Simulation of Interactions and Emergent Failure Behavior During Complex System Design

Nikolaos Papakonstantinou

Seppo Sierla
e-mail: seppo.sierla@aalto.fi

Department of Automation and Systems Technology,
School of Electrical Engineering,
Aalto University,
P.O. Box 15500,
Aalto, 00076 Finland

David C. Jensen

Irem Y. Tumer[1]
e-mail: irem.tumer@oregonstate.edu

Complex Engineered System Design Laboratory,
School of Mechanical, Industrial, and
Manufacturing Engineering,
Oregon State University,
204 Rogers Hall,
Corvallis, OR 97331

Emergent behavior is a unique aspect of complex systems, where they exhibit behavior that is more complex than the sum of the behavior of their constituent parts. This behavior includes the propagation of faults between parts, and requires information on how the parts are connected. These parts can include software, electronic and mechanical components, hence requiring a capability to track emergent fault propagation paths as they cross the boundaries of technical disciplines. Prior work has introduced the functional failure identification and propagation (FFIP) simulation framework, which reveals the propagation of abnormal flow states and can thus be used to infer emergent system-wide behavior that may compromise the reliability of the system. An advantage of FFIP is that it is used to model early phase designs, before high cost commitments are made and before high fidelity models are available. This has also been a weakness in previous research on FFIP, since results depend on arbitrary choices for the values of model parameters and timing of critical events. Previously, FFIP has used a discrete set of flow state values and a simple behavioral logic; this has had the advantage of limiting the range of possible parameter values, but it has not been possible to model continuous process dynamics. In this paper, the FFIP framework has been extended to support continuous flow levels and linear modeling of component behavior based on first principles. Since this extension further expands the range of model parameter values, methods and tools for studying the impact of parameter value changes are introduced. The result is an evaluation of how the FFIP results are impacted by changes in the model parameters and the timing of critical events. The method is demonstrated on a boiling water reactor model (limited to the coolant recirculation and steam outlets) in order to focus the analysis of emergent fault behavior that could not have been identified with previously published versions of the FFIP framework. [DOI: 10.1115/1.4007309]

1 Introduction

1.1 Challenges for Complex System Design. The design of large complex systems faces numerous challenges due to the complexity of the design information content, unknown interactions between subsystems, and the need to integrate expertise of designers or teams of designers from different domains. Specifically, we recognize three challenging tasks that arise in the design stage for these systems: (1) Codesign of the multiple domains of technology; (2) Determining the effects of emergent behavior; and, (3) Determining risks across the system from fault propagation. The method presented in this paper is aimed at addressing the challenges in these tasks.

The term codesign is most often used to refer to technologies that require close integration of electrical hardware and software systems as found in mechatronics and consumer electronics [1,2]. However, the technical challenges and the general solution approach are also applicable to the integration of electromechanical and human control systems. Specifically, it is challenging to represent necessary system design information across technical domains at a similar and relatable abstraction level. This representation is necessary for the development of interfaces and interaction behavior. While it is advantageous to develop different subsystems concurrently, often in the design stage different subsystems are at various levels of design refinement. Formal model representation languages, such as Systems Modeling Language (SysML) [3], serve to implement a model-based design approach to address this challenge. That is, high-level system models may be composed of multiple representations to capture design information across domains even when subsystems are at different levels of design refinement.

Emergent behavior is a unique aspect of systems where they exhibit behavior that is more complex than the sum of the behavior of their constituent parts. Along with the emergent behavior, which is intended and is used to implement the desired functionality of the system, unforeseen behavior may also occur. In this work, focus in on emergent behavior that may compromise the reliability or safety of the system. Emergent behavior is defined as degradation or loss of functionality of a subsystem that is caused by poorly chosen design parameters in other subsystems. Our goal is to provide methods and tools for studying how changing several such parameters impacts the occurrence of emergent behavior.

Finally, the third challenging task is concerned with evaluating systems designs for risk and safety. Traditional approaches focus on identifying faults and their probabilities of occurrence and consequence. For large systems this can create certain statistical limitations for risk analysis. For example, in complex systems faults are rarely independent. Further, the consequence of a component fault depends on the connection that the component has with the rest of the system. This is evidenced by the increase in failures that affect systems that deploy both computational and physical elements. Therefore a risk assessment of complex systems should be based on a failure analysis that accounts for fault propagation across domains and subsystems.

1.2 Prior Work. A function failure analysis approach was developed and presented in prior work to identify the functional

[1]Corresponding author.

Contributed by the Design Engineering Division of ASME for publication in the JOURNAL OF COMPUTING AND INFORMATION SCIENCE IN ENGINEERING. Manuscript received October 25, 2011; final manuscript received July 19, 2012; published online August 21, 2012. Assoc. Editor: Shuming Gao.

losses and determine their effects through downstream propagation. This method was developed to help address the three challenging designs tasks presented in Sec. 1.1. Specifically, the FFIP framework was developed to capture the effects of complex system interactions early in the design stage and presenting the effects and propagation of faults in terms of functional losses [4–7]. The FFIP design stage analysis framework was developed to identify the system-wide functional effect of component failures, even when the fault propagation paths cross the boundaries of electronic, mechanical, and software subsystems [8,9].

To facilitate the codesign challenge, the FFIP method uses a functional model (FM) to represent multiple domain subsystems at the same abstraction level. Further, these functions are linked to generalized components configuration representation. The FFIP method is a model-based simulation approach and behavior associated to these components can be at differing levels of design refinement. The state-based simulation of nominal and faulty behavior also assists in identifying emergent behavior early in the design stage. Finally, the FFIP analysis method consists of triggering a fault and recording the fault propagation and system impact. These results can then be used to evaluate the risk of potential system designs without independence assumptions and across the different technical domains of the system.

Thus, FFIP is applied at the early concept design phase using an abstract system representation to eliminate unreliable designs before costly design commitments are made. High fidelity system models are not available at this phase, therefore the results of an FFIP analysis are strongly affected by specific modeling choices in the component model and the sequence and timing of critical event scenarios.

1.3 Objectives and Contributions. In this paper, the FFIP framework and supporting tools are extended significantly to better address the challenges of emerging behavior. While high fidelity simulation at an early phase is not possible, the results should not be totally dependent on specific model parameter values. The simulation approach presented in this paper permits varying the values of key design parameters and the timing of critical events; simulation results reveal the impact of the variations on the emergent behavior. Previous research has expressed values for energy, material, and signal (EMS) flows with an enumeration such as [zero, low, nominal, high], so that component behavioral models (BM) determine output flow values according to input flow values and the current nominal or faulty state of the component. This approach is not sufficiently fine-grained for capturing how several parameter changes might either reinforce or weaken each other in causing emergent behavior, so the framework needs to be modified to support continuous flow values in order to describe feedback loops. This extension of the FFIP approach to system representation addresses the first challenge identified earlier by allowing for more detailed subsystem behavior, while maintaining the equal abstraction system-level representation necessary for codesign.

