INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'07
28 - 31 AUGUST 2007, CITE DES SCIENCES ET DE L'INDUSTRIE, PARIS, FRANCE

FFIP: A FRAMEWORK FOR EARLY ASSESSMENT OF FUNCTIONAL FAILURES IN COMPLEX SYSTEMS

Tolga Kurtoglu[1], Irem Y. Tumer[2]
[1]Graduate Research Assistant[*], The University of Texas at Austin, Austin, TX, USA, 78712
[2]Associate Professor, Oregon State University, Corvallis, OR, USA, 97330

ABSTRACT
Ensuring the reliability of complex software intensive systems is becoming a critical requirement for all military and commercial aerospace applications, and becomes especially more challenging when implemented for autonomous and evolving deployments required of such applications. To ensure reliability, this research asserts that knowledge, data, and models of such complex systems must be integrated with their intended systems starting from the early design stages, hence enabling designers and engineers to plan for contingencies, redundancies, and potential changes early, before costly design decisions have been made. In this paper, a general system-level design methodology is introduced to perform simulation-based failure identification and propagation analysis of software-hardware systems. In particular, the Functional Failure Identification and Propagation (FFIP) analysis framework is introduced as a novel approach for designing reliable software-intensive systems. A combination of function, structure, and behaviour modelling is proposed to simulate failure propagation paths and the resulting functional failures to determine mitigation options, integrating hierarchical system models with behavioural simulation and qualitative reasoning. The overall goal of this research is to develop a formal framework and simulation-based design tool for design and system engineering teams to evaluate and assess the potential of functional failures of software intensive systems throughout the lifecycle.

Keywords: failure prevention and analysis, risk based design, functional modelling, conceptual design

1 INTRODUCTION
Risk analysis and management is an important requirement in the development of complex systems. Many types of risk analyses are used during the lifecycle of complex systems [1-5] including quantitative and probabilistic methods [6], reliability analysis techniques applied to design [7,8], or knowledge-based approaches such as lessons learned databases and hazard analyses [9]. One of the critical shortcomings of these methods is the difficulty in applying them during the early stages of design, where the models are vague, knowledge and decisions are difficult to capture, and probabilities are hard to assign. Studies and design reviews have pointed to the early design stages as one of the best times to catch potential failures and anomalies [10]. It is at these stages, where many decisions and tasks are still open (e.g., sensor and measurement point selection, safeguards, redundancies, diagnosis, signature and data fusion schemes) must be decided to effectively reduce the cost of risk mitigation efforts and increase the safety of designed systems.

However, we currently lack formal representations and methods for enabling risk analysis at the early design stages. Most existing risk analysis techniques require very detailed, high-fidelity models of system components in order to infer faulty system behaviour 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 facilitate early identification of risks, the focus should be kept on a system's functional models, and hence it is crucial to be able to

[*] The research was conducted at MCT/NASA Ames Research Center, Moffett Field, CA, 94035, USA.

reason at the functional level and identify what functions are likely to fail and what the overall effect of loss of these functions will be on a system's behaviour and performance.

In this paper, we introduce the functional failure identification and propagation (FFIP) framework that enables these capabilities. FFIP assists 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 can be lost, what the impact to the overall system will be. The main advantage of the framework 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 reduce risk through exploration of system components and their functionality, and facilitates the development of more reliable system configurations.

2 MOTIVATING EXAMPLE

Consider the Reaction Control System (RCS) example shown in Figure 1. A Reaction Control System is one of the major subsystems of spacecraft propulsion. It provides attitude control during orbiting through the use of a set of thrusters [11]. The RCS is composed of three subsystems – the forward, left, and right RCS located in the nose and aft sections of the shuttle. Figure 1 details the simplified layout of a single RCS subsystem. A subsystem is composed of two pressurization and propellant feed systems – one for fuel (MMH), and one for oxidizer (NTO).

The two systems are independent and connect at the thruster chambers at the end of the manifold. Each pressurization and propellant feed system contains temperature and pressure sensors in the helium and propellant tanks and pressure sensors in the manifolds. Pressure in the helium tank passes through a series of regulators that reduce the pressure to the desired working pressure. The propellant tanks are pressurized with helium which in turn expels the propellant into the tank lines, towards the manifold. Finally, the fuel and the oxidizer from the pair of feed systems are pushed into the thruster chamber where they ignite on contact and produce a hot gas and thrust.

