A Graph-Based Fault Identification and Propagation Framework for Functional Design of Complex Systems

Article *in* Journal of Mechanical Design · May 2008

DOI: 10.1115/1.2885181

CITATIONS
197

READS
1,608

2 authors:

Tolga Kurtoglu
Palo Alto Research Center
67 PUBLICATIONS **2,186** CITATIONS

SEE PROFILE

Irem Y. Tumer
Oregon State University
283 PUBLICATIONS **3,886** CITATIONS

SEE PROFILE

A Graph-Based Fault Identification and Propagation Framework for Functional Design of Complex Systems

Tolga Kurtoglu
Research Scientist
Missio Critical Technologies,
NASA Ames Research Center,
MS 269-3, Moffett Field, CA 94035

Irem Y. Tumer[1]
Associate Professor
School of Mechanical, Industrial and
Manufacturing Engineering,
Oregon State University,
204 Rogers Hall,
Corvallis, OR 97331

In this paper, the functional-failure identification and propagation (FFIP) framework is introduced as a novel approach for evaluating and assessing functional-failure risk of physical systems during conceptual design. The task of FFIP is to estimate potential faults and their propagation paths under critical event scenarios. The framework is based on combining hierarchical system models of functionality and configuration, with behavioral simulation and qualitative reasoning. The main advantage of the method is that it allows the analysis of functional failures and fault propagation at a highly abstract system concept level before any potentially high-cost design commitments are made. As a result, it provides the designers and system engineers with a means of designing out functional failures where possible and designing in the capability to detect and mitigate failures early on in the design process. Application of the presented method to a fluidic system example demonstrates these capabilities. [DOI: 10.1115/1.2885181]

Keywords: failure prevention, reliability engineering, risk based design, functional modeling, failure analysis, concept generation, configuration flow graph, simulation-based design, model based design

1 Introduction

The complexity of modern world engineered systems is growing constantly. New technologies are creating the potential for higher levels of integration, multifeedback control loops, and resulting systems contain a larger number of dynamically interacting components, relations among which are increasingly nonlinear. These interactions become extremely difficult for any one individual to comprehend, analyze, or synthesize [1]. Often times, it is beyond the grasp of designers to consider and model all potential system states and associated risks, and for operators to handle critical events, abnormal situations, and deviations safely and effectively. This complexity, in turn, leads to unexpected behaviors and consequences, some of which have proven to be fatal, such as the two shuttle disasters.

A key technical challenge in developing such complex systems is to ensure that the individual components and technologies are reliable, effective, and low cost, resulting in turn in safe, reliable, and affordable systems. To ensure safety and reliability, the subsystem and component functionality, decisions, and knowledge have to be incorporated into the product life cycle as early as possible. Furthermore, formal tools and methodologies need to be in place to allow program managers and lower-level designers to formulate a clear understanding of the impact of the decisions in the early design phases. Most existing failure analysis and risk assessment efforts focus on system diagnostics and utilize reasoning tools that require very detailed, high-fidelity models of system components in order to infer faulty system behavior and its consequences. At the early design stages, however, selection of specific components has not been made, and hence such detailed models of system components and design parameters are not yet available. Instead, the designs are represented using low-fidelity, high-level models of intended functionality. In order to make early design trades, the focus should be kept on a system's functional requirements, and hence it is crucial to be able to reason at the functional level and identify what functions are likely to fail or degrade and what the overall effect of loss of these functions will be on system behavior and performance.

In this paper, the functional-failure identification and propagation (FFIP) framework is introduced that enables these capabilities. FFIP allows the designers to proactively analyze the functionality of the systems early in the design process, understand functional failures and their propagation paths, and determine what functions are lost, what the impact to the overall system will be, and what redundancies and safeguards should be added. The main advantage of the method is that it permits the analysis of functional failures and fault propagation at a highly abstract system topology level before any potentially high-cost design commitments are made. This influence on system design supports decision making early in the design process, guides the designers to eliminate failure through exploration of system components and their functionality, and facilitates the development of more reliable system configurations.

The task of FFIP is to estimate potential faults and their propagation paths under critical event scenarios using behavioral simulation. The framework is based on combining hierarchical system models with behavioral simulation and qualitative reasoning. There are three major components to the FFIP framework: the graphical system model, the behavioral simulation, and the reasoning scheme called the function-failure-logic (FFL). The graph-based modeling approach provides a coherent, consistent, and formal schema to capture function-configuration-behavior architecture of a system at an abstract level and facilitates the assessment of potential functional failures and the resulting fault propagation paths through the FFL reasoner that translates the dynamics of the system into functional-failure identifiers.

There are several unique aspects of the developed FFIP framework. First, the FFL employed by the reasoner is a novel approach in that it allows to reason about the system state at a very high,

[1]Corresponding author.

Contributed by the Design Automation Committee of ASME for publication in the JOURNAL OF MECHANICAL DESIGN. Manuscript received November 30, 2006; final manuscript received August 12, 2007; published online March 25, 2008. Review conducted by Kemper Lewis.

	Modeling					Reasoning					Application		
	Model-based	Event-based	Function-based	Failure-based	Models System Dynamics and Component Interactions	Forward Logic	Backward Logic	Dependent on Expert Knowledge	Reasoning About Multiple Faults	Reasoning About Fault Propagation	Real-Time Application	Detailed Design Application	Conceptual Design Application
Failure Modes and Effects Analysis (FMEA)				X		X		X				X	
Fault Tree Analysis (FTA)		X		X			X	X				X	
Probabilistic Risk Assesment (PRA)		X		X		X	X	X				X	
Model-Based Diagnosis (MBD)	X	X		X	X		X		X	X	X		
Fault Propagation Graphs				X		X	X	X	X	X	X	X	
Function Failure Design Method (FFDM)			X	X								X	X
Functional Failure Identification and Propagation	X	X	X	X	X	X			X	X		X	X

Fig. 1 Comparison of FFIP to current fault assessment methods

functional level (such as "supply liquid"). This goes beyond the traditional reasoning tools that use low-level sensor values to infer failed components given the signs of errors in system operation, but are short of quantifying the impact of failures on functional performance under off-nominal conditions. Second, the integration of the qualitative reasoning scheme of the FFL with a behavioral simulation enables automatic computation of functional failures and fault propagation paths directly from physical behavior and component interactions of the system. This brings a much needed formalism to fault assessment and is different from approaches that rely solely on expert opinion to assess failure and its consequences. Third, FFIP captures various nonlinear aspects of fault propagation, most notably the deviation of fault propagation paths from strict structural and functional connectivity. Finally, the developed framework is able to identify functional failures that do not result from direct component failures but rather from global component interactions. Application of the presented method to a fluidic system example demonstrates these capabilities in evaluating and assessing functional-failure risk of physical systems during conceptual design.