In order to systematically investigate the effects of different parameter values and different timing of critical events, it will be necessary to run the FFIP simulation for each combination of parameter values, so the number of simulation runs grows exponentially as more parameters are brought into the scope of the study. In order for this to be feasible, a user interface is needed for specifying the parameters and the ranges in which they are varied. The second goal of this paper is then to propose a generic and scalable algorithm for automatically running the complete set of simulation runs implied by the user's choices. The algorithm and user interface are implemented and interfaced to the Matlab/Simulink based FFIP framework, producing Excel output for each run, which can be further sorted or filtered according to the health status of a function of interest specified by the user. The paper demonstrates the use of these tools for identifying the most relevant parameters for hazards and for focusing their values to the interesting range in subsequent series of automatic simulation runs. With these changes, the failure analysis can provide better results for risk assessment early in the design stage.

2 Background

The method presented in this paper uses an early design stage system representation and behavioral simulation to evaluate risk in functional terms. A variety of methods and techniques have been developed for risk assessment, system information sharing, and simulating expected system performance, discussed next.

2.1 Risk Assessment in Complex System Design. This research fundamentally aims to reduce or eliminate the risks of malfunctions and failures. Many methods exist in practice and in research literature to enable risk mitigation. Failure modes and effects analysis (FMEA) is used in safety critical industries for safety analysis. It is based on past experience and defines the effects of a single failure at a time for a number of component failure modes. These effects can be identified by using the fault tree analysis (FTA) method [10]. Some research has used FMEA for multiple failures but if detailed design documents are used the combinations of failures grow beyond a manageable size in complex systems [11]. Functional models can be used with FMEA to describe more clearly the effects upon the functions of complex systems [12]. The functional failure analysis (FFA) is a method similar to FMEA but at a functional level of the system design [13]. Although attempts of partially automating this process have been made, FMEA still remains a manual and laborious process [14]. HAZard and OPerability (HAZOP) is a qualitative method that can be performed on the functional model of a complex system. System functions are combined with a set of keywords with the aim to identify all possible failure modes [15]. This method also requires significant manual effort, although there are attempts to automate parts of it by using knowledge databases [16]. Probabilistic risk assessment (PRA) [17] is a quantitative method that can identify weaknesses and vulnerabilities in complex systems. It requires a detailed design and the failure rate information of the components used. Therefore, it can only be applied in the latter phase of the system design process [18].

In general, a high level of system behavioral and interaction knowledge is required to perform these assessment methods. Therefore these methods are most advantageous for verifying the safety of a well-refined design. The goal of this work is to utilize risk assessment early in the design stage to help designers make design decisions, before a refined design has to be completed.

2.2 Complex System Design Representation. The primary method for conveying the relationship between elements in the design stage of complex systems is through abstract engineering design models of functionality and interface requirements. These models range from detailed function and behavior information to loose hierarchical artifact relationships. These approaches aim at capturing the complexity of the system through a single (or set) of static representations. Examples include: block diagrams based on "function logic" [19], a historical design repository approach proposed as a framework to store component information and relate the elements of information to each other [20], and other function-based representations that support search and modeling processes of conceptual design [21]. These approaches utilize the concept of function to connect multiple domain system elements at the same abstraction level. Other research has explored languages for describing product functionality and relating it to product behavior [22] and using these languages to reason about designs [23]. SysML is a formalism for representing many different aspects of system design including function, behavior and structure. Several recent works have focused on extending the static system representation of SysML for dynamic system simulation in other tools [24]. Finally, several researchers have introduced representation

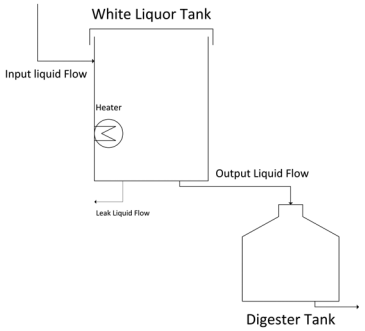

Fig. 1 Piping and instrumentation diagram of the example process

Fig. 2 Functional model of the example process

as part of methods of tackling design time failure analysis. Examples include: a function failure design method [25] which uses functional models and historical failure data to map the functionality of a system, a Bayesian network analysis tool for evaluating properties of function structures using dependencies between flows and functions [26], and a risk based decision-making and cost-benefit analysis method [27] to assess the risk of integrating health management capabilities in aerospace systems at the system-level design stage.

This research builds on the functional modeling approach to design representation. These functional representations are also mapped to generic component and behavioral representations. The presented approach leverages the above modular approach to design, where system functions, structure and behavior are created from combining lower-level models.

2.3 Simulation-Based Design. This research utilizes behavioral simulation of models at multiple abstraction levels with increasingly higher fidelity, including models that do not require component geometry details to be implemented at the architecture level. Simulation-based design methods require the capability of specifying detailed input design parameters and using them to obtain a model response. This simulation process is called a forward model, which system designers use to account for the effects of variability in the input and design parameters on the model response, thereby incorporating uncertainty into the design process. Common techniques for forward model analysis include sampling techniques (e.g., Monte Carlo simulation), response surface models and metamodeling [28,29]. These methods support reliability-based and robust design optimization techniques. Reliability-based methods [30] estimate the probability of system response based upon specified probability distributions but no effort is made to minimize variation, whereas robust design [31] strives to minimize the effects of variation.

Simulation-based techniques have also been used to assess risk factors in a design and analyzing the effects of faults in a system. For example, directed graphs and Multi-Signal Flow Graphs [32] are used in many domains to analyze dependencies and fault propagation and model cause-effect dependencies.

At the early design stage, the detailed specifications of component geometry and system topology required by some methods are usually not available. To address this problem, we expand our prior work to address simulation and reasoning in early design [8,33–35].

3 Methodology

The FFIP framework was developed to capture the effect of complex system interactions early in the design stage and presenting the effect and propagation of faults in terms of functional losses [4–7]. The simulation and reasoning approach in FFIP has its roots in qualitative physics [36] and qualitative reasoning [35,37,38]. FFIP utilizes a finite state representation of system behavior, and performs reasoning based on qualitative relationships between functional and behavioral models of system components. Prior to this research, FFIP has been demonstrated on relatively simple example systems to demonstrate different aspects of the methodology. This paper presents how the FFIP simulation and analysis can be improved to better meet some of the challenges found in the design of large complex systems [8,9].

This section introduces the FFIP methodology by a simple example, so that the reader may understand how the results in Sec. 6 were obtained. The example is a simplified subprocess from the pulp and paper industry (Fig. 1). White liquor is heated and supplied to a digester in order to enable a chemical reaction necessary to pulp production. The white liquor tank has a continuous flow of incoming and outgoing liquor, and the tank serves the purpose of intermediate storage and heating of the liquor. The desired functionality is expressed as a functional model, according to the functional basis defined in Ref. [39] (Fig. 2). Failure propagation analyses with FFIP use the functional model to identify loss of functions that are not necessarily associated with failed components.

