Transforming Functional Requirements and Risk Information into Models for Analysis and Simulation

Jane T. Malin
NASA Johnson Space Center
2101 NASA Road 1
Houston, TX 77058
281 483-2046
jane.t.malin@nasa.gov

David R. Throop
The Boeing Company
2100 Space Park Drive
Houston, TX 77058
281 460-8415
david.r.throop@boeing.com

Land Fleming
Hernandez Engineering
17625 El Camino Real
Houston, TX 77058
281 483-2055
land.d.fleming1@jsc.nasa.gov

Luis Flores
Lockheed Martin Space Operations
2400 NASA Road 1
Houston, TX 77058
281 333-6423
luis.flores@lmco.com

Abstract—[1,2]A method is presented for transforming functional requirements into system-of-subsystems function models. Text from requirements and risk tools is transformed into simple semantic models. An ontology of functions, entities and problems provides structure for the transformation and for deriving functional models. Functions, operands of functions and functional agents can be captured. Generic hazards and vulnerabilities can be identified for types of functions and operands. These models are transformed into functional architectures of connected subsystems. This approach permits application of graph analysis and lightweight simulation to investigate the effects of problems and countermeasures in scenarios. We discuss a hazard identification tool and hybrid simulation tool where these methods are being applied.

TABLE OF CONTENTS

1. INTRODUCTION

A task of conceptual design is to perform preliminary analysis of the impacts of potential problems and risks on system performance and operations, mission goals and safety. Another task is to evaluate countermeasures and influences on countermeasure effectiveness. It is a challenge to accomplish these analyses early in design, with available functional models. It is also challenging to produce models that can be transformed into ones that are appropriate for later stage analyses. We are exploring the development of multi-use functional models that can be transformed into functional architecture models for analysis and then component-connection models for simulation. We present a method for transforming functional requirements into function models. We have developed an ontology of functions, entities and problems that provides structure for functional decomposition nodes and functional models [1]. The nodes of the functional decomposition tree are imperative sentences that are transformed into simple semantic models. Functions, operands of functions and functional agents can be captured. Generic hazards and vulnerabilities can be identified for types of functions and operands. A method is presented for transforming these models into functional architectures of connected subsystems. The functional models can support analysis of both problem impacts and resource use. This supports early analyses to predict the impacts of failures and problems in system designs.

In this paper we first briefly review the model-based hazard analysis work that is the context for developing methods for transforming requirements into models. We next present an overview of methods and tools for transforming requirements and risk text into structured data objects, and methods for deriving system-of-subsystems models. We discuss the use of a parser and an ontology to achieve these transformations. Finally, we discuss an example case, where text from functional requirements, risks and countermeasures is transformed and included in models. The Models are then used to explore possible problem impacts by analysis and simulation, and this leads to redesign.

2. MODEL-BASED HAZARD ANALYSIS

We have been developing technology to aid early identification of system problems that impact performance, operations or safety [1]. We have automated analysis of transmission paths for potential to propagate system threats in operations, due to interactions among subsystems and with resources (power, thermal, data). We have developed an approach for automated analysis of effectiveness of counteractions and mitigations, including redundant systems, software and human operations.

The prototype Hazard Identification Tool (HIT) helps engineers capture and integrate design information during

[1] 0-7803-8870-4/05/$20.00© 2005 IEEE
[2] IEEEAC paper #1530, Version 3, updated December 20, 2004

conceptual design, top-down from the system level, with emphasis on system functions and structure. Information about functions, performance and risk can be used in graph analysis and simulation to investigate hazard paths and sequences. Analyses of operations scenarios can support impact analysis for preliminary hazard analysis and hazard and operability analysis (HAZOP) [2].

HIT provides vocabularies and library-based help for identifying and analyzing system functions, threats and counteractions. Usign knowledge acquisition forms, users can select types of components and entities, functions and problems from libraries. The libraries help engineers consider potential problems and counteractions that are associated with types of functions and entities. Component-connection models of systems can be developed, using libraries of models of generic components or subsystems. These subsystem models can then be mapped to behavioral models in the simulation tool.

HIT uses graph analysis of system models to help engineers identify and script candidate accident scenarios for simulation. By mapping HIT data to models in the simulation tool, it is possible to simulate event sequences during operations. The CONFIG hybrid modeling and simulation tool includes a library of generic component behavior models that is synchronized with the HIT component library. These lightweight abstract behavioral models include a broad variety of types of performance problems and hazards [3,4].