2 Related Work

2.1 Review of Fault Assessment and Management Techniques. The aerospace industry currently employs three major reliability tools and methods: failure mode and effect analysis (FMEA), fault tree analysis (FTA), and probability risk assessment (PRA). FMEA [2] is a method that systematically examines individual system components and their failure mode characteristics to assess risk and reliability. FTA [3] is performed to capture event paths from failure root causes to top-level consequences. PRA [4] is a method used for quantification of failure risk by answering three questions: what can go wrong, how likely is it to happen, and what are the consequences [5]. PRA combines a number of fault/event modeling techniques, such as master logic diagrams, event sequence diagrams, and fault trees, and integrates them into a probabilistic framework to guide decision making during design.

Apart from these reliability based design methods, diagnostic reasoning has been a central theme in fault assessment and management. Diagnostic reasoning approaches share a common process in which a system is monitored and a comparison is performed on observed and expected behaviors of the system to detect anomalous conditions. The major approaches to fault modeling and diagnostic reasoning include expert systems, simulation and model-based diagnosis methods, and data classification techniques. Expert systems [6] are extensively used in diagnosis, where knowledge acquired from human experts is formulated in different ways, such as "if-then" rules or decision trees [7,8]. Artificial intelligence-based approaches to diagnosis, on the other hand, rely mostly on qualitative knowledge to predict the behavior of a system [9]. When observations disagree with the predicted behavior, some diagnostic technique is initiated to identify the faults. The broadest category for diagnostic reasoning is model-based diagnosis (MBD) [10–12]. At NASA, for example, Livingstone [13] and its extension L2 [14] are two of the most notable tools that utilize algorithms adapted from MDB. In these approaches, the state estimation is a search over the transitions of component configurations to find a global configuration consistent with sensor measurements. Besides the model-based approaches, some researchers have employed fault propagation graphs as the system model for diagnostic reasoning [15–20]. Among those, directed graphs [21] are one of the techniques used to analyze component dependencies and fault propagation. Multisignal flow graphs developed by Deb et al. [22] is another comprehensive methodology to model cause-effect dependencies of complex systems. This work has culminated itself into the testability engineering and maintenance system (TEAMS) [23] design tool that is used for automated test sequencing and testability analysis in industry and the government. Finally, in cases where physical cause-effect relationships are difficult to model in analytical form, statistical and probabilistic classification methods are applied [24,25].

One of the critical shortcomings of the methods discussed above is the difficulty in applying them in the early stages of design. In order to address this need, prior work has introduced methods for integrating early risk assessment and management tools into the complex system design process [26–29]. Most notably, the authors have presented the details of a calculation which links a product's functional model to potential failures [30,31]. This method, called the function failure design method (FFDM), promotes early identification of potential failures by linking them to product functions. FFDM defines a matrix based relationship between a system's functions and its failure modes. This relationship is derived from documented, historical data and is formalized with the help of two standardized taxonomies, one describing functions [32] and the other failure modes [33] of complex systems. FFDM provides a starting point for determining the likelihood of system failure based on a set of functions that the system has to deliver. Using this method, designers can analyze potential functional failures before any component selection is made [34,35]. To extend FFDM, Grantham-Lough et al. [36] developed the risk in early design (RED) method to formulate a functional-failure likelihood and consequent risk assessment. This approach classifies high-risk to low-risk function failure combinations and provides designers a tool that can be used to qualitatively rank/order functional failures and their consequences during conceptual design.

2.2 Comparison of FFIP to Current Fault Assessment and Management Methods. Figure 1 presents a comparison of the FFIP methodology to existing methods including FMEA, FTA, PRA, MBD, fault propagation graphs, and FFDM. FMEA is a bottom-up approach that follows forward logic to determine critical component failures and their consequences. FMEA analysis starts with decomposition of the system into subsystems and finally into individual components. Ways in which each component can potentially fail (failure mode(s)) are then recorded and as-

sessed separately to determine what effect they have at the component level, and then at the system level. FTA is an event-oriented analysis that starts with identification of a high-level failure event. A backward logic is then followed to drive contributing events that could lead to the occurrence of immediate higher-level events. At the end, the analysis presents the chain of events combined with logical gates in a tree structure. Model-based diagnostic and fault propagation graphs are concerned with run-time system diagnostics, testability, and observability. These methods share an after-the-fact approach that looks at effects and traces them back to the causes of those effects to identify faults.

While these methods greatly benefit failure and risk analysis during system development, their applicability during conceptual system design is highly limited. Function-based modeling has shown significant promise in enhancing the conceptual design process. One of the main objectives of the work presented in this paper is to move function-based modeling into the model-based reasoning realm for designing robust and reliable system configurations. This requires an approach that combines fault, event, failure, and model-based information under a unified framework. FFIP achieves this by integrating hierarchical system models of function, configuration, and behavior. This integration enables the computation of specific component interactions and resulting failure identifiers and eliminates the dependency on expert input. In this regard, the FFIP methodology is less prone to criticism of subjectivity compared to FMEA, FTA, and fault propagation graphs. Moreover, FFIP reasons about the propagation of faults by capturing fault propagation paths formally. FMEA and FFDM do not offer this capability, and FTA only captures linear relationships. On the other hand, compared to FFDM, FFIP is applicable to a variety of systems and it is not constrained by a database of documented, historical failure data. The details of the FFIP framework are explained in the next section.

3 FFIP Framework

The overall goal of this research is to develop a formal framework to be used during conceptual design for identifying functional failures of complex systems and their propagation. To achieve this goal, our method combines hierarchical system models with behavioral simulation and qualitative reasoning.