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. 3). The CFG and functional model have the same flows between functions and components, following the taxonomy defined in Ref. [39]. 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

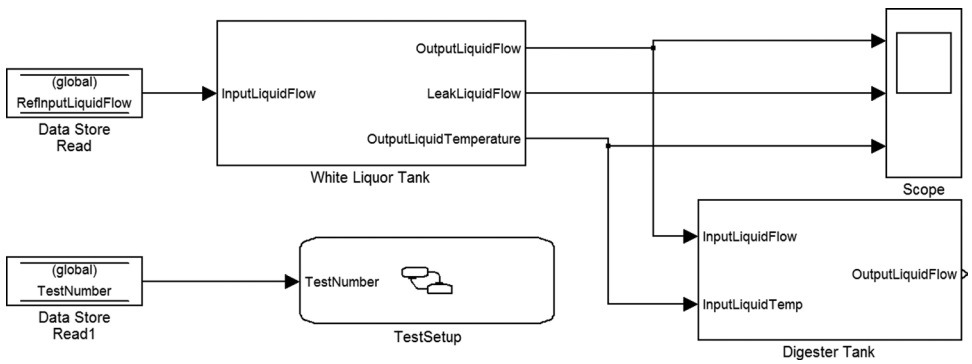

Fig. 3 Configuration flow graph of the example process

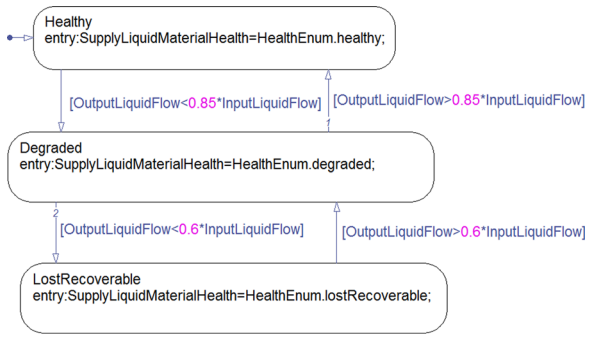

Fig. 4 Function failure logic for the supply liquid material (white liquor) function

determine if a function defined in the functional model (Fig. 2) has been degraded or lost. The FFL for the supply liquid material (white liquor) function is shown in Fig. 4; it compares the input and output flows of the white liquor tank shown in Fig. 3.

The relationship between input and output flows of a component in a CFG is defined by a BM. The BM for the white liquor tank is shown in Fig. 5. 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 tank 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 TankLeaking state.)

In earlier versions of the FFIP framework, flow levels were described with an enumeration (zero, low, nominal, high), but this approach is insufficient for our boiling water reactor (BWR) 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 Nominal in Fig. 5. Since the behavioral model is executed by a fixed-step solver, the tank 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 components internal variables (such as the tank 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 Leaking 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 objective in this article is to observe how parameter changes in the behavioral models affect the emergent behavior. 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. RefInputLiquidFlow in Fig. 3 is the reference liquid flow that should go through the white liquor tank and the digester in normal operation of the process; it is a design parameter. In Fig. 5, Leaking state, leakSize is a parameter of a fault.

The methodology used in this paper 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. 6 illustrates the methodology

Fig. 5 Behavioral model of the white liquor tank component

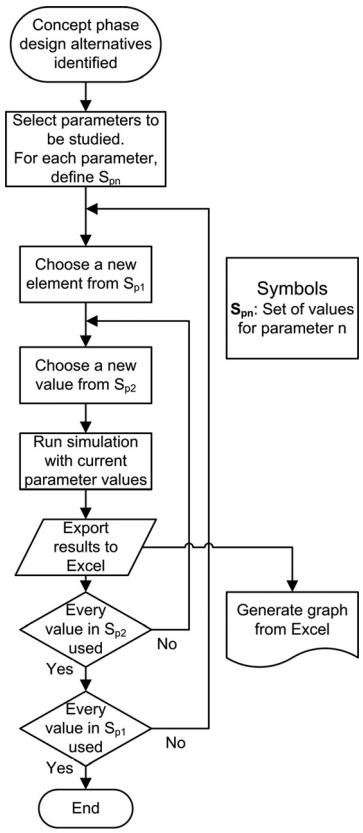

Fig. 6 A flowchart describing how every combination of parameter values is simulated systematically to determine those combinations of parameter values that result in degradation or loss of functions

in the case of two parameters, and additional parameters can be handled by adding a nested loop for each new parameter. Figure 7 displays the graph that is obtained in the flowchart when parameter 1 is RefInputLiquidFlow, S_{p1} is [5.0, 10.0, 20.0], parameter 2 is leakSize and S_{p2} is [0.5, 2.0, 3.5]; each curve in the graph is the trend for the tank level in one simulation run. The output of function failure logic for the simulation in Fig. 7 is shown in Table 1. The Transmit Thermal Energy function is lost when the tank level drops below the level of the heating element. The status is lost recoverable, since the function can return to healthy when the level rises above the heating element. The Convert Thermal Energy to Chemical function, which is associated with the digester component, may fail due to fault propagation from another component, in this case a leak in the white liquor tank.

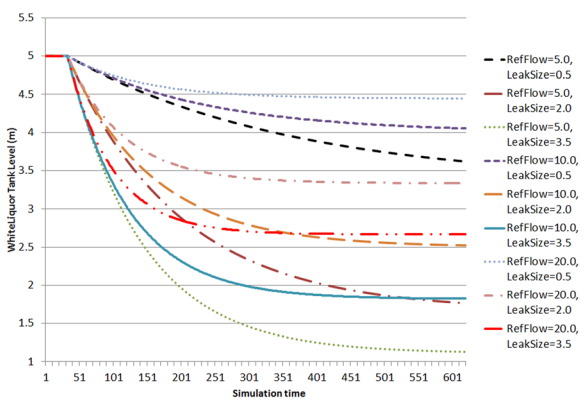

Fig. 7 The result of performing the procedure in Fig. 6 when parameter 1 is RefInputLiquidFlow and parameter 2 is leakSize.

The purpose of this simple example is not to demonstrate the paper's contribution; the more complex boiling water reactor case study is used for this purpose. The aim of this section has been to describe how the results in Sec. 6 were obtained.

4 Case Study: Analysis of a Boiling Water Reactor Design

The proposed approach is applied to a BWR, focusing on its steam outlets. The BWR uses the thermal energy from nuclear fission inside the reactor core to produce high-pressure steam. The steam is guided via pipes and valves to steam turbines to provide motion to electrical generators. The steam flow to the turbine is controlled by a pressure control valve, such that the pressure inside the reactor vessel is kept steady. Fuel rods are arranged into assemblies, which are submerged under water. The space between the fuel rods forms flow channels through which the coolant pumps circulate water, which acts as both coolant and moderator. As a coolant, it removes thermal energy from the core, and a greater flow rate reduces the void fraction, which is the proportion of steam in the coolant. As a moderator, water slows neutrons, and the number of thermal neutrons is increased when the void fraction is decreased.

Several feedback loops affect the energy production at the core. An increase of pressure in the reactor vessel decreases the void fraction and increases power output. A decrease of coolant flow rate has the opposite effect, but insufficient flow can cause a heat transfer crisis from fuel rods to coolant due to steam buildup; this can lead to fuel damage due to overheating of the rods. The Doppler Effect can also reduce the neutron flux. This is a physical phenomenon in which neutron absorption (without fission) by the 238U and 240Pu isotopes increases when the fuel temperature increases [40].