Figure 1 also shows the functional model of the RCS subsystem describing the basic functionality of the system. It is at this high, functional level that we target to reason about failures, and their propagation. Specifically, FFIP helps answer questions regarding critical what-if scenarios such as "What happens if the helium line leaks?", and more importantly "What are the effects of this failure on overall system functionality?" In most complex systems, such as the RCS, such an assessment relies heavily on human expertise. A human expert faced with this scenario is likely to go through the following reasoning steps: "…the leak will stop the helium line from delivering its intended function, i.e., transferring gas from the helium tank to the MMH tank causing the loss of the pipe's "transfer gas" functionality (labelled as failure start point on the functional model of the RCS, point (1), in Figure 1.b) This will force the check valve to close and the propellant to be trapped in the tank. As a result, the propellant in the MMH tank will be depleted depending on the remaining helium pressure in the tank, and if the helium pressure drops under the minimum level required for forcing the propellant into the tank line, the "supply liquid" functionality of the MMH tank will seize to continue (labelled as point (2), in Figure 1.b.) The loss of this function will propagate to the thruster chamber next, in which the two propellants are designed to come in contact, explode, and produce thrust. Since, the supply from one of the propellant lines is interrupted, the "mix liquid", "convert chemical energy to thermal energy", "convert thermal energy to pneumatic energy", and "convert pneumatic energy to mechanical energy" (labelled as points (3),(4),(5), and (6) in Figure 1.b.) functions will be lost, eventually preventing the main functionality of the entire RCS subsystem, i.e. to "produce thrust". This is a critical failure that the system should be guarded against.

In this research, we automate the kind of failure reasoning demonstrated in the abovementioned scenario. Accordingly, we aim to automatically compute functional failures, fault propagation paths, and failure consequences using only the knowledge available during the early stages of design. Our method brings a much needed formalism to fault assessment and is significantly different then approaches that rely solely on expert elicitation to assess failure and its consequences.

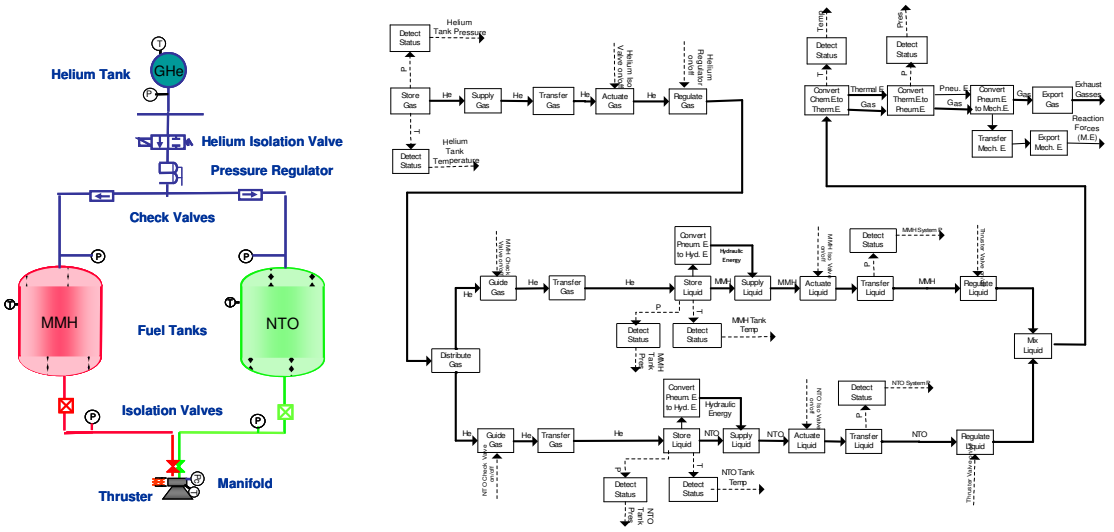

Figure 1.a. The functional model and the schematic of a simplified Reaction Control System.

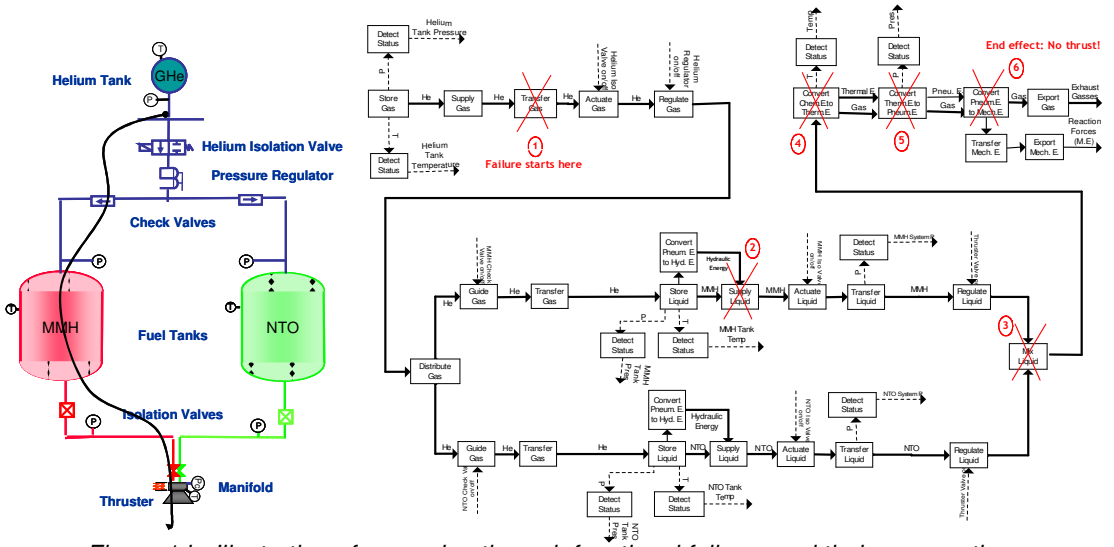

Figure 1.b. Illustration of reasoning through functional failures and their propagation.