At the conceptual stage of design, the system's functional requirements may be defined, but detailed models of system components and their design parameters are not yet available. Instead, the design is expressed as an interconnectivity of elemental abstract components that is often defined as the topology or the configuration of a system. The topology imposes a certain structure on the system that permits delivery of desired functionality through specific component interactions or behavior. In order to integrate a fault prevention and management ability at this stage, a modeling paradigm that is capable of representing the desired functionality of the individual components, their structure, as well as their interactions is required. Accordingly, our framework represents system function, configuration, and behavior by an interrelated array of graph-based models. This modeling approach is schematically illustrated in Fig. 2. This approach provides a coherent, consistent, and formal schema to capture function-configuration-behavior relations of a system at an abstract level and facilitates reasoning about potential system faults.

The detailed architecture of the developed FFIP framework is presented in Fig. 2. The task of FFIP is to estimate potential faults and their propagation under critical event scenarios using behavioral simulation. In this framework, the physical system is represented by three sets of graphs and the mapping relations between them. The graph-based system model constitutes an environment where knowledge about system function, behavior, and control is integrated and used to automatically predict functional failures. There are three major components to the FFIP framework: the graphical models, the behavioral simulation, and the reasoning

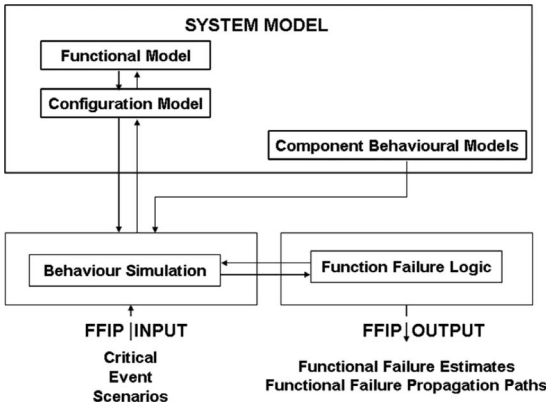

Fig. 2 The architecture of the FFIP framework

through the FFL. In the following sections, these three components are explained in detail with their interactions and specific roles.

3.1 System Modeling Using Graphical Representations. In FFIP, system *function* is represented using function structures [37]. A function structure is a graphical, form-independent representation of a system that shows the decomposition of the overall system function into smaller, more fundamental subfunctions. The subfunctions are connected by energy, material, and signal flows that they operate on. Overall, a functional model represents how input flows of a system are transformed into output flows. This approach to function-based modeling relies on verb-noun descriptions of elemental functions and a canonical list of flow types based on a standardized taxonomy called the Functional Basis [38]. Using the terms defined in the Functional Basis, designers can generate a model for the actual or desired functionality of a system. The *structure* in FFIP, on the other hand, is captured using configuration flow graphs (CFGs) [39]. A CFG strictly follows the functional topology of a system and maps the desired functionality into the component configuration domain. In a CFG, nodes of the graph represent system components, whereas arcs represent energy, material, or signal flows between them. For flow naming, the Functional Basis terminology is adopted, while the components of the graph are named using a taxonomy of standard electromechanical components [40]. The component types in a CFG can be thought of as generic abstractions of common component concepts (valve, tank, junction, dc motor, battery, etc.) The CFG is a specific implementation of the topology or the configuration of a system. Figure 3 shows an example of a functional model, a configuration flow graph, and a simplified system schematic of a hold-up tank system. The construction of a functional model (FM) and the corresponding CFG captures a direct mapping between the functional and the structural architecture of a system [41]. Each mapping represents a transformation that shows how a functional requirement was addressed in the actual design by the use of a specific component concept. In order to extract these mappings in a consistent manner, the flow information, i.e., the fact that the two graphs share the same flow types, is utilized. Accordingly, by following the "flow paths," one can define boundaries that isolate the mapping between functional nodes of a function structure and component nodes of a CFG. For example, in the hold-up tank example of Fig. 3, the component "valve" addresses functions "import liquid," and "guide liquid." Similarly, the component "tank" provides "store liquid" and "supply liquid" functions in the system. Capturing this mapping between functionality and component configuration of a system is crucial for accurately reasoning about failures at a functional level.

Finally, the *behavior* of the system is represented using a component oriented modeling approach. The approach involves the

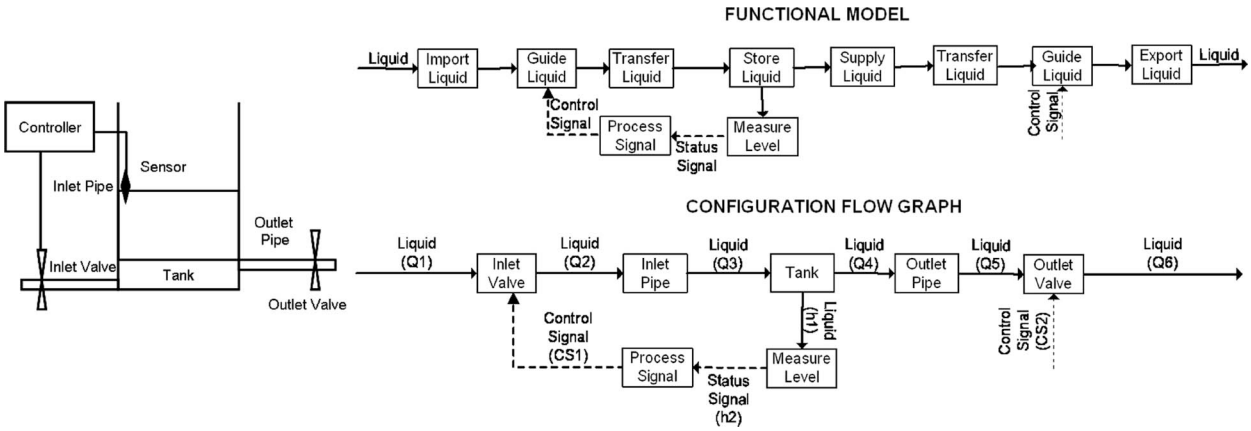

Fig. 3 The FM, the schematic, and the CFG of a hold-up tank

development of high-level, qualitative behavior models of system components at various discrete nominal and faulty modes. The transitions between these discrete modes are defined by mode transition diagrams and the component behavior in each mode is derived from input-output relations and underlying first principles. These modular, reusable component behavior models follow the form of CFG. Accordingly, state variables critical to the system behavior are incorporated into the representation by associating them with their respective (CFG) flows. For example, a designer may track the flow rate, pressure, or temperature of a fluid flow. The individual models describe the input-output relationships between these state variables in each component mode. In the behavioral modeling approach taken here, components are assumed to have a finite number of modes (i.e., they are assumed to be finite-state machines). During system operation, a component may switch between these discrete modes. The transitioning between these modes is caused by *events* either through control commands issued, human interactions, environmental effects, or due to the component malfunctioning. For example, a valve can transition from a "nominal on" mode to a "nominal off" mode as a result of a command generated by the system control. Similarly, a failure mechanism can cause the valve to transition to a certain failure mode, such as "failed open" or "failed closed." The dynamic behavior of the component in each mode is governed by a different set of physical laws and mathematical relations, and is therefore defined separately. For example, output pressure and output flow rate for a valve in nominal on mode remains equal to input pressure and input flow rate (assuming no pressure drop), whereas they both become zero for a valve in failed closed mode.