A reactor protection system is present which can initiate an emergency shutdown of the reactor, referred to as a "scram". Some conditions that can trigger a scram are related to the neutron flux exceeding a threshold value, the coolant flow rate being below a threshold value or the external electrical power coming from the power grid being lost. Maintaining coolant flow at all times is necessary to prevent steam buildup at the core, since this may cause a heat transfer crisis and damage to the fuel rods due to overheating. However, a BWR has positive feedback between reactor power and coolant flow, so a controlled decrease of flow in emergency situations is necessary. During scram, a common design in a BWR is to drive coolant pumps down to minimum rotations per minute gradually by a ramp. The slope of this ramp and the need for emergency power for the pumps are design parameters of interest in this case study.

There are some known hazards to maintaining acceptable pressure and flow in the reactor vessel that a BWR design must mitigate. The pressure control valve in the steam outlet pipe is driven by a sophisticated controller that must react rapidly to changes. If the pressure control subsystem closes the valve, it notifies the turbine protection subsystem, which relieves the pressure by opening the dumper valve. However, due to the sophisticated nature of the pressure control subsystem, unintentional closure of the valve due to software malfunction is possible, resulting in no notification and a pressure shockwave back to the reactor vessel.

4.1 Abstractions: Functional Model and Configuration Flow Graph. The first step in analyzing the early design of the BWR is identification of desired functionality. The processes inside a BWR are expressed through functions. These functions are aggregated and connected to create a structure based on the flows of EMS affected by each function. This function structure is known as a FM. The Functional Basis [39] is used as a taxonomy for naming the functions and the flows. The scope of the case

Table 1 The output of function failure logic for the simulation in Fig. 7

	Supply liquid material health	Transmit thermal energy health	Convert thermal energy to chemical health
RefFlow = 5.0, LeakSize = 0.5	Degraded (t232)	Healthy	Degraded (t232)
RefFlow = 5.0, LeakSize = 2.0	Degraded (t76)	Lost recoverable (t416)	Degraded (t76), lost recoverable (t416)
RefFlow = 5.0, LeakSize = 3.5	Degraded (t57), lost recoverable (t113)	Lost recoverable (t197)	Degraded (t57), lost recoverable (t197)
RefFlow = 10.0, LeakSize = 0.5	Degraded (t310)	Healthy	Degraded (t310)
RefFlow = 10.0, LeakSize = 2.0	Degraded (t78), lost recoverable (t234)	Healthy	Degraded (t78)
RefFlow = 10.0, LeakSize = 3.5	Degraded (t58), lost recoverable (t123)	Lost recoverable (t292)	Degraded (t58), lost recoverable (t292)
RefFlow = 20.0, LeakSize = 0.5	Healthy	Healthy	Healthy
RefFlow = 20.0, LeakSize = 2.0	Degraded (t83)	Healthy	Degraded (t83)
RefFlow = 20.0, LeakSize = 3.5	Degraded (t59), lost recoverable (t162)	Healthy	Degraded (t59)

Table 2 Mapping of system functions to components

Function	Component
Transmit thermal energy	Flow channels and fuel rods
Condition radioactive energy	Flow channels
Supply electrical energy	Power supply rail
Transport liquid material	Coolant pumps
Process status signals	Reactor protection
Inhibit radioactive energy	Control rods
Regulate and guide gas material	Pressure control valve
Process status signals	Pressure control

study is restricted to acute situations that might require emergency shutdown of the reactor. The model is not intended for studying residual heat removal, which is a process that can take months, involving additional physical phenomena and technical systems that are not included in this model.

A CFG is produced by selecting components to implement the functions, and connecting them with the same EMS flows as in the functional model. Table 2 contains the direct mapping between some of the functions and components for a BWR design used in the FFIP analysis. The components include mechanical, electrical and software parts and can contain components hierarchically to obtain the level of granularity needed to support mapping between the CFG and FM. In Fig. 8, the top level CFG is provided, and the internals of the reactor component are shown in Fig. 9.

The reason for having separate a FM and CFG is that, due to fault propagation, a component failure may result in degradation or loss of functionality mapped to another component. For example, a leak in a pipe (a faulty component state) can cause a degraded functional state for a "store liquid" function, even though the tank component mapped to this function is in a nominal state. In the analysis, the CFG is simulated, critical events are inserted as faulty component states, and a logic relating function