3 RELATED WORK

Various failure assessment, reasoning, and reliability modelling methods have been in existence for some time, and are reviewed here briefly for completeness.

3.1 Failure and Risk Assessment Methods

The industries dealing with complex systems currently employ three major reliability tools and methods: FMEA, FTA, and PRA. Failure Modes and Effects Analysis (FMEA) [7] is a method that systematically examines individual system components and their failure mode characteristics to assess risk and reliability. The 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 evaluated separately to determine what effect they have at the component level, and then at the system level. It is a widely used method that is easy to understand and implement. However, the analysis requires a detailed level of system design, and thus is not optimal to be used during conceptual design [12]. Moreover, FMEA does not capture component interactions explicitly, and relies heavily on expert knowledge to assess failure consequences and their criticality [12]. As a result, it is often considered to be a highly subjective method.

Fault Tree Analysis (FTA) [8] is performed to capture event paths from failure root causes to top-level consequences. Using this approach, possible event paths from failure root causes to top-level consequences can be captured. Like FMEA, FTA is a well-accepted, standard technique. When conducted properly, it is likely to identify more possible failure causes than single-component oriented FMEA. However, FTA also relies greatly on expert input and shares similar criticism that FMEA is subject to [12]. Moreover, since the failure domain is represented using events in FTA, low-level component interactions and dynamics leading to failure are only considered informally, during expert identification of event-consequence relationships. Formally capturing component interactions and system dynamics is however crucial for supporting design decisions during early concept development.

Probabilistic Risk Assessment (PRA) [6] 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? [13] PRA combines a number of fault/event modelling 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. Recently, PRA has been extended to include event/behaviour simulation into the analysis as demonstrated by the SIMPRA tool developed by Mosleh et al. [14]. SIMPRA uses Simulink as its simulation environment, and thus demands a fully-specified system model as part of the analysis. Such detailed, high-fidelity models, however are not available during conceptual design.

3.2 Failure Reasoning Methods

A variety of diagnostic reasoning tools have been proposed for fault assessment and diagnosis. Diagnostic reasoning approaches share a common process in which a system is monitored and a comparison is performed of observed and expected behaviour of the system to detect anomalous conditions. The major approaches to fault modelling and diagnostic reasoning include expert systems, simulation and model-based diagnosis methods, and data classification techniques. Expert systems [15] are extensively used in diagnosis, where knowledge acquired from human experts is formulated in different ways such as “if-then” rules or decision trees [16, 17].

Model-based-based approaches to diagnosis, on the other hand, rely mostly on qualitative knowledge to predict the behaviour of a system [18]. When observations disagree with the predicted behaviour, some diagnostic technique is initiated to identify the faults. The broadest category for diagnostic reasoning is model-based diagnosis (MBD) [19, 20, 21, 22]. Besides the model-based approaches, some researchers have employed fault propagation graphs as the system model for diagnostic reasoning [23, 24, 25, 26, 27]. Among those, directed graphs [28] are one of the techniques used to analyze component dependencies and fault propagation [29, 30, 27]. Multi-Signal Flow Graphs developed by Deb et al. [31] is another comprehensive methodology to model cause-effect dependencies of complex systems.

Finally, in cases where physical cause-effect relationships are difficult to model in analytical form, statistical and probabilistic classification methods are applied [32, 33]. In summary, while fault propagation analysis tools exist, they require designers to explicitly formulate a fault propagation model by specifying paths of causal relationships, which is not feasible during the early stages of design where information about the system specifics is scarce.

3.3 Methods for Failure Analysis during Early Functional Design Stage

Prior work has addressed the need for integrating early risk assessment and management tools and methodologies into the vehicle and system design [34-37]. Most notably, the Function Failure Design Method (FFDM), promotes early identification of potential failures by linking them to product functions [38,39]. 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 [40] and the other failure modes [41] 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. This is achieved using matrix representations that map functions to components, and components to failure modes. The product of these two

matrices results in a third matrix that relates a system's functions to failures. Using this output, designers can analyze potential functional failures before any component selection is made. FFDM is successfully applied to the Bell 206 rotorcraft and has lately been extended to include spacecraft systems [42,43].

As an extension to FFDM, Grantham-Lough et al. [44] 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.

The FFIP method described in this paper is complementary to these approaches in that it is function oriented and thus has a conceptual design focus. However, FFIP comes with a number of additional features. First, it integrates qualitative reasoning with behavioural simulation that enables the computation of component interactions that are likely to result in functional failures. Second, it allows the identification of not only the functional failures but also their propagation paths that are derived from the functional and structural topology of a system. Third, the approach is applicable to a variety of systems and it is not constrained by a database of documented, historical failure data.

4 THE FFIP FRAMEWORK

The Functional Failure Identification and Propagation (FFIP) analysis framework is introduced as a novel approach for designing reliable complex systems. A combination of function, structure, and behaviour modelling is proposed to simulate failure propagation paths and the resulting functional failures to determine mitigation options, integrating hierarchical system models with behavioural simulation and qualitative reasoning, discussed next [12].