3.2 Behavioral Simulation. The objective of the simulation is to determine the system behavior under certain conditions. These conditions are represented by the occurrence of events that cause specific component mode transitions. During the simulation, both the discrete component modes and the set of system state variables need to be tracked. Accordingly, the overall system state at time t is described by

$$X(t) = \Theta(c(t), \mathbf{v}(\mathbf{t})) \tag{1}$$

where $X(t)$ is the overall system state, $c(t)=[c_1, c_2, c_3, \ldots . c_N]$ is a vector of discrete component modes where each component $c=1, \ldots . N$ (N: number of components in the system) assumes a discrete mode from its own set of M modes $c_i=(c_{i1}, c_{i2}, c_{i3}, \ldots . c_{iM})$, and $v(t)=[v_1, v_2, v_3, \ldots . , v_K]$ is a vector of system state variables.

During conceptual design, the system state variables are not known quantitatively. Therefore, these continuous variables are discritized into a set of qualitative values. The vector $\mathbf{v}(\mathbf{t})$ then defines these qualitative values for each state variable v_i from a set of P possible values $v_i=(v_{i1}, v_{i2}, v_{i3}, \ldots . , v_{iP})$. For example, a liquid flow rate variable may take on values from the set of {zero, low, nominal, and high}. Similarly, a control signal variable may have values of {no signal, on, off}, etc.

The evolution of the overall system state (Eq. (1)) is traced by the use of the system CFG. The CFG provides a graph-based, formal language for individual components models to be integrated into a system-level behavioral model. Accordingly, the values of component modes and system state variables are defined through the nodes and arcs of the CFG. The overall system behavior is simulated and the physical system state is captured by changes of the system CFG.

To start the simulation, the modes of individual components (nodes) in the CFG are initialized along with the values of system state variables associated with input flows (arcs). Then, the state of the system is simulated by solving the continuous-time system in the intervals between discrete events. Each time step propagates values of certain state variables depending on the mode of components, the behavioral models in that particular mode, and the defined component constraint relations. When an event occurs, the continuous-time simulation is stopped, and the corresponding component mode transition is executed. Using this scheme, critical events, consequences of which are investigated can be inserted into the simulation at any time step. Following this approach, the simulation may be run over a certain number of time steps, or until the system reaches a prescribed end state.

The modeling scheme used in FFIP is common for modeling hybrid, event driven, complex systems. For example, function-behavior-structure (FBS) paths developed by Qian and Gero [42] constitute a method that presents a formalism to represent function, structure, and behavior of systems. In this technique, relations among function, behavior, structure, and processes are utilized to define FBS paths, which are then used to retrieve design information to conduct analogy-based design. However, FBS paths only model nominal behavior of components. FFIP models both nominal and faulty behaviors of components and failure conditions of functions so that functional failures and their propagation can be assessed from a system model using behavioral simulation. Similar to FFIP, Livingstone [13], L2 [14], and SIMPRA [43] diagnostics and risk assessment tools also describe the system behavior using component models having finite number of nominal and failure modes and by transitions between these modes. These approaches vary in the way they perform the simulation task. SIMPRA relies on MATLAB's Simulink tool and the theory of probability dynamics, and Livingstone and L2 use constraint propagation algorithms. There are, however, other techniques that can be used to carry out the behavioral simulation. Algorithms from qualitative process theory [44] or QSIM [45]

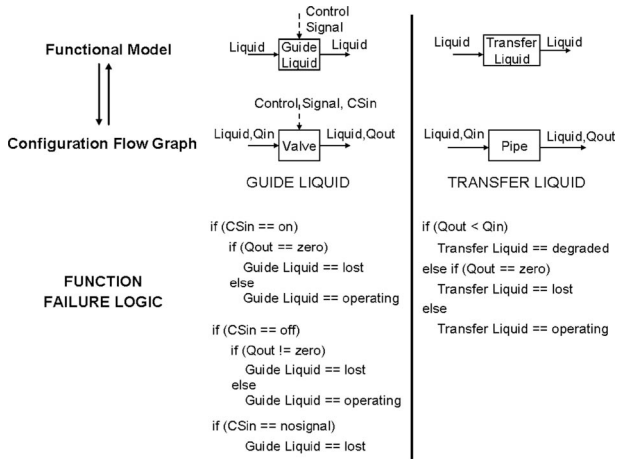

Fig. 4 FFL for guide liquid and transfer liquid functions

could potentially also be used to perform the kind of behavioral simulation presented here.

FFIP is not mainly concerned with simulating the dynamic behavior of physical systems. That process is fairly well understood and documented in the literature. The main novelty of the FFIP framework is enabling the reasoning at the high, functional level by monitoring the relationship between behavioral dynamics and the function of components. The reasoning scheme is explained in the next section.

3.3 Reasoning Using the FFL. The assessment of potential functional failures and fault propagation paths are made through a reasoner that translates the state changes in the system configuration graph into *functional failures*. The FFL employed by the reasoner is developed upon a philosophy that emphasizes qualitative functional reasoning. As stated in Ref. [1], "we cannot define all the things that might go wrong behaviorally, however, we can, in principle, determine what it means for the system to function properly." Thus, the FFL defines a set of form-independent system function models that describe conditions under which functions deviate from their intended operation. In FFL, each system function is classified as operating, degraded, or lost.