For the libraries used in this project, we developed ontologies to help classify and describe attributes associated with classes of function verbs, entities, hazards, vulnerabilities and counteractions. We used text parsing and matching results from the Reconciler tool [5] to refine these ontologies and map them to the space domain. We parsed imperative sentences and classified terms from the International Space Station (ISS) Reliability Block Diagrams (RBDs) and from the ISS Flight and System data books. We realized that we could reapply this method to analyze text from requirements tools and risk analysis tools. We could map information from the sentences into the classes in the ontology. Then we could use these data objects from the other tools to derive component-connection models and populate them with risk and counteraction data.

3. Models From Functional Requirements

We present a systematic method for transforming functional requirements into function models and risk data into problem and countermeasure data in HIT model subsystems or components.

1. Acquire text from requirements tools and risk tools.

2. Reconcile selected requirements text concerning subsystems and their shift (send/receive/transfer) or serve (provide-to/require-from) functions with function and entity classes and attributes in the HIT ontology.

3. Derive HIT component-connection model for system-of-subsystems from the requirements objects.

4. Reconcile selected risk and mitigation text for the subsystems or components in the HIT model with problem and countermeasure function classes and attributes in the HIT ontology.

5. Enhance default hazard, vulnerability and countermeasure data for HIT subsystems or components by adding data from risk objects.

The nodes of functional decompositions such as the ISS RBDs and text from requirements specification tools such as SpecTRM [6] are imperative sentences (if the factual statement forms were amended to include "shall"). Examples include:

- "Power the radar."
- "Condition the cabin atmosphere."
- "The CDHC shall receive a compressed picture file from the camera."
- "The CDHC shall send the Telecommunication Subsystem telemetry."

The Reconciler tool is used to parse the sentences and identify the function or service verb (e.g., send), the operand of the verb (e.g., telemetry) and the destination/user (e.g., Telecommunication Subsystem). Operands of functions (usually the grammatical objects) and their affected attributes are as important to capture as the services (the grammatical verbs). The functional agent or server that carries out the imperative can be identified if mentioned (e.g., CDHC). The agent, if unspecified, can be given a noun form of the verb (e.g., Power; Conditioner).

The resulting simple semantic models fill in available attributes in classes in the ontology and can be used to create a system model made up of instances of the identified subsystems, transmitted entities and transmission paths or connections. For systems where operands of functions move among subsystems (e.g. assembly lines, fluid flow, data transmission), these models can be transformed into functional architectures. Figure 1 shows requirements text, its transformation into a Requirements (function) model instance, and the derived functional architecture.

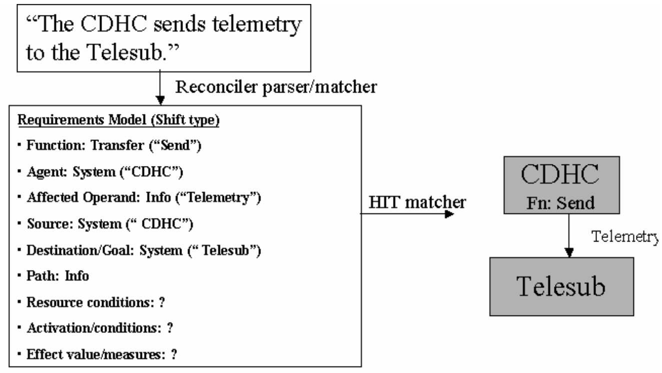

"The CDHC sends telemetry to the Telesub."
Reconciler parser/matcher
Requirements Model (Shift type)
• Function: Transfer ("Send")
• Agent: System ("CDHC")
• Affected Operand: Info ("Telemetry")
• Source: System (" CDHC")
• Destination/Goal: System (" Telesub")
• Path: Info
• Resource conditions: ?
• Activation/conditions: ?
• Effect value/measures: ?
HIT matcher
CDHC
Fn: Send
Telemetry
Telesub

Figure 1 - Transforming requirement to element of model

The types and attributes in the HIT ontology of functions, entities and problems provide the structure for deriving functional models. For example, a Shift type of function provides a structure for matching to agent, operand, source, destination and path type, as well as other attributes. The Shift type of function is a subtype of Place, and its sibling types under Place are Hold and Arrange. Shift subtypes include Send/Release, Receive, Shift-in-place and Transfer.