4.1 The FFIP Architecture

There are three major modules in the FFIP analysis framework: the graphical system model, the behavioural simulation, and the functional-failure logic (FFL) reasoner, as shown in Figure 2. The framework represents system function, configuration, and behaviour by an interrelated array of graph-based, elemental component models. The graph-based modelling approach provides a coherent, consistent, and formal schema to capture function-configuration-behaviour architecture of a system at an abstract level and facilitates the assessment of potential functional failures and resulting fault propagation paths through the FFL reasoner that translates the dynamics of the system into functional failure identifiers [12,45].

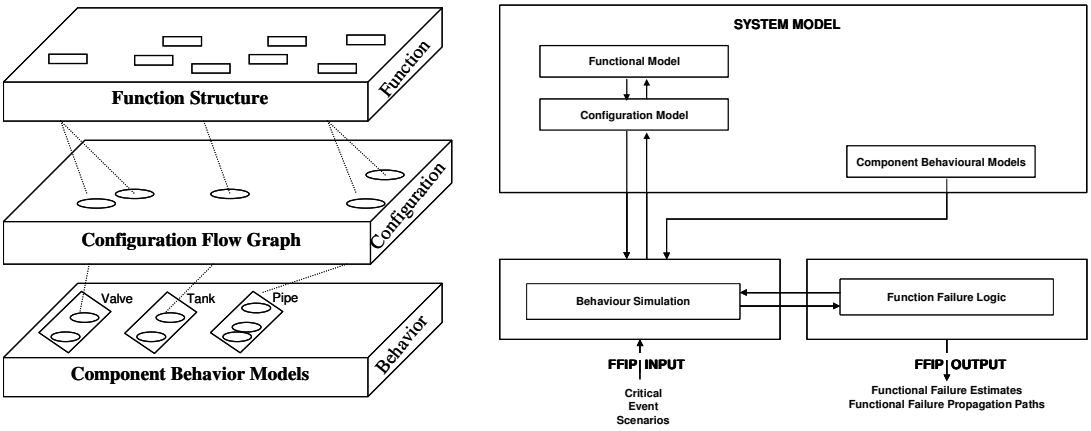

Figure 2. The architecture of the FFIP framework

4.1.1 Models of Systems

At the conceptual stage of design, 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 behaviour. In order to integrate failure propagation analysis capability at

this stage, a modelling paradigm that is capable of representing the desired functionality of the individual components, their structure, as well as their interactions, is required. Capturing these component interactions is especially critical for supporting decision-making during concept development. For example, in the hardware domain, over-pressurization of a liquid tank as a result of a certain interaction may guide a designer to choose between adding a pressure release valve and increasing the strength of the tank without changing the topology of the design. To achieve this, the knowledge of functionality, configuration, and behaviour should be integrated in such a way that it will enable the computation of component interactions that are likely to result in functional failures leading to the identification of system faults and their propagation paths.

To achieve these goals, FFIP represents system *function* using function structures [46]. Recall that 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 sub-functions. Using the terms defined in the Functional Basis [40], designers can generate a model for the actual or desired functionality of a system.