The goal of the FFL reasoner is to determine the state of each system function (i.e., whether it is operational, degraded, or lost) at any time t given the physical state of the system $X(t) = \Theta(c(t), \mathbf{v}(\mathbf{t}))$ (Eq. (1)). The simulation feeds the physical state of the system to the FFL reasoner at the end of each time step and the state of each system function is evaluated at these discrete points.

FFL allows the assessment of the operability of a function to be made based on the values of the input and output state variables of the CFG that corresponds to the component by which the function is realized. Therefore, capturing the mapping between the FM (function) and the CFG (behavior) is fundamental to the employment of the FFL. Figure 4 presents two rules from the developed FFL. The first rule defines the failure logic for a guide liquid function that is addressed by a generic valve component. Depending on the values of the input control signal (CSin) and the output flow rate (Qout) of the valve, the state of the function is classified. This rule basically states that the function guide liquid will be lost if (1) there is no outflow from the valve when it is commanded on, (2) there is an outflow from the valve when it is commanded off, or (3) there is "no signal" to the valve. In all other cases, the function is considered to be operating normally. Similarly, the second rule defines the failure logic for a "transfer liquid" function implemented by the use of a "pipe" component. This function is modeled to be lost if the outflow from the pipe is zero, degraded if the pipe outflow is less than the pipe inflow, and operating for all other values of corresponding system state variables. Similar models are developed for different function component mappings in the electromechanical system domain and their use is illustrated in the next section with a case study.

Table 1 The component modes modeled for the hold-up tank example

Component	Nominal and Faulty Modes			
Valve	Nominal on	Nominal off	Failed open	Failed closed
Pipe	Nominal	Failed clogged	Failed leak	
Tank	Nominal	Failed leak		
Controller	Nominal	Failed no signal		
Sensor	Nominal	Failed burst		

4 Case Study: Hold-Up Tank Problem

In this section, the evaluation of potential functional failures of a "hold-up tank" concept is presented. The problem is to design a system that would regulate the liquid amount in an open tank. Different versions of this problem have widely been studied by various risk analysis researchers [46–49].

For the implementation of the FFIP method, a FM of the hold-up tank is generated following the function-based design process [37]. Figure 3 shows the FM, the schematic, and the CFG of this developed model. The hydraulic system consists of seven components: a tank, two valves, two pipes, a controller, and a level sensor. One of the valves, the outlet valve, is manually controlled by an operator. The inlet valve, on the other hand, is actuated through a controller based on sensor measurements from the level sensor installed on the tank. The flow rate of both valves is assumed to be the same. To simplify the analysis, it is also assumed that the liquid supply to the system is uninterrupted. Moreover, control laws are designed such that the controller shuts off the inlet valve if the liquid level reaches an overflow threshold to prevent a potentially hazardous tank overflow. Similarly, the operator is expected to shut off the outlet valve if the liquid level reduces below a hazardous dry-out threshold. As is shown on the system CFG, there are ten state variables (attached to the flows (arcs) of the CFG). The seven components of the system have a total of 13 distinct modes, 6 nominal modes, and 7 fault modes are given in Table 1.

The system is analyzed under two critical scenarios involving malfunctioning of critical system components as well as an operator error. In the first scenario, the effects of a clogged pipe and a valve failure are investigated, whereas in the second scenario, the consequences of a sensor failure and an operator error are examined. In these analyses, the goal of the FFIP framework is to provide answers to specific questions like "what functions will be lost (and when) if the inlet pipe gets clogged followed by the outlet valve failing open." The detailed results of these two FFIP analyses are presented in the next section.

5 Results

As an initial implementation, two simulation scenarios are formulated and executed. To start the simulation, the modes of the seven components and the input state variables are specified. There are two input state variables for the hold-up tank system: Q_1 (the input flow rate; set to {nominal}) and h_1 (the level of the liquid in the tank; set to {nominal}). In addition to setting input state variables, all system components are initialized to be functioning nominally.

In the first scenario, there are two events considered: the inlet pipe getting clogged, and the outlet pipe failing open. These events are injected into the simulation in the specified order. Several snapshots of the system state during the simulation are shown in Fig. 5 (at simulation Time Steps 0, 5, 8, 10, 14, 19, 20, 22, and 25). The system starts out as working nominally until Time Step 8, when the mode of the inlet pipe is set to "failed clogged." The

	at t=0	at t=5	at t=8	at t=10	at t=14	at t=19	at t=20	at t=22	at t=25
Component Modes									
Inlet Valve	nominal on	nominal on	nominal on	nominal on	nominal on	nominal on	nominal on	nominal on	nominal on
Inlet Pipe	nominal	nominal	clogged	clogged	clogged	clogged	clogged	clogged	clogged
Tank	nominal	nominal	nominal	nominal	nominal	nominal	nominal	nominal	nominal
Outlet Pipe	nominal	nominal	nominal	nominal	nominal	nominal	nominal	nominal	nominal
Outlet Valve	nominal on	nominal on	nominal on	nominal on	nominal off	nominal off	failed open	failed open	failed open
Sensor	nominal	nominal	nominal	nominal	nominal	nominal	nominal	nominal	nominal
Controller	nominal	nominal	nominal	nominal	nominal	nominal	nominal	nominal	nominal
State Variables									
Liquid Flow: Q1	nominal	nominal	nominal	nominal	nominal	nominal	nominal	nominal	nominal
Liquid Flow: Q2	?	nominal	?	nominal	?	nominal	?	nominal	nominal
Liquid Flow: Q3	?	nominal	?	zero	?	zero	?	zero	zero
Liquid Flow: Q4	?	nominal	?	nominal	?	zero	?	nominal	nominal
Liquid Flow: Q5	?	nominal	?	nominal	zero	zero	?	nominal	nominal
Liquid Flow: Q6	?	nominal	?	nominal	zero	zero	?	nominal	nominal
Control Signal Flow: CS1	?	on	?	on	?	on	?	on	on
Control Signal Flow: CS2	?	on	?	off	?	off	?	off	off
Liquid Flow: h1	nominal	nominal	nominal	hdot	hdot	hdot	hdot	hdo	hdo
Status Signal Flow: h2	?	nominal	?	hdot	?	hdot	?	hdo	hdo
System Functions									
Import Liquid		operating		operating		operating		operating	operating
Guide Liquid		operating		operating		operating		operating	operating
Transfer Liquid		operating		lost		lost		lost	lost
Store Liquid		operating		operating		operating		operating	operating
Supply Liquid		operating		degraded		degraded		degraded	lost
Transfer Liquid		operating		operating		operating		operating	operating
Guide Liquid		operating		operating		operating		lost	lost
Export Liquid		operating		operating		operating		operating	operating
Measure Level		operating		operating		operating		operating	operating
Process Signal		operating		operating		operating		operating	operating