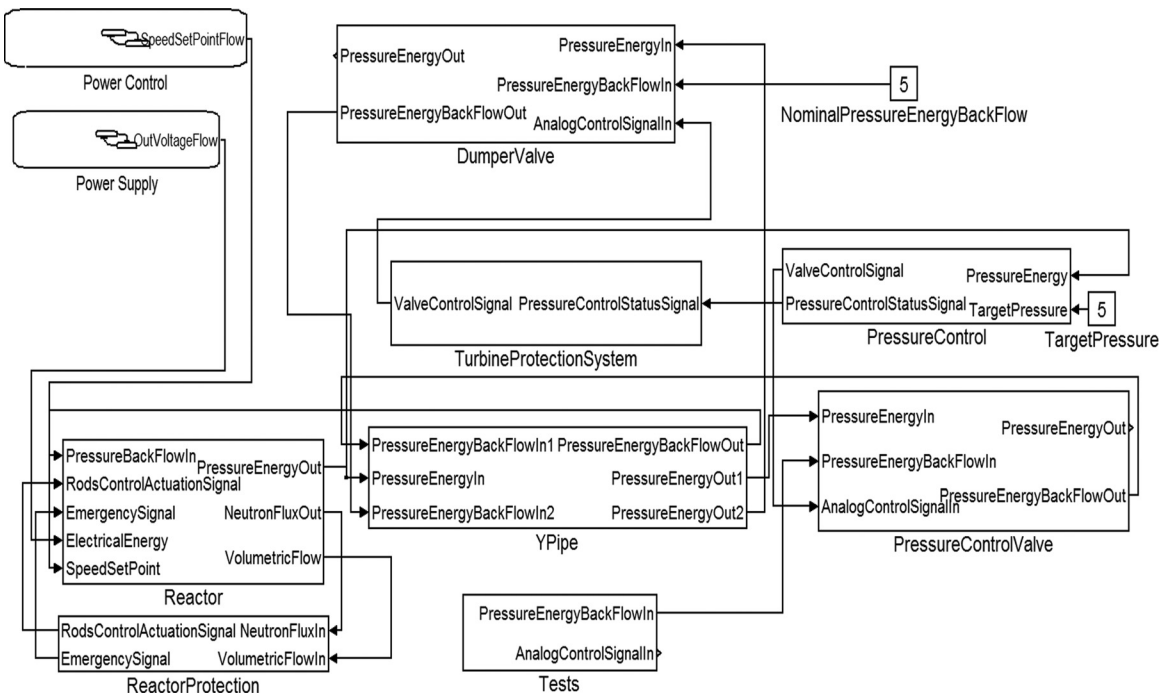

Fig. 8 Top level CFG model for boiling water reactor core and its steam outlets

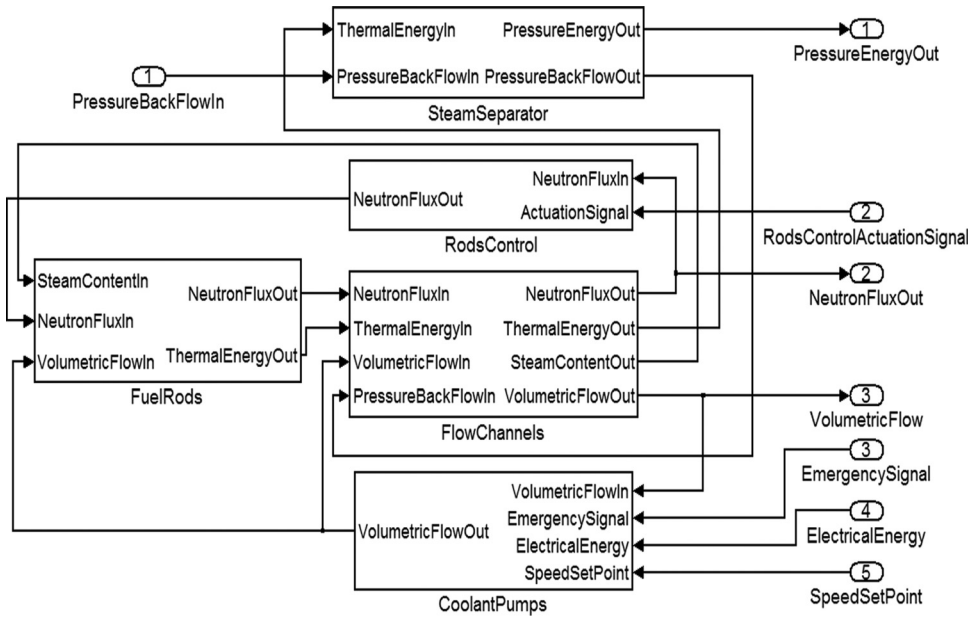

Fig. 9 Internals of the reactor component in Fig. 8

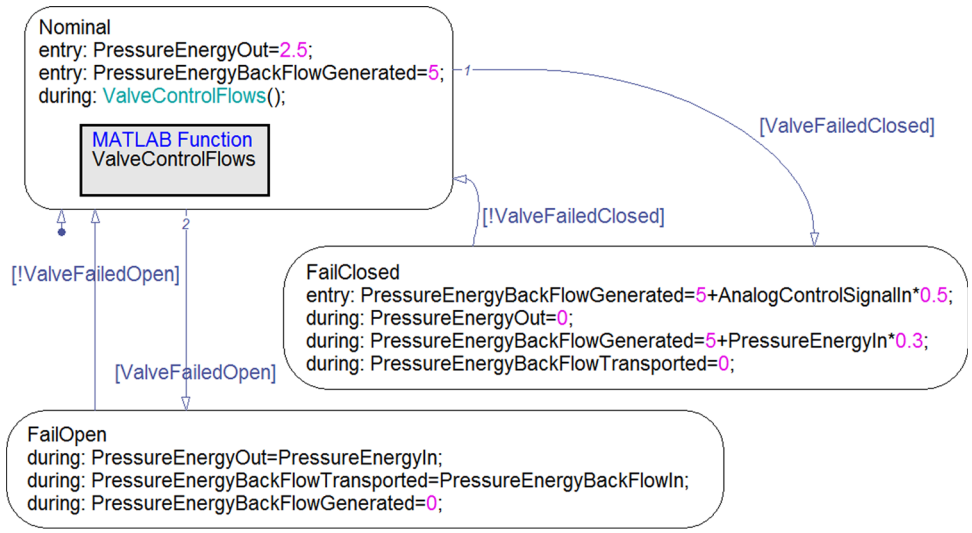

Fig. 10 Behavioral logic for the PressureControlValve component in Fig. 8

to failures, namely, the FFL, is used to observe component input and output flows and determine the health of the related function. This is possible because the FM and CFG use the same EMS flows.

4.2 Behavior and Function Failure Logic Reasoning. The behavior of the components is implemented using the Simulink environment and the Stateflow package, which are part of the Matlab tool. Stateflow models are used to represent hybrid behavior. There is a state for each nominal and faulty component mode, and within these states there can be continuous behavior. An example of behavior logic for the pressure control valve component is demonstrated in Fig. 10. State transitions are triggered by critical events that are inserted into the simulation. The behavioral models should be constructed by experts familiar with the application domain, the physical phenomena and the impact of various parameter value changes on these phenomena and design alternatives.

The FFL reasoner code for the "transmit thermal energy" function, which is partly implemented by the fuel rods, is presented in Fig. 11. The FFL reasoner determines the health of this function by the value of the temperature parameter during the simulation. The values of 5.5, 6, and 7 for entering the health status "degraded", "lost recoverable" and "lost," respectively, are based on qualitative descriptions for the Temperature levels. The value of 5 is defined as nominal.

5 Automation Framework and User Interface

Figure 12 shows a screenshot the user interface for specifying a set of FFIP simulation scenarios to be generated and run automatically. After specifying the Simulink.m file containing the parameter, the user can select any of these parameters and then specify the limits of its range and the interval between samples. From the selections in the "choices made" box, the heat transfer coefficient (HTC) will be given values [1,3,5] in separate simulations. Similarly, the time delay between the injected pressure control valve closure malfunction and the injected power rail failure (Delay) will be one of the following [0, 40, 80]. The slope of the ramp for

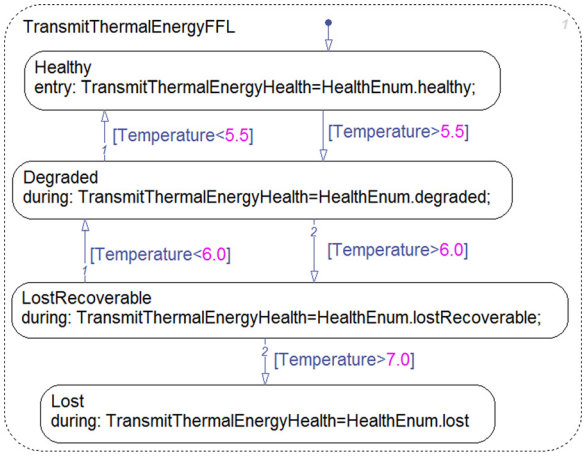

Fig. 11 Stateflow chart of the FFL reasoner for the "transmit thermal energy" function in Table 2