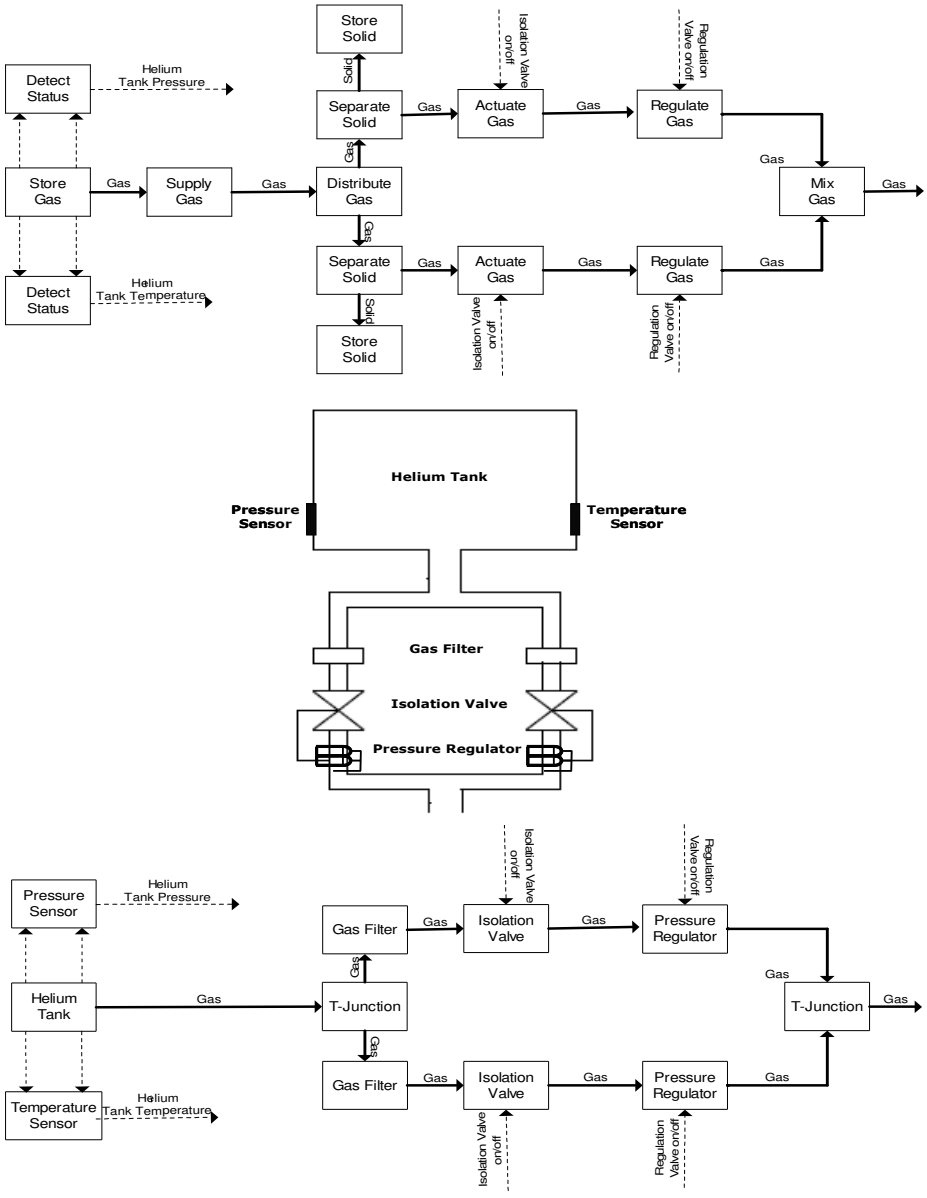

Figure 3: Functional model, schematic, and CFG for a fuel tank stage of the RCS system

The *structure* in FFIP, on the other hand, is captured using configuration flow graphs (CFG's) [45]. 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 [47].

As an example, Figure 3 shows a configuration flow graph, and a simplified system schematic of a partial Reaction Control System (RCS), introduced earlier. The figure details the layout of the helium feed stage of an RCS. This subsystem is composed of two parallel feed lines that are used to pressurize separate fuel and oxidizer tanks located downstream (not shown in figure). Figure 3 also shows the functional model of the helium feed stage describing the basic functionality of the subsystem. For this example, the component "gas filter" addresses functions "separate solid", and "store solid". Similarly, the component "tank" provides "store gas" and "supply gas" 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, FFIP represents *system behaviour* using a component oriented modeling approach. The approach involves the 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 configuration flow graphs. 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. Figure 4 shows the behavioral model of a generic "pipe" component. In this case, a pipe can transition from a "nominal" mode to a "failed clogged" or "failed leak" mode as a result of a fault event. The dynamic behaviour of the component in each mode is governed by a different set of physical laws and mathematical relations, and is therefore defined separately.

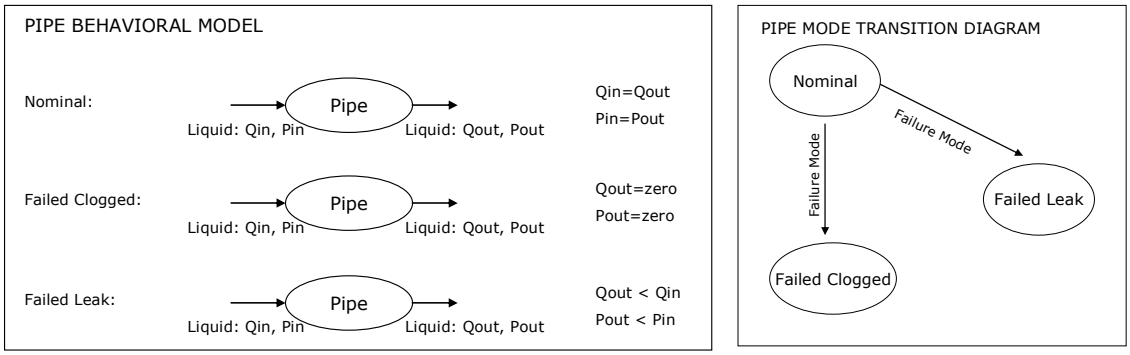

Figure 4. The behavioral model for a generic "pipe" component.

4.1.2 Simulation of System Behaviour

The next step is to determine the system behavior under certain conditions, 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. The overall system state $X(t)$ at time t is described by:

$$X(t) = \Phi(c(t), v(t)) \quad (1)$$

where,

$c(t) = [c_1, c_2, c_3, ..., c_N]$ is a vector of discrete component modes where each component $c = 1, ..., N$ (N : number of components in the system) assumes a discrete mode from its own set of M modes $c_i = (c_{i1}, c_{i2}, c_{i3}, ..., c_{iM})$, and, $v(t) = [v_1, v_2, v_3, ..., v_K]$ 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 $v(t)$ then defines these qualitative values for each state variable v_i from a set of *P* possible values $v_i = (v_{i1}, v_{i2}, v_{i3}, ..., v_{iP})$. For example, a liquid flow rate variable may take on values from the set of {zero, low, nominal, high}. Similarly, a control signal variable may have values of {nosignal, on, off}, etc.