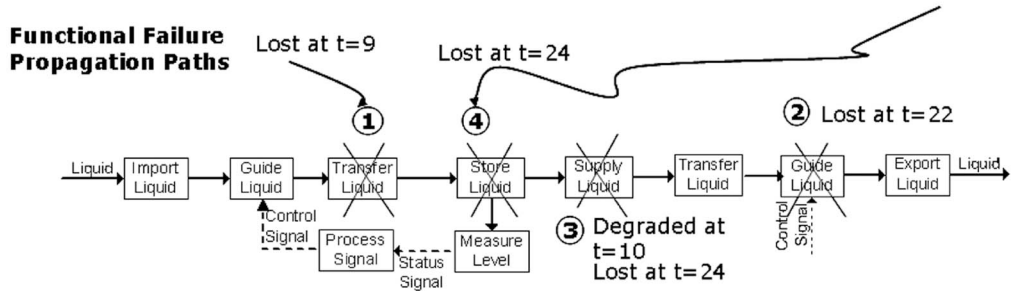

Fig. 5 The evolution of system state for the first critical scenario

system responds to this component failure through the designed control laws. Accordingly, the operator closes off the outlet valve at the 14th time step to prevent a potential dry-out. Then, at $t=20$, the outlet pipe is assumed to "fail open." This causes the tank level to drop further resulting in a tank dry-out. The estimates of the FFL reasoner during this simulation are also listed in Fig. 5. The first functional loss occurs at $t=9$ for the transfer liquid function. This is illustrated in Fig. 5 by the column $t=10$. Leading to this time step, Q_3 (i.e., the output flow of the inlet pipe) becomes "zero." Then, the corresponding FFL rule reasons about the status of this function and identifies the transfer liquid function as "lost." Following this functional loss, the "supply liquid" function is degraded due to the relatively low liquid level in the tank. After the valve failure, the guide liquid function is lost at $t=22$. Finally, the "supply" and store liquid functions are lost at $t=24$.

In the second scenario, the system initial conditions are the same as in the first simulation; however, a different set of events is considered: a sensor failure and an operator error. In this simulation, the system is fully functional until the level sensor fails at $t=7$. Leading to this time step, h_2 (i.e. the output status signal of the sensor) takes the value of "no signal." Accordingly, the corresponding FFL rule reasons about the status of this function and identifies the "measure level" function as lost. At this point, the system is working with the inlet valve under nominal on mode, and the absence of an on control signal to the valve does not have an immediate negative effect on the system. However, at $t=20$, the operator mistakenly shuts off the outlet valve. This causes the liquid level to rise, eventually requiring an off signal to be issued for the inlet valve to prevent a potential tank overflow. However, since the sensor is burst, the rise of the liquid level cannot be detected and therefore the inlet valve cannot be shut off. The liquid level continues to rise and the tank overflows. The first functional loss occurs at $t=9$ for the measure level function. Immediately following this, the "process signal" and the guide liquid functions are lost at $t=10$ and $t=11$.

6 Discussion of FFIP and Specific Contributions

The two scenario cases for the hold-up tank problem show how the FFIP framework can be used to assess and evaluate potential functional failures in a physical system. As these examples illustrate, the use of FFIP helps designers in identifying what functional failures may occur in the system if certain events take place. Moreover, the propagation paths of these failures are identified and presented to designers/analysts for further design refinement. There are several unique characteristics of the FFIP framework.

(1) Through FFL, FFIP reasons at the functional level. This goes beyond the traditional reasoning tools that use low-level sensor values to infer failed component behavior given the signs of errors in system operation. In most systems, even with automated fault detection and diagnosis, the determination of real impact of problems relies on human expertise. The assessment of the impact of failures on system performance requires modeling and reasoning methods establishing the relationship between components, their failure modes, the functionality of components, the propa-

gation of failure effects between components, and the role of functionality in mission accomplishment. From this perspective, the FFIP framework is a significant step toward automating the impact analysis.

(2) In FFIP, no a priori assumptions need to be made regarding possible fault propagation paths. Existing fault propagation analysis tools, such as TEAMS [23], SymCure [50], and the HFPG [51], require designers to explicitly formulate a fault propagation model by specifying paths of causal relationships. In contrast, FFIP only uses information available during conceptual design to determine potential failures and their propagation paths. Accordingly, only a system's functional and structural topology and their mapping are used as input for failure analysis. This topological knowledge is integrated with qualitative behavioral models of abstract, generic components and a first-order FFL that facilitates identification of failures and fault propagation paths directly from component interactions and the resulting dynamics of the system.

(3) FFIP captures various nonlinear aspects of fault propagation. It is too simplistic and often incorrect to assume that faults propagate by following the functional or structural connectivity of a system. For example, a valve getting "stuck open" should not impose any fault propagation to its neighboring pipe component. In this case, the pipe will continue to transfer its flow as intended. Similarly, as it is illustrated in the first failure scenario, the loss of the guide liquid function eventually causes the loss of supply and store liquid functions in the system. These two functions, however, are neither connected to the initiating function nor are on its path downstream in the FM. In FFIP, a proper mapping between the behavior of the system, its physical state, and the system functions enable the identification of these nontrivial, nonlinear fault propagation paths.

(4) Another important feature of FFIP is its ability to identify functional failures that do not result from direct component failures but rather from global component interactions. While component failures almost always lead to functional failures, functions can also fail without malfunctioning of their associated components. The tank in the first scenario is a good example. Throughout the simulation, the tank was in its nominal mode, without experiencing a faulty state. However, the two functions the tank has to deliver were among the functional losses that are estimated to occur. These failures resulted from events that were not relevant to the component "tank." It is important to be able to identify such functional failures that are not directly related to component failures.

(5) Finally, the FFIP framework is not built upon the unrealistic single fault assumption. Accordingly, any number of component failures can be introduced to the simulation and the framework supports the analysis of multiple failures that affect the system in parallel. Examples of this are present in both simulation scenarios.