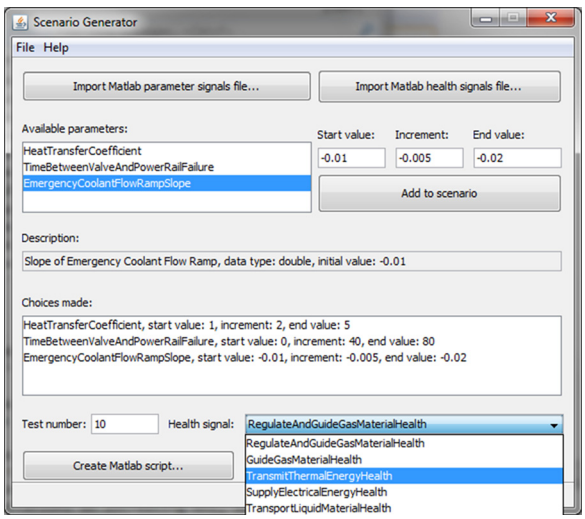

Fig. 12 User interface for specifying a set of FFIP simulation scenarios

driving the coolant pumps to minimum rotations per minute (Slope) is one of: [−0.01, −0.015, −0.02]. The simulation will be run with each combination of parameter values, resulting in 27 simulation runs. For each of these runs, the value of one signal is logged for each step of the discrete time simulation; in Fig. 12, the function health of "transmit thermal energy" from fuel rods to coolant is selected. The process is repeated so that the Temperature of the fuel rods is logged, resulting in 27 temperature curves and information of the worst case health status of each curve; in Sec. 6, the health status is used to group the curves onto separate charts for readability.

The algorithm for generating the set of parameterized FFIP simulations is shown in Fig. 13. The information entered through the user interface is stored into a list, with one element for each parameter; in this case the parameters are HTC, Delay, and Slope. A CreateLoop function is called, which iterates through the list in such a way that every combination of parameters is covered. This is accomplished by a recursive function call shown with a bold dashed arrow. This design ensures that the no-branch of decision 1 is taken exactly once for each combination of parameter values, resulting in an entry in the Matlab script for running the FFIP simulation with those values. Recursion was used to improve readability and compactness of code [41,42].

6 Simulation Results

In the initial critical event scenario, emergency power for coolant pumps was cut off before the pressure shockwave occurred. In the user interface, three different values are selected for HTC, the slope for driving coolant pumps to minimum rotations per minute (Slope) and the time delay between the unintentional pressure control valve closure and power rail failure (Delay). 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 first phase of the simulation will result in 27 fuel rod temperature curves, one for running the simulation with 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. Accordingly, the curves for fuel rod temperature are grouped in three graphs: the simulation runs during which this function's health was always healthy (Fig. 14), the runs during which the health was never worse than degraded (Fig. 15) and the runs during which this function was lost (Fig. 16). When several curves were overlapping and visually indistinguishable on the

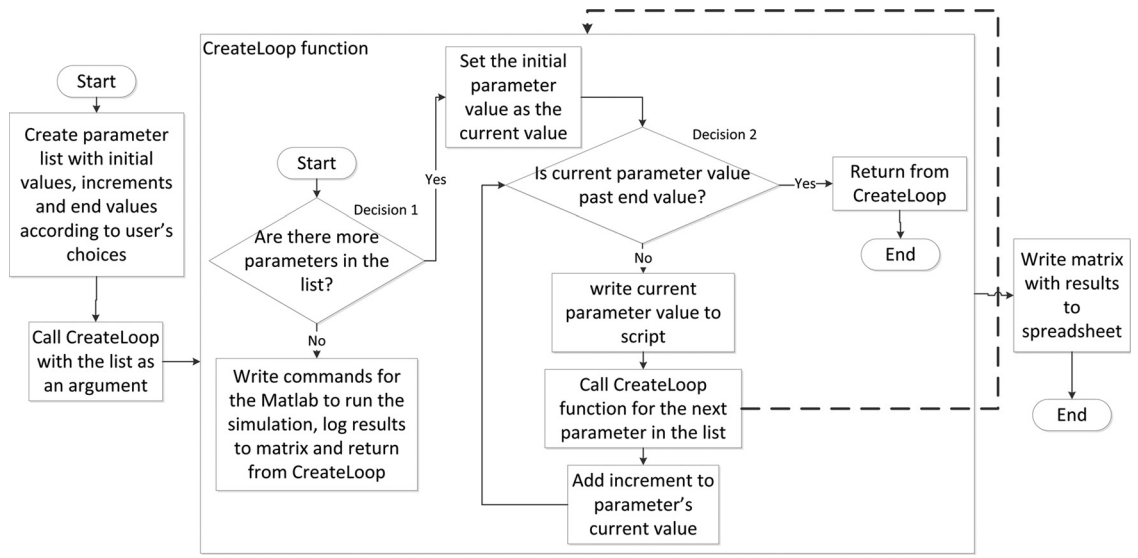

Fig. 13 Algorithm for generating the set of parameterized FFIP simulations

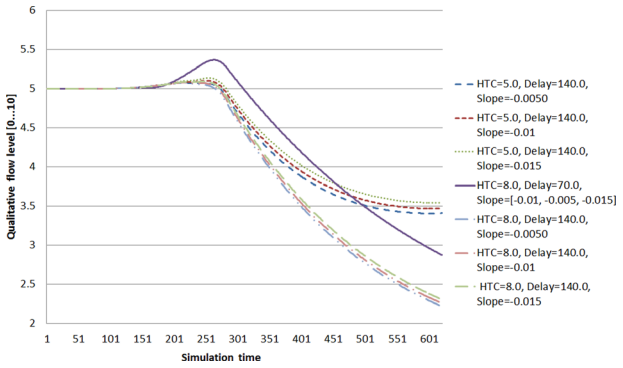

Fig. 14 Temperature of fuel rods in first phase FFIP simulation scenarios with parameter values resulting in healthy FFL verdicts

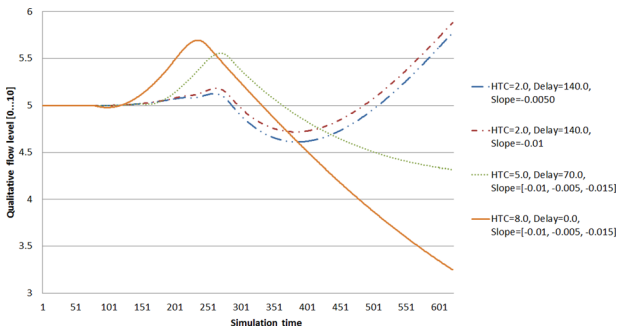

Fig. 15 Temperature of fuel rods in first phase FFIP simulation scenarios with parameter values resulting in degraded FFL verdicts

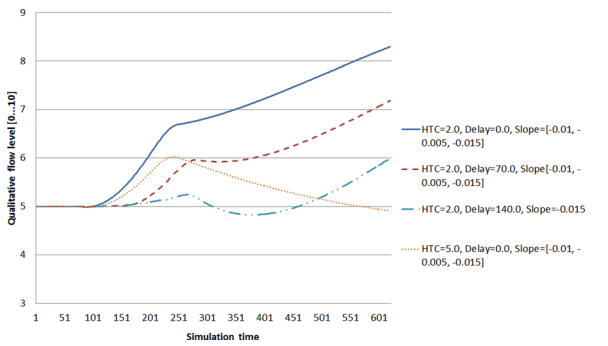

Fig. 16 Temperature of fuel rods in first phase FFIP simulation scenarios with parameter values resulting in lost FFL verdicts