The same parsing and matching strategy can be used with text from risk tools such as DDP [7]. This has included use of the hierarchical outline structure of the risk and mitigation data to infer application of risk words up and down the risk hierarchy. The Reconciler also extracts and provides the link data between objectives, risks and mitigations. The imported risk data can be used to enhance default hazards and vulnerabilities from the libraries of types of functions and operands. Figure 2 shows risk and mitigation text and its transformation into problem and countermeasure structures for use in the model. The problem structure in Figure 2 uses a Function Problem type from the problem ontology. The counteraction model is based on a Replace type of function, which is a subtype of Recover, which is a subtype of Counteract/preserve. The sibling types of Replace under Recover are Restore and Undo.

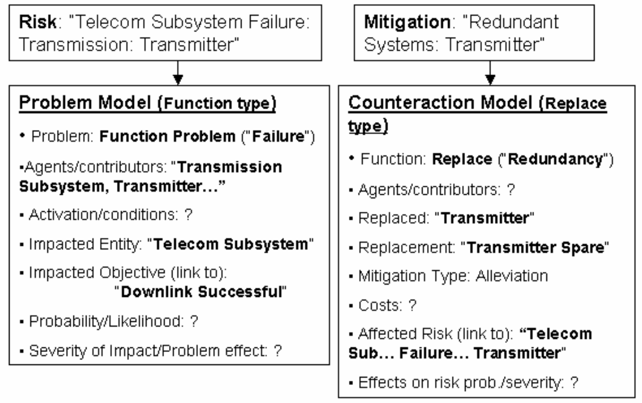

Risk: "Telecom Subsystem Failure: Transmission: Transmitter"
Mitigation: "Redundant Systems: Transmitter"
Problem Model (Function type)
• Problem: Function Problem ("Failure")
•Agents/contributors: "Transmission Subsystem, Transmitter..."
• Activation/conditions: ?
• Impacted Entity: "Telecom Subsystem"
• Impacted Objective (link to): "Downlink Successful"
• Probability/Likelihood: ?
• Severity of Impact/Problem effect: ?
Counteraction Model (Replace type)
• Function: Replace ("Redundancy")
• Agents/contributors: ?
• Replaced: "Transmitter"
• Replacement: "Transmitter Spare"
• Mitigation Type: Alleviation
• Costs: ?
• Affected Risk (link to): "Telecom Sub... Failure... Transmitter"
• Effects on risk prob./severity: ?

Figure 2 – Transforming risk and mitigation for the model

This approach provides an automated way to build and expand functional and risk models. It enhances HIT support for manual model building and risk capture by engineers. The HIT combination of functional architecture and embedded risk data also enables application of graph analysis and abstract high-level behavior simulation to identify and investigate the effects of problems and countermeasures in scenarios.

