of Daegu, Daegu, KR, 2009.
[76] J. Liu, J. Dehlinger, H. Sun, R. Lutz, State-Based Modeling to Support the Evolution and Maintenance of Safety-Critical Software Product Lines, in: IEEE ECBS, 2007, pp. 596 – 608.
[77] J. Liu, Safety analysis of software product lines using state-based modeling, in: IEEE ISSRE, Chicago, Illinois, USA, 2005.
[78] C. Kastner, S. Apel, Type-Checking Software Product Lines - A Formal Approach, in: IEEE/ACM ASE, 2008, pp. 258 - 267.
[79] J.v. Amerongen, Mechatronic design, Mechatronics, 13 (2003) 1045–1066.
[80] K. Thramboulidis, Model-Integrated Mechatronics – Toward a New Paradigm in the Development of Manufacturing Systems, IEEE Transactions on Industrial Informatics, 1 (2005) 54-61.
[81] H. Sun, M. Hauptman, R. Lutz, Integrating Product-Line Fault Tree Analysis into AADL Models, in: IEEE HASE, Dallas, Texas, USA, 2007, pp. 15 – 22.
[82] R. Lutz, Enabling Verifiable Conformance for Product Lines, in: Software Product Line Conference SPLC, Limerick, Ireland, 2008.
[83] Q. Feng, R.L. Robyn, Bi-directional safety analysis of product lines, Journal of Systems and Software, 78 (2005) 111-127.
[84] N. Papakonstantinou, S. Sierla, K. Koskinen, Object oriented extensions of IEC 61131-3 as an enabling technology of software product lines, in: IEEE Emerging Technologies in Factory Automation (ETFA), Toulouse, France, 2011.
[85] R. Froschauer, D. Dhungana, P. Grunbacher, Managing the Life-cycle of Industrial Automation Systems with Product Line Variability Models, in: 34th Euromicro Conference Software Engineering and Advanced Applications, Parma, 2008.
[86] K. Thramboulidis, Different perspectives [Face to Face; "IEC 61499 function block model: Facts and fallacies"], IEEE Industrial Electronics Magazine, 3 (2009) 7.
[87] M. Bonfe, C. Fantuzzi, Design and verification of mechatronic object-oriented models for industrial control systems, in: IEEE Conference Emerging Technologies and Factory Automation, Lisbon, 2003.
[88] S. Ono Kajihara, H. M. Houzouji, H. Taruishi, Y. Takayanagi, Development and products of the object-oriented engineering tool for the integrated controller based on IEC 61131-3, in: SICE Annual Conference Sapporo, 2004.

[89] M. Bonfe, C. Fantuzzi, Object-oriented approach to PLC software design for a manufacture machinery using IEC 61131-3 norm languages, in: IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Como, 2001.
[90] G. Aiello, M. Alessi, M. Bruccoleri, C. D'Onofrio, G. Vella, An Agile methodology for Manufacturing Control Systems development, in: 5th IEEE International Conference on Industrial Informatics, Vienna, 2007.
[91] V.M. González, A.L. Sierra Diaz, P. García Fernández, A. Fernández Junquera, R.M. Bayón, MIOOP. An object oriented programming paradigm approach on the IEC 61131 standard, in: IEEE Conference on Emerging Technologies and Factory Automation, Bilbao, 2010.
[92] D. Witsch, B. Vogel-Heuser, Close integration between UML and IEC 61131-3: New possibilities through object-oriented extensions, in: IEEE Conference on Emerging Technologies & Factory Automation, Mallorca, 2009.
[93] R. Drath, A. Luder, J. Peschke, L. Hundt, AutomationML - the glue for seamless automation engineering, in: IEEE International Conference on Emerging Technologies and Factory Automation, Hamburg, 2008.
[94] R.J.J. Stamm'Ler, M.J. Abbate, Methods of Steady-State Reactor Physics in Nuclear Design, Academic Pr, 1983.
[95] P. Abagyan, V.I. Golubev, N.D. Golyaev, A.V. Zvonarev, Y.F. Koleganov, M.N. Nikolaev, M.Y. Orlov, Propagation of neutrons in uranium dioxide II. Doppler effect in U238, Atomic Energy, 25 (1968) 1090-1094.
[96] D. Jensen, I.Y. Tumer, T. Kurtoglu, Modeling the propagation of failures in software-driven hardware systems to enable risk-informed design, in: ASME (Ed.) International Mechanical Engineering Congress and Exposition, Boston, MA, 2008.
[97] D. Jensen, I.Y. Tumer, T. Kurtoglu, Design of an Electrical Power System using a Functional Failure and Flow State Logic Reasoning Methodology, in: P. Society (Ed.) Prognostics and Health Management, San Diego, 2009.
[98] D. Weld, J. de Kleer, Readings in qualitative physics, Morgan Kauffman, 1987.
[99] K. Forbus, Qualitative Process Theory, Artificial Intelligence, 24 (1984) 85-168.
[100] P. Struss, Mathematical Aspects of Qualitative Reasoning, International Journal of Artificial Intelligence in Engineering, 3 (1988) 156-169.
[101] B.J. Kuipers, Qualitative Simulation, Artificial Intelligence, 29 (1986) 289-338.
[102] R. Stone, K. Wood, Development of a Functional Basis for Design, Journal of Mechanical Design, 122 (2000) 359-370.
[103] N. Papakonstantinou, D. Jensen, S. Sierla, I. Tumer, Capturing interactions and emergent failure behavior in complex engineered systems and multiple scales, in: ASME IDETC/CIE, Washington, DC, USA, 2011.
[104] M. Davis, R. Sigal, E.J. Weyuker, Computability, Complexity, and Languages, 1994.
[105] J.E. Gaffney, C.F. Davis An Approach to Estimating Software Errors and Availability, in: Eleventh Minnowbrook Workshop on Software Reliability, 1988.
[106] N. Papakonstantinou, S. Sierla, I.Y. Tumer, J. D., Using Fault Propagation Analyses for Early Elimination of Unreliable Design Alternatives of Complex Cyber-Physical Systems, in: ASME IDETC/CIE 2012, Chicago, Illinois, USA, 2012.
[107] D. Pilone, UML2.0 in a Nutshell, O'Reilly Media Inc, 2005.
[108] D. Rechenthin, Project safety as a sustainable competitive advantage, Journal of Safety Research, 35 (2004) 297-308.

Aalto-DD 148/2012

9 789526 048611

ISBN 978-952-60-4861-1
ISBN 978-952-60-4862-8 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934
ISSN 1799-4942 (pdf)

Aalto University
School of Electrical Engineering
Department of Automation and Systems Technology
www.aalto.fi