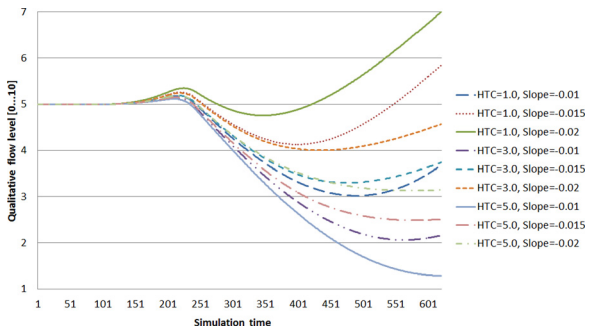

Fig. 17 Temperature of fuel rods in second phase FFIP simulation: emergency power present

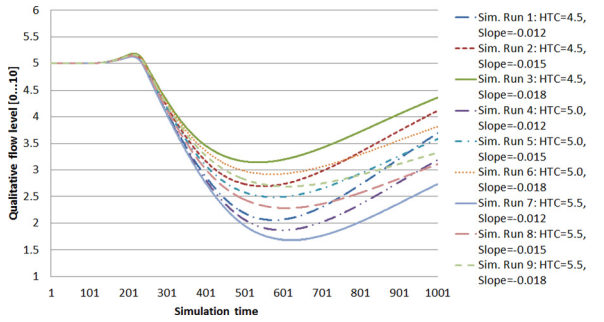

Fig. 18 Temperature of fuel rods in third phase FFIP simulation scenarios with narrowed ranges for parameter values

graph, the parameter values corresponding to them are shown in square brackets in the legend; for example, in Fig. 16, the temperature curves for HTC = 2 and Delay = 0 were identical for all values of Slope.

It is clear that better performance was obtained when the power rail failure occurred later, since coolant flow was maintained for a longer time. This indicates that from the perspective of fuel rod health, steam removal from the core is more important than an acute drop in reactor power, which could be achieved by stopping the pumps. A very low value for the HTC indicated stability problems regardless of the other values, and this is a physical design constraint for this reactor type. A very high value for HTC provided consistently stable behavior, although a temporary overheating of fuel rods is possible; however, it might be physically impossible to realize this design. With medium values of HTC, fuel rod cooling depends primarily on the availability of power for the coolant pumps and secondarily on having a smaller slope for the ramp that drives down the rotations per minute of the coolant pumps. Based on this analysis, it is concluded that the availability of power for coolant pumps must be ensured with emergency power, so a different critical event scenario, in which the emergency power is not cut off, is selected as the basis for the next set of simulation runs. In this case, the time delay between the shockwave and power rail failure does not significantly influence the results, so this parameter is excluded from further study.

The second phase of FFIP simulation focuses on the combinations of HTC and ramp slope, resulting in nine automatically executed simulations. From Fig. 17, it is clear that when emergency power is present, stable performance can be achieved even with lower HTC values when a gentler slope is used for the ramp. However, a longer ramp implies a slower decrease of reactor power during emergency shutdown, so a third simulation run is performed by narrowing the parameter values to the range in which these two parameters are likely to be optimized. The results in Fig. 18 are given as a starting point for detailed design, which will refine the conceptual design through specific decisions on component types, subsystem design and control algorithm design. One decision that emerges already from the simulation of the BWR system design is that availability of electric power for coolant pumps must be ensured by emergency power supplies even when the power rail is lost.

In every scenario on Fig. 18, the fuel rod cooling is performed satisfactorily during the acute situation requiring emergency shutdown of the reactor. The simulation time is set to a value that is of interest for the emergency shutdown safety function. It is unclear if the temperature would remain below nominal if the simulation time were to be increased. In the case study, it was stated that the model covers only emergency shutdown functionality and is not aimed at studying residual heat removal, which has a much longer time constant; this would require additional detail to the system model and a different set of simulation runs. The emergency shutdown of the reactor is used in this paper to demonstrate the application of FFIP automation framework.

7 Conclusions and Future Work

The design and failure analysis of large complex systems presents numerous challenges. While the FFIP framework developed previously was intended to meet some of these challenges, the extensions presented in this paper greatly increase the ability of this analysis approach to be used as a tool in the early concurrent design and risk assessment processes. The extensions to the FFIP framework presented here enable the study of the reliability of a range of design alternatives for achieving the specified functionality. For alternatives that can be expressed in terms of parameters of component behavior or timing of critical events, the scalability of the approach and supporting tools to a larger number of simulation runs is only limited by computing power and human capability of understanding large data sets. The latter problem is mitigated by the possibility of filtering and sorting the simulation runs in terms of the health status of a specific function, but further research on user interface design will improve the feasibility of the approach for complex industrial-scale systems. The behavior of electromechanical components is described with first order linear approximations in this case study, but they are implemented as Simulink blocks, so the approach permits more sophisticated modeling if it is deemed appropriate in the early concept design phase.

The investigation of a range of designs under a range of critical event scenarios is restricted to alternatives that can be expressed by varying values of parameters. The availability of emergency power could be controlled manually by selecting different simulation scenarios, but currently it is not possible to cover more fundamental differences such as different types or placement of sensors or entirely different software algorithms. This can be achieved with further work on integrating a feature modeling capacity to the configuration flow graph, so that every valid configuration of the feature model will undergo FFIP simulation in order to filter out unreliable alternatives. The automated FFIP simulation presented in this paper solves many algorithmic and technical challenges related to the generation and simulation of every valid configuration.

References

[1] Thramboulidis, K., 2005, "Model-Integrated Mechatronics—Toward a New Paradigm in the Development of Manufacturing Systems," IEEE Trans. Ind. Inf., **1**(1), pp. 54–61.

[2] Amerongen, J. V., 2003, "Mechatronic Design," Mechatronics, **13**, pp. 1045–1066.

[3] Weilkiens, T., 2007, *Systems Engineering With SysML/UML: Modeling, Analysis, Design*, Morgan Kaufmann, San Francisco, CA.

[4] Kurtoglu, T., and Tumer, I. Y., 2008, "A Graph-Based Fault Identification and Propagation Framework for Functional Design of Complex Systems," J. Mech. Des. **130**(5), p. 051401.

[5] Kurtoglu T., Tumer, I. Y., and J. D., 2010, "A Functional Failure Reasoning Methodology for Evaluation of Conceptual System Architectures," Res. Eng. Des., **21**(4), pp. 209–234.

[6] Jensen D., Tumer, I. Y., and Kurtoglu, T., 2008, "Modeling the Propagation of Failures in Software-Driven Hardware Systems to Enable Risk-Informed Design," ASME IMECE.

[7] Jensen D., Tumer, I. Y., and Kurtoglu, T., 2009, "Design of an Electrical Power System Using a Functional Failure and Flow State Logic Reasoning Methodology," Prognostics and Health Management Society.