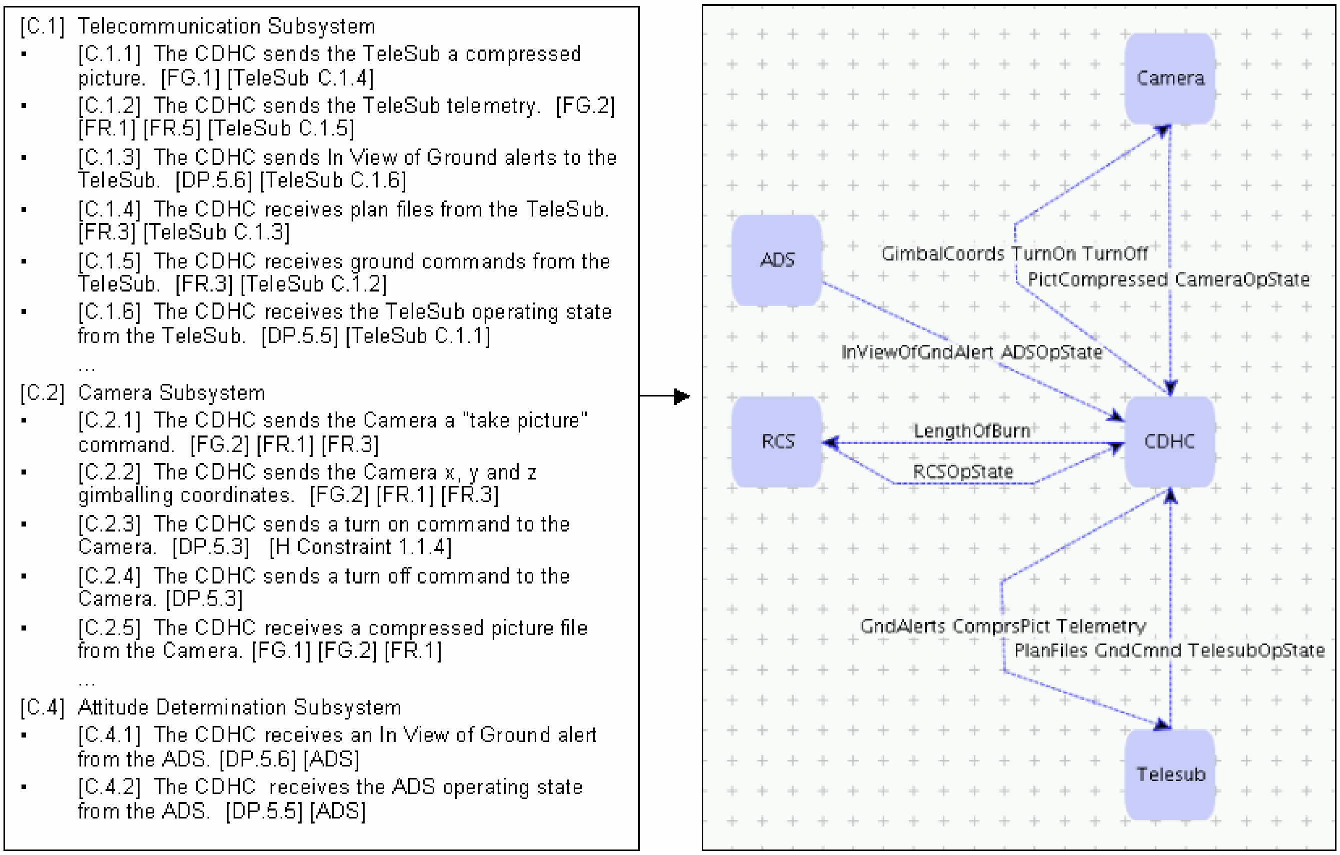
4. EXAMPLE CASE

These capabilities are designed to support the following type of scenario. In this scenario, the tools help the designers collect and integrate data, build models and explore their implications.

- Science spacecraft functional design by specialist teams: Telecommunication, Power, Thermal Control, Attitude Determination, Command and Data Handling…
- HIT system architecture models integrate information from teams and HIT libraries. As versions are developed and elaborated, more subsystems and risk countermeasures are included.
- HIT analysis of threats, vulnerabilities and paths in the system of subsystems finds new potential hazard interactions that countermeasures do not handle.
- CONFIG abstract hybrid simulation details the events the potential mishaps.
- Teams make design change: Enhance countermeasure strategy to handle the discovered interactions.
- Information feeds back to risk and requirements tools.

The case that illustrates our approach is a generic spacecraft that collects science data and transmits it to ground. Redundancy management for transmission is the risk mitigation that is the focus of this case. Autonomous fault management has become an important concern in satellite and spacecraft design [8].

We used requirements that had been independently developed in the SpecTRM specification tool, which focuses on software and transmission of commands and data. We parsed and transformed the requirements with Reconciler and derived an architecture model of the spacecraft subsystems described in the requirements. Figure 3 shows example requirements and the derived model. The Command and Data Handling Computer (CDHC) is the conceptual center of this model.



[C.1] Telecommunication Subsystem

- [C.1.1] The CDHC sends the TeleSub a compressed picture. [FG.1] [TeleSub C.1.4]
- [C.1.2] The CDHC sends the TeleSub telemetry. [FG.2] [FR.1] [FR.5] [TeleSub C.1.5]
- [C.1.3] The CDHC sends In View of Ground alerts to the TeleSub. [DP.5.6] [TeleSub C.1.6]
- [C.1.4] The CDHC receives plan files from the TeleSub. [FR.3] [TeleSub C.1.3]
- [C.1.5] The CDHC receives ground commands from the TeleSub. [FR.3] [TeleSub C.1.2]
- [C.1.6] The CDHC receives the TeleSub operating state from the TeleSub. [DP.5.5] [TeleSub C.1.1]

...

[C.2] Camera Subsystem

- [C.2.1] The CDHC sends the Camera a "take picture" command. [FG.2] [FR.1] [FR.3]
- [C.2.2] The CDHC sends the Camera x, y and z gimballing coordinates. [FG.2] [FR.1] [FR.3]
- [C.2.3] The CDHC sends a turn on command to the Camera. [DP.5.3] [H Constraint 1.1.4]
- [C.2.4] The CDHC sends a turn off command to the Camera. [DP.5.3]
- [C.2.5] The CDHC receives a compressed picture file from the Camera. [FG.1] [FG.2] [FR.1]

...