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

Note that, FFIP is not mainly concerned with simulating the dynamic behaviour 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 behavioural dynamics and the function of components, explained next.

4.1.3 Reasoning about Failures using the Function-Failure Logic

The final step is to reason about potential failures with the aid of a Function-Failure Logic (FFL) reasoner. FFL is used 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 $X(t) = \Phi(c(t), v(t))$ (Equation 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.

As an example, Figure 5 illustrates three possible rules that come from the FFL reasoner. 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 modelled to be lost if the outflow from the pipe is zero, degraded if the pipe outflow is less then 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.

Note that, 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 functional model (function) and the configuration flow graph (behaviour) is fundamental to the employment of the function failure logic.

5 APPLICATION TO THE REACTION CONTROL SYSTEM

Let us return to the motivating example presented in Section 2, where the conceptual application of FFIP is discussed. The major goal with using this example is to implement the framework in a large-scale, highly complex system design scenario consisting of hardware and software components. The initial Reaction Control System (RCS) model developed to assess the performance of the FFIP method

includes 37 components with 113 operational and faulty modes, 51 system functions, and 76 state variables (See Figure 1). Specifically, the effects of the difference in fidelity of component models and the difference in discretization of continuous state variables must be tested by exploring various representations and choice of functional and configurational models. Moreover, the mapping from the conceptual design to the finite state machine representation needs to be studied in order to determine the proper level of fidelity for component modelling and the discretization of state variables. The full scale implementation of the FFIP method on the RCS model is planned to be developed with our collaborators at the Air Force Research Laboratory and NASA. 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.

A smaller scale implementation of the FFIP method was tested on a "hold-up tank" design example by Kurtoglu and Tumer [12]. This simpler problem consisted of designing a system that would regulate the liquid amount in an open tank. 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. A software module controls the laws that are designed to shut off the inlet valve if the liquid level reaches an overflow threshold. Similarly, the operator is expected to shut off the outlet valve if the liquid level reduces below a hazardous dry-out threshold.

For hold-up tank example, the functional and the configurational model of this system, as well as the behavioural models were developed for the aforementioned seven components. Furthermore, the FFIP method was tested on the hold-up tank example using two simulation scenarios. In the first scenario, two events were considered: the inlet pipe getting clogged, and the outlet pipe failing open. In the second scenario, a different set of critical events was considered: a sensor failure and an operator error. In both scenarios, FFIP estimated potential functional failures and their propagation paths by using behavioural simulation and failure-based qualitative reasoning. The results of the hold-up tank study are presented in detail in [12] and show how the FFIP method and its unique characteristics provide an effective system level modelling and analysis approach. Specifically, this example illustrates how FFIP enables 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.

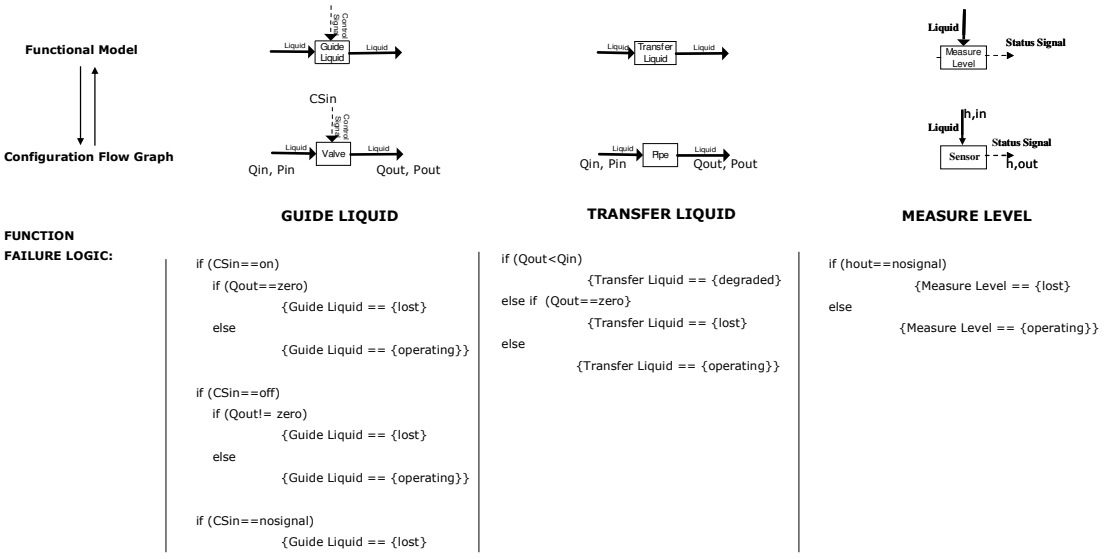

Figure 5. Function-failure logic for guide liquid, transfer liquid, and measure level functions.

5 CONCLUSIONS

In this paper, the Functional Failure Identification and Propagation (FFIP) analysis framework is introduced as a novel approach to enable the analysis of functional failures and their propagation paths during the early stages of design for complex systems. The overall goal of this research is to develop a *formal framework* and *simulation-based design tool* for design and system engineering teams to evaluate and assess the potential of *functional failures* of *software intensive systems* throughout the lifecycle.