[8] Tumer, I. Y., and Smidts, C. S., 2010, "Integrated Design and Analysis of Software-Driven Hardware Systems," IEEE Trans. Comput., Special Issue on Science of Design of Safety-Critical Systems, **60**(8), pp. 1072–1084.

[9] Papkonstantinou, N., Sierla, S., Jensen, D. C., and Tumer, I. Y., 2011, "Capturing Interactions and Emergent Failure Behavior in Complex Engineered Systems at Multiple Scales," International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, ASME, Washington, DC.

[10] Vesely, W. E., Goldberg, F. F., Roberts, N. H., and Haasi, D. F., 1981, *The Fault Tree Handbook*, U.S. Nuclear Regulatory Commission.

[11] Price, C. J., and Taylor, N. S., 1998, "FMEA for Multiple Failures," Reliability and Maintainability Symposium, Anaheim, CA.

[12] Hu, T., Yu, J., and Wang, S., 2009, "Research on Complex System FMEA Method Based on Functional Modeling," Reliability, Maintainability and Safety, Chengdu.

[13] Mauri, G., McDermid, J. A., and Papadopoulos, Y., 1998, "Extension of Hazard and Safety Snalysis Techniques to Address Problems of Hierarchical Scale," IEE Colloquium on Systems Engineering of Aerospace Projects, Digest No. 1998/249.

[14] Papadopoulos, Y., Parker, D., and Grante, C., 2004, "Automating the Failure Modes and Effects Analysis of Safety Critical Systems," High Assurance Systems Engineering, Tampa, FL.

[15] Pasquale, T., Rosaria, E., Pietro, M., and Antonio, O., 2003, "Hazard Analysis of Complex Distributed Railway Systems," Reliable Distributed Systems, Florence, Italy.

[16] Schreiber, S., Schmidberger, T., Fay, A., May, J., Drewes, J., and Schnieder, E., 2007, "UML-Based Safety Analysis of Distributed Automation Systems," Emerging Technologies and Factory Automation, Patras, Greece.

[17] Stamatelatos, M., and Apostolakis, G., 2002, "Probabilistic Risk Assessment Procedures Guide for NASA Managers and Practitioners," NASA, Safety and Mission Assurance.

[18] Perera, J., and Holsomback, J., 2004, "Use of Probabilistic Risk Assessments for the Space Station Program," Aerospace Conference.

[19] Sturges, R. H., Kilani, M., and OShaughnessy, K., 1996, "Computational Model for Conceptual Design Based on Extended Function Logic," Artif. Intell. Eng. Des. Manuf. J., **10**, pp. 255–274.

[20] Szykman, S., Sriram, R. D., Bochenek, C., and Racz, J., 1998, "The NIST Design Repository Project," *Advances in Soft Computing—Engineering Design and Manufacturing*, Springer-Verlag, London.

[21] Terpenny, J., and Mathew, D., 2004, "Modeling Environment for Function-Based Conceptual Design," Design Automation Conference/IDETC/CIE 2004, Salt Lake City, UT.

[22] Sasajima, M., Kitamura, Y., Ikeda, M., and Mizoguchi, R., 1996, "A Representation Language for Behavior and Function: FBRL," Expert Syst. Appl., **10**(3/4), pp. 471–479.

[23] Qian, L., and Gero, J. S., 1996, "Function-Behaviour-Structure and Their Roles in Analogy-Based Design," Artif. Intell. Eng. Des. Anal. Manuf., **10**, pp. 289–312.

[24] Huang, E., Ramamurthy, R., and McGinnis, L., 2007, "System and Simulation Modeling Using SysML," Conference on Winter simulation, IEEE Press, Washington, DC.

[25] Tumer, I. Y., and Stone, R. B., 2003, "Mapping Function to Failure During High-Risk Component Development," Res. Eng. Des., **14**(1), pp. 25–33.

[26] Wang, K.-L., and Jin, Y., 2002, "An Analytical Approach to Functional Design," International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Montreal, Canada.

[27] Hoyle, C., Tumer, I. Y., Mehr, A. F., and Wei, C., 2009, "Health Management Allocation During Conceptual System Design," J. Comput. Inf. Sci. Eng., **9**(2), p. 021002.

[28] Simpson, T. W., Peplinski, J., Koch, P. N., and Allen, J. K., 2001, "Metamodels for Computer-Based Engineering Design: Survey and Recommendations," Eng. Comput., **17**(2), pp. 129–150.

[29] Box, G. E. P., and Wilson, K. B., 1951, "On the Experimental Attainment of Optimum Conditions," J. R. Stat. Soc. Ser. B (Methodol.), **13**(1), pp. 1–45.

[30] Guo, J., and Du, X., 2010, "Reliability Analysis for Multidisciplinary Systems With Random and Interval Variables," AIAA J., **48**(1), pp. 82–91.

[31] Zang, C., Friswell, M. I., and Mottershead, J. E., 2005, "A Review of Robust Optimal Design and Its Application in Dynamics," Comput. Struct., **83**(4–5), pp. 315–326.

[32] Deb, S., Pattipati, K. R., Raghavan, V., Shakeri, M., and Shrestha, R., 1995, "Multisignal Flow Graphs: A Novel Approach for System Testability Analysis and Fault Diagnosis," IEEE Aerospace and Electronics Systems Magazine, pp. 14–25.

[33] Kurtoglu, T., and Tumer, I. Y., 2008, "A Risk-Informed Decision Making Methodology for Evaluating Failure Impact of Early System Designs," 2008 International Design Theory and Methodology Conference, IDETC/CIE2008, Brooklyn, NY.

[34] de Kleer, J. K., Lukas, K., Liu, J., Price, B., Do, M., and Zhou, R., 2009, "Continuously Estimating Persistent and Intermittent Failure Probabilities," SafeProcess 2009.

[35] Forbus, K., 1984, "Qualitative Process Theory," Artif. Intell., **24**, pp. 85–168.

[36] Weld, D., and de Kleer, J., 1987, *Readings in Qualitative Physics*, Morgan Kauffman, San Francisco, CA.

[37] Struss, P., 1988, "Mathematical Aspects of Qualitative Reasoning," Int. J. Artif. Intell. Eng., **3**(3), pp. 156–169.

[38] Kuipers, B. J., 1986, "Qualitative Simulation," Artif. Intell., **29**(3), pp. 289–338.

[39] Stone, R., and Wood, K., 2000, "Development of a Functional Basis for Design," J. Mech. Des., **122**(4), pp. 359–370.

[40] Abagyan, L. P., Golubev, V. I., Golyaev, N. D., Zvonarev, A. V., Koleganov, Y. F., Nikolaev, M. N., and Orlov, M. Yu., 1968, "Propagation of Neutrons in Uranium dioxide II. Doppler Effect in U238," At. Energy, **25**(4), pp. 1090–1094.

[41] Davis, M., Sigal, R., and Weyuker, E. J., 1994, *Computability, Complexity, and Languages*, Morgan Kaufmann, San Francisco, CA.

[42] Gaffney, J. E., and Davis, C. F., 1988, "An Approach to Estimating Software Errors and Availability," Eleventh Minnowbrook Workshop on Software Reliability.