[C.4] Attitude Determination Subsystem

- [C.4.1] The CDHC receives an In View of Ground alert from the ADS. [DP.5.6] [ADS]
- [C.4.2] The CDHC receives the ADS operating state from the ADS. [DP.5.5] [ADS]

Camera
ADS
GimbalCoords TurnOn TurnOff
PictCompressed CameraOpState
InViewOfGndAlert ADSOpState
RCS
LengthOfBurn
CDHC
RCSOpState
GndAlerts ComprsPict Telemetry
PlanFiles GndCmnd TelesubOpState
Telesub

Figure 3 – SpecTRM Command and Data Handling Requirements (L) are Used to Derive Initial HIT Spacecraft Model (R)

Imported risk data (see Figure 2) indicates that the risk of transmitter failure can be ameliorated by using redundant transmitters in the design. A telecommunication designer alters the model to focus on data transmission and the redundancy strategy. The problem is elaborated to indicate that the transmission failure can be activated by noise generated by the failed transmitter. The controller design is elaborated to respond to degraded data rate due to noise. The engineer adds two types of memory and ground station and associated connections. Some Power and Thermal information becomes available and is incorporated in the model. New types of power and heat connections are added. The Thermal Control System model includes a hazard. It is a source of electrical noise when on. The noise can be carried by Power connections to Telecommunication System transmitters. Figure 4 shows the state of the architecture model at this point in design.

Then the engineer requests a static reachability analysis of system interactions, to evaluate potential risks. The static analysis first finds pairs of hazard sources and corresponding functions or components that are vulnerable to that hazard. Then the graph of the architecture model is searched (in paths that could transmit the hazard), to find a way that the hazard could reach the vulnerable entity.

A simplified version of the primary reachability rule is:

IF Component **C1** is vulnerable to Entities of type **E**

AND Component **C2** is a source of and Entities of type **E**

AND both **C1** and **C2** connect to paths of type **P** that can carry Entities of type **E**

AND there is a path **P1** by which entities of type **E** can reach **C1** from **C2**

THEN a potential hazard exists

Figure 5 shows the results of the analysis of the spacecraft model in Figure 4. It shows that external noise that can get to the transmitters over power connections. Transmitter redundancy would be an ineffective countermeasure for this threat.

This information can also be used to develop a simulation script for the abstract spacecraft model, to explore risk interactions in operations and evaluate countermeasure strategies. A CONFIG simulation model is automatically derived from the HIT spacecraft model. The common generic model types in HIT and CONFIG enable this transformation. Figure 6 shows the derived simulation model.

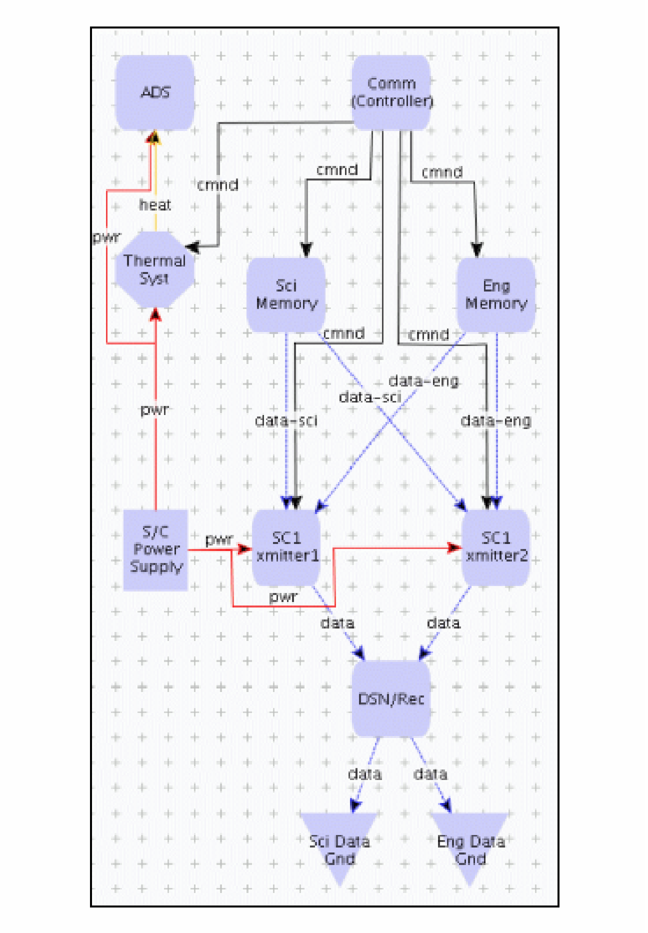
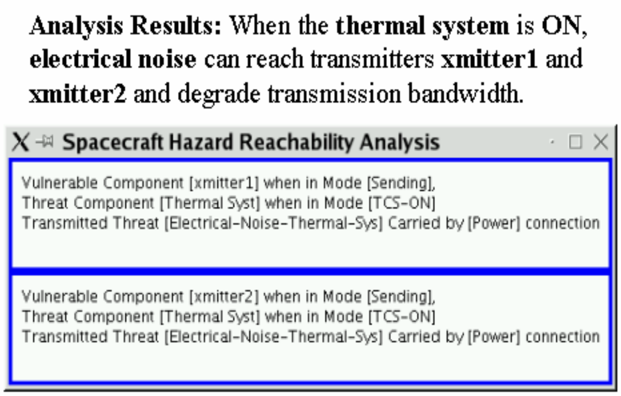