This paper focused on the main principles underlying the FFIP analysis framework. This framework uses a combination of *functional, structural, and behavioural models* as a means to represent early system design trades, and *high-level behavioural simulation* to estimate potential faults and their propagation under failure scenarios. In addition, a fundamental relationship is established between such failures and the system level behaviour and response of complex engineered systems. Finally, a *function-failure reasoner* is developed that can be used generically by system designers to reason about potential failures and their consequences. A conceptual application of the approach was discussed using the Reaction Control System (RCS) example. Ongoing work is exploring the automation of the FFIP as a usable methodology and testing using the complex RCS system.

The FFIP framework presents several unique opportunities. First, the function-failure-logic (FFL) employed by the reasoner is a novel approach in that it allows to reason about the system state at a very high, functional level. 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 function-failure-logic with a behavioural simulation enables automatic computation of functional failures and fault propagation paths directly from physical behaviour and component interactions of the system. This brings a much needed formalism to fault assessment and is significantly different than approaches that rely solely on expert input to assess failures. Third, the developed framework is able to identify functional failures that do not result from direct component failures, but rather from global component interactions. Finally, FFIP captures various non-linear aspects of fault propagation, most notably the deviation of fault propagation paths from strict structural and functional connectivity.

REFERENCES

[1] Zang, T. A., Hemsch, M. J., Hilburger, M. W., Kenny, S. P., Luckring, J. M., Maghami, P., Padulam, S. L., and Stroud, W. J., 2002, "Needs and Opportunities for Risk-Based Multidisciplinary Design Technologies for Vehicles," Technical memorandum NASA/TM-2002-211462, NASA Langley Research Center, Hampton, VA.

[2] Backman, B., 2000, "Design, Innovation and Risk Management: A Structural Designer's Voyage into Uncertainty," in *ICASE Series on Risk-based Design*.

[3] Choi, K., 2001, "Advances in Reliability-Based Design Optimization and Probability Analysis - PART II," in *ICASE Series on Risk-based Design*.

[4] Smith, N. and Mahadevan, S., 2003, "Probabilistic Methods for Aerospace System Conceptual Design," *Journal of Spacecraft and Rockets*, 40(3), pp. 411-418.

[5] V. Venkatasubramanian, Jinsong Zhao and Shankar Viswanathan. "Intelligent Systems for HAZOP Analysis of Complex Process Plants," *Computers and Chemical Engineering*, 24 (2000), 2291-2302.

[6] Greenfield, M.A. "NASA's Use of Quantitative Risk Assessment for Safety Upgrades". In IAAA Symposium. 2000. Rio de Janeiro, Brazil.

[7] Department of Defense, "Procedures for performing failure mode, effects, and criticality analysis." MIL-STD-1629A.

[8] Vesely, W. E., Goldberg, F. F., Roberts, N. H. and Haasi, D. F., The Fault Tree Handbook, US Nuclear Regulatory Commission, NUREG 0492, 1981.

[9] NASA Exploration Systems Mission Directorate. *Methodology for Conduct of Project Constellation Hazard Analyses*. (Document 0000028585, 2005).

[10] Mahadevan, S. and Smith, L., 2003, "System Risk Assessment and Allocation in Conceptual Design," Contractor report. NASA/CR-2003-212162, NASA.