There are also several areas where the presented FFIP framework can be improved upon. First, the use of function structures and CFGs as the underlying modeling scheme for representing a system introduces inherent limitations. For example, nonfunctional, spatial, and temporal interfaces cannot be modeled using these two graphical representations and hence are not currently accommodated by the presented framework. Second, the reasoning module FFL is dependent on the localized modeling of the different fault modes of components, and those failure modes that are not modeled cannot be reasoned about during the simulation. Capturing all possible failure modes for each component exhaustively and definitively is a common challenge for all model-based approaches to fault assessment and reasoning. Third, in the current implementation of the FFIP framework, the sequence of events to simulate is chosen by the designer. Unavoidably, a designer may miss certain sequence of events that could lead to failures. Exploring the event sequence space automatically for exhaustive coverage is an interesting area of research and is undertaken by a group of PRA researchers [43].

Overall, FFIP and its unique characteristics provide a system-level modeling approach that enables the designers to understand the interactions that may lead to functional failures and help them improve the quality of the systems at the earliest stages of the design process.

7 Conclusions and Future Work

In this paper, a new FFIP framework is presented as a formal method to evaluate and assess potential functional failures in complex engineered systems. Our goal in this research is to establish an early influence on system design so that functional failures and their propagation paths can be identified, and potential functional losses and their impact to the overall system can be determined. With the help of FFIP, the design teams can systematically explore risks and vulnerabilities without committing to design decisions too early, and can develop ISHM systems that effectively and efficiently guard against system failures. Application of the presented method to the hold-up tank example demonstrates the unique capabilities of the FFIP framework and how it can be used toward more reliable system design.

Areas identified for future work include the implementation of the proposed framework on a spacecraft reaction control system (RCS). The initial RCS system model developed for this purpose includes 37 components with 113 operational and faulty modes, 51 system functions, and 76 state variables. The major goal with using this example is to implement the developed FFIP framework in a large-scale, highly complex system design scenario. Specifically, it is planned that the effects of the difference in fidelity of component models and the difference in discretization of continuous state variables are tested by exploring various representations and the choice of functional and configurational models. The mapping from the conceptual design to the finite-state machine representation does not need to be unique or one-to-one. This study will help us understand the proper fidelity level for component modeling and the discretization of state variables. Using this well documented design example will also provide the opportunity to directly compare the performance of FFIP with other approaches to fault assessment including FMEA and FTA.

Another area for future work is aimed at investigating ways in which the developed FFIP framework can be used to quantitatively determine the safety and reliability of different conceptual design alternatives. This will allow system designers to quantify system safety beginning from very early stages of design and to explore various conceptual design alternatives guided by safety and reliability requirements. Using the FFIP framework, system designers can determine which concepts are more reliable and safer than others. Moreover, they can make decisions about what components to select, how to configure them, the types and locations of necessary safeguards, and the proper level of system redundancy, all guided by potential functional failures and their impact on overall system performance as determined by the use of the FFIP framework. Such quantification of system safety and reliability through functional failure analysis will allow the use of risk as a commodity in trade-off studies among other system-level objectives during concept development of complex engineering systems.

References

[1] Johnson, S., 2005, "Introduction to System Health Engineering and Management in Aerospace," *First Integrated Systems Health Engineering and Management Forum*, Napa, CA, Nov.

[2] Department of Defense, "Procedures for Performing Failure Mode, Effects, and Criticality Analysis," MIL-STD-1629A.

[3] Vesely, W. E., Goldberg, F. F., Roberts, N. H., and Haasi, D. F., 1981, *The*