[11] Wertz, J.R. and Larson, W. J. Space Mission Analysis and Design. 3rd Ed. Space Technology Library. Microcosm Press and Kluwer Academic Publishers. 1999.
[12] T. Kurtoglu and I. Y. Tumer, *A graph based fault identification and propagation framework for functional design of complex systems*, ASME Journal of Mechanical Design. Accepted for publication (May 2007).
[13] Stamatelatos, M. and Apostolakis, G. "Probabilistic Risk Assessment Procedures Guide for NASA Managers and Practitioners v1.1". 2002, NASA, Safety and Mission Assurance.
[14] A. Mosleh, F. Groen, Y. Hu, D. Zhu, H. Najad and T. Piers, "Simulation-based probabilistic risk analysis report," Center for Risk and Reliability, University of Maryland, 2004.
[15] J. C. Giarratano and G. D. Riley, *Expert systems: Principles and programming*, PWS Publishing Company, Boston, MA, 2004.
[16] E. Shortliffe, Mycin: Computer-based medical consultations, Elsevier, New York, 1976.
[17] R. A. Touchton, Emergency classification: A real time expert system application, SouthCon, 1986.
[18] J. deKleer and B. C. Williams, *Diagnosing multiple faults*, AI 32 (1987), 97-130.
[19] J. Chen and R. J. Patton, *Robust model-based fault diagnosis for dynamic systems*, Kluwer Academic Publishers, 1998.
[20] D. Dvorak and B. J. Kuipers, *Model based monitoring of dynamic systems*, IJCAI, 1989.
[21] B. C. Williams and P. P. Nayak, A model-based approach to reactive self-configuring systems, AAAI, 1996, pp. 971-978.
[22] J. Kurien and P. Nayak, Back to the future with consistency-based trajectory tracking, AAAI, 2000, pp. 370-377.
[23] M. A. Kramer and B. L. Palowitch, Jr, A rule-based approach to fault diagnosis using the signed directed graph, AIChE Journal 33, no. 7, 1067-1078.
[24] N. S. V. Rao, On parallel algorithms for single-fault diagnosis in fault propagation graph systems, IEEE Transactions on Parallel and Distributed Systems 7 (1996), no. 12, 1217-1223.
[25] S. Chessa and P. Santi, *Operative diagnosis of graph-based systems with multiple faults*, IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans 31 (2001), no. 2.
[26] R. W. Stevenson, J. G. Miller and M. E. Austin, *Failure environment analysis tool (feat) development status*, AIAA Computing in Aerospace VIII Conference, 1991.
[27] S. Abdelwahed, G. Karsai and G. Biswas, "System diagnosis using hybrid failure propagation graphs," Vanderbilt University, 2003.
[28] I. J. Sacks, *Digraph matrix analysis*, IEEE Transactions on Reliability R-34 (1985), no. 5, 437-446.
[29] QSI, "Testability engineering and maintenance system (teams) tool."
[30] R. Kapadia, *Symcure: A model-based approach for fault management with causal directed graphs*, IEA/AIE 2003, 2003, pp. 582-591.
[31] S.Deb, K. R. Pattipati, V. Raghavan, M. Shakeri and R. Shrestha, "Multisignal flow graphs: A novel approach for system testability analysis and fault diagnosis," *IEEE Aerospace and Electronics Systems Magazine*, vol. 10, 1995, pp. 14-25.
[32] T. Yairi, Y. Kato and K. Hori, *Fault detection by mining association rules from house-keeping data*, SAIRAS, 2001.
[33] H. Berenji, J. Ametha and D. Vengerov, *Inductive learning for fault diagnosis*, 12th IEEE International Conference on Fuzzy Systems, 2003, pp. 726-731.
[34] Mehr, A.F. and I. Y. Tumer, "Risk based decision making for managing resources during the design of complex aerospace systems." ASME Journal of Mechanical Design. Vol. 128, No. 4, pp. 1014-1022. July 2006.
[35] Hoyle, C., Mehr, A. F., Tumer, I. Y., Chen, W. "On quantifying cost-benefit of ISHM in Aerospace systems." 2007 IEEE Aerospace Conference.
[36] Tumer, I. Y., "Towards ISHM Co-Design: Methods and practices for fault avoidance and management during early phase design". 1st Integrated Systems Health Engineering and Management Forum (to be published as a book chapter). Napa, CA. November 2005.
[37] Hutcheson, R. and Tumer, I. Y., "Function-based Co-design Paradigm for Robust Health Management." The 5th International Workshop on Structural Health Monitoring. Stanford, CA. September 2005.
[38] Tumer, I.Y. and R.B. Stone, "Mapping Function to Failure During High-Risk Component

Development". Research in Engineering Design, 2003. 14: p. 25-33.
[39] Stone, R.B., Tumer, I.Y. and VanWie, M. "The function-failure design method." Journal of Mechanical Design, 2005. 127(3): p. 397-407.
[40] Hirtz, J., Stone, R., McAdams, D., Szykman, S., and Wood, K., "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts," Research in Engineering Design 13(2): 65-82, 2002.
[41] Tumer, I.Y., Stone, R., and Bell, D., "Requirements for a Failure Mode Taxonomy for use in Conceptual Design," Proceedings of the International Conference on Engineering Design, ICED, Stockholm, 2003, paper 1612.
[42] Roberts, R., Stone, R., and Tumer, I.Y., "Deriving Function-Failure Information for Failure-Free Rotorcraft Component Design," Proceedings of ASME Design Engineering Technical Conference, DETC2002/DFM-34166, Montreal, Canada, 2002.
[43] R. Hutcheson and I. Y. Tumer, "Function-based design of a spacecraft power subsystem diagnostics testbed". ASME IMECE2005-81120. Orlando, FL. 2005.
[44] Grantham Lough, K., Stone, R., and Tumer, I., "Prescribing and Implementing the Risk in Early Design (RED) Method", Proceedings of the ASME DETC, Philadelphia, PA, 2006.
[45] Kurtoglu, T., Campbell, M.I.,Gonzales, J., Bryant, C.R., McAdams, D.A., Stone, R.B., ,2005, "Capturing Empirically Derived Design Knowledge for Creating Conceptual Design Configurations," Proceedings of DETC2005, Sept. 24-28, Long Beach, California.
[46] Pahl, G. and Beitz, W., Engineering Design: A Systematic Approach, Design Council, London,1984.
[47] Kurtoglu, T., Campbell, M., Bryant,C, Stone, R., McAdams, D., 2005, "Deriving a Component Basis for Computational Functional Synthesis" Proceedings of ICED'05, Melbourne, Australia.

Contact: Irem Y. Tumer
Department of Mechanical Engineering – Oregon State University
204 Rogers Hall, Corvallis, OR 97331-6001, USA
Phone: +1-541-737 6627
e-mail: irem.tumer@oregonstate.edu