Fault Tree Handbook, US Nuclear Regulatory Commission, NUREG 0492, Washington, DC.
[4] Greenfield, M. A., 2000, "NASA's Use of Quantitative Risk Assessment for Safety Upgrades," *IAAA Symposium*, Rio de Janeiro, Brazil.
[5] Stamatelatos, M., and Apostolakis, G., 2002, "Probabilistic Risk Assessment Procedures Guide for NASA Managers and Practitioners v1.1," NASA, Safety and Mission Assurance.
[6] Giarratano, Joseph C., and Riley, Gary D., 2004, *Expert Systems: Principles and Programming*, 4th ed., PWS, Boston MA, 2004.
[7] Shortliffe, E., 1976, *MYCIN: Computer-Based Medical Consultations*, Elsevier, New York.
[8] Touchton, R. A., 1986, "Emergency Classification: A Real Time Expert System Application," Proceedings of SouthCon.
[9] deKleer, J., and Williams, B. C., 1987, "Diagnosing Multiple Faults," Artif. Intell., **32**, pp. 97–130.
[10] Chen, J., and Patton, R. J., 1998, *Robust Model-Based Fault Diagnosis for Dynamic Systems*, Kluwer Academic, Dordrecht.
[11] Dvorak, D., and Kuipers, B. J., 1989, "Model Based Monitoring of Dynamic Systems," IJCAI.
[12] Patton, R., Frank, P., and Clark, R., 1989, *Fault Diagnosis in Dynamic Systems: Theory and Applications*, Prentice Hall, Hertfordshire, UK.
[13] Williams, B. C., and Nayak, P. P., 1996, "A Model-Based Approach to Reactive Self-Configuring Systems," *AAAI*, pp. 971–978.
[14] Kurien, J., and Nayak, P., 2000, "Back to the Future with Consistency-based Trajectory Tracking," *AAAI*, pp. 370–377.
[15] Kramer, M. A., and Palowitch, B. L., Jr., 2004, "A Rule-Based Approach to Fault Diagnosis Using the Signed Directed Graph," AIChE J., **33**(7), pp. 1067–1078.
[16] Rao., N. S. V., 1996, "On Parallel Algorithms for Single-Fault Diagnosis in Fault Propagation Graph Systems," IEEE Trans. Parallel Distrib. Syst., **7**(12), pp. 1217–1223.
[17] Chessa, S., and Santi, P., 2001, "Operative Diagnosis of Graph-Based Systems With Multiple Faults," IEEE Trans. Syst. Man Cybern., Part A. Syst. Humans, **31**(2), pp. 112–119.
[18] Tu, F., Pattipati, K. R., Deb, S., and Malepati, V. N., 2003, "Computationally Efficient Algorithms for Multiple Fault Diagnosis in Large Graph-Based Systems," IEEE Trans. Syst. Man Cybern., Part A. Syst. Humans, **33**, pp. 73–85.
[19] Stevenson, R. W., Miller, J. G., and Austin, M. E., 1991, "Failure Environment Analysis Tool (FEAT) Development Status," *AIAA Computing in Aerospace VIII Conference*, Baltimore, MD, AIAA 91-3803.
[20] Abdelwahed, S., Karsai, G., and Biswas, G., 2003, "System Diagnosis Using Hybrid Failure Propagation Graphs," Vanderbilt University, Technical Report ISIS-02-302.
[21] Sacks, I. J., 1985, "Digraph Matrix Analysis," IEEE Trans. Reliab., **R-34**(5), pp. 437–446.
[22] 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 Aerosp. Electron. Syst. Mag., **10**(5), pp. 14–25.
[23] QSI, Q.S.I., Testability Engineering and Maintenance System (TEAMS) Tool.
[24] Yairi, T., Kato, Y., and Hori, K., 2001, "Fault Detection by Mining Association Rules From House-Keeping Data," Proceedings of SAIRAS.
[25] Berenji, H., Ametha, J., and Vengerov, D., 2003, "Inductive Learning For Fault Diagnosis," *Proceedings of the 12th IEEE International Conference on Fuzzy Systems*, pp. 726–731.
[26] Mehr, A. F., and Tumer, I. Y., 2006, "Risk Based Decision Making for Managing Resources During the Design of Complex Aerospace Systems," ASME J. Mech. Des., **128**(4), pp. 1014–1022.
[27] Hoyle, C., Mehr, A. F., Tumer, I. Y., and Chen, W., 2007 "On Quantifying Cost-Benefit of ISHM in Aerospace Systems," *2007 IEEE Aerospace Conference*.
[28] Tumer, I. Y., 2005, "Towards ISHM Co-Design: Methods and Practices for Fault Avoidance and Management During Early Phase Design," *First Integrated Systems Health Engineering and Management Forum*, Napa, CA, Nov.
[29] Hutcheson, R., and Tumer, I. Y., 2005, "Function-Based Co-Design Paradigm for Robust Health Management," *The fifth International Workshop on Structural Health Monitoring*, Stanford, CA, Sep.
[30] Tumer, I. Y., and Stone, R. B., 2003, "Mapping Function to Failure During High-Risk Component Development," Res. Eng. Des., **14**, pp. 25–33.
[31] Stone, R. B., Tumer, I. Y., and VanWie, M., 2005, "The Function-Failure Design Method," ASME J. Mech. Des., **127**(3), pp. 397–407.
[32] Hirtz, J., Stone, R., McAdams, D., Szykman, S., and Wood, K., 2002, "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts," Res. Eng. Des., **13**(2), pp. 65–82.
[33] Tumer, I. Y., Stone, R., and Bell, D., 2003, "Requirements for a Failure Mode Taxonomy for Use in Conceptual Design," *Proceedings of the International Conference on Engineering Design*, ICED, Stockholm, Paper No. 1612.
[34] Roberts, R., Stone, R., and Tumer, I. Y., 2002, "Deriving Function-Failure Information for Failure-Free Rotorcraft Component Design," *Proceedings of ASME Design Engineering Technical Conference*, Montreal, Canada, DETC2002/DFM-34166.
[35] Hutcheson, R., and Tumer, I. Y., 2005, "Function-Based Design of a Spacecraft Power Subsystem Diagnostics Testbed," ASME IMECE2005–81120.
[36] Grantham Lough, K., Stone, R., and Tumer, I., 2006, "Prescribing and Implementing the Risk in Early Design (RED) Method," *Proceedings of the ASME DETC*, Philadelphia, PA.
[37] Pahl, G., and Beitz, W., 1984, Engineering Design: A Systematic Approach, Design Council, London.
[38] Hirtz, J., Stone, R., McAdams, D., Szykman, S., and Wood, K., 2002, "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts," Res. Eng. Des., **13**(2), pp. 65–82.
[39] Kurtoglu, T., Campbell, M. I., Gonzales, J., Bryant, C. R., McAdams, D. A., and Stone, R. B., 2005, "Capturing Empirically Derived Design Knowledge for Creating Conceptual Design Configurations," *Proceedings of DETC2005*, Long Beach, CA, Sep. 24–28.
[40] Kurtoglu, T., Campbell, M., Bryant, C., Stone, and R., McAdams, D., 2005, "Deriving a Component Basis for Computational Functional Synthesis," *Proceedings of ICED'05*, Melbourne, Australia.
[41] Wertz, J. R., and Larson, W. J., 1999, *Space Mission Analysis and Design*, 3rd ed., Space Technology Library, Microcosm, Kluwer Academic, Dordrecht.
[42] 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.
[43] Mosleh, A., Groen, F., Hu, Y., Zhu, D., Najad, H., and Piers, T., 2004, "Simulation-Based Probabilistic Risk Analysis Report," Center for Risk and Reliability, University of Maryland.
[44] Forbus, K., 1984, "Qualitative Process Theory," Artif. Intell., **24**, pp. 85–168.
[45] Kuipers, B. J., 1986, "Qualitative Simulation," Artif. Intell., **29/3**, pp. 289–338.
[46] Aldemir, T., 1987, "Computer-Assisted Markov Failure Modeling of Process-Control Systems," IEEE Trans. Reliab., **36**(1), pp. 133–149.
[47] Cojazzi, G., 1996, "The DYLAM Approach for the Dynamic Reliability Analysis of Systems," Reliab. Eng. Syst. Saf., **52**(3), pp. 279–296.
[48] Siu, N., 1994, "Risk Assessment For Dynamic Systems—An Overview," Reliab. Eng. Syst. Saf., **43**(1), pp. 43–73.
[49] Hu, Y., 2005, "A Guided Simulation Methodology for Dynamic Risk Assessment of Complex Systems," Dissertation, University of Maryland, College Park.
[50] Kapadia, R., 2003, "SymCure: A Model-Based Approach for Fault Management With Causal Directed Graphs," IEA/AIE 2003, LNAI 2718, pp. 582–591.
[51] Mosterman, P. J., and Biswas, G., 1999, "Diagnosis of Continuous Valued Systems in Transient Operating Regions," IEEE Trans. Syst. Man Cybern., Part A. Syst. Humans, **29**(6), pp. 545–565.