Figure 4 – Revised Spacecraft Model Architecture

The transmitters in the simulation are modeled as Servers in an abstract client/server model. These servers react to overload (more than 1.0 normalized capacity) by slowing the service rate (bandwidth). The Clients are science and engineering memory, each requesting transmission rates of 0.4, with total load of 0.8, within the Server capacity limit. Noise from the Thermal Control System can take up 0.3 of normalized server capacity.



Analysis Results: When the **thermal system** is ON, **electrical noise** can reach transmitters **xmitter1** and **xmitter2** and degrade transmission bandwidth.

Spacecraft Hazard Reachability Analysis

Vulnerable Component [xmitter1] when in Mode [Sending],
Threat Component [Thermal Syst] when in Mode [TCS-ON]
Transmitted Threat [Electrical-Noise-Thermal-Sys] Carried by [Power] connection

Vulnerable Component [xmitter2] when in Mode [Sending],
Threat Component [Thermal Syst] when in Mode [TCS-ON]
Transmitted Threat [Electrical-Noise-Thermal-Sys] Carried by [Power] connection

Figure 5 – Results of Spacecraft Reachability Analysis

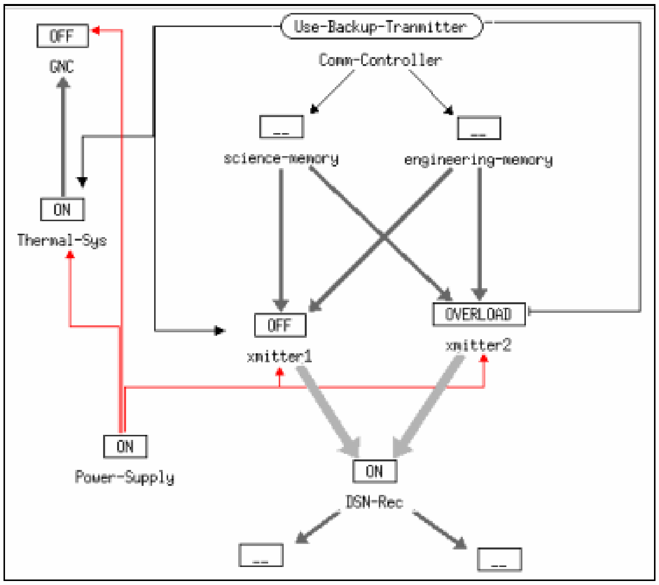


Figure 6 – CONFIG Simulation Model of Spacecraft

In the simulation script, the system is powered up and the Comm-Controller software state set to ON, starting data transmission. Then Thermal System set to ON, generating noise.

The event sequence in the simulation is summarized below.
1. Nominal Xmitter1 transmission rate is 0.8, the total requested rate.

2. When TCS is turned ON, noise travels from TCS to Xmitters via power connections and takes up 0.3 of capacity.

3. Xmitter 1 is overloaded (total 1.1 “requested”), changing transmitter data rate proportionally to 0.8/1.1 = 0.723 (too slow).

4. Control software unsuccessfully tries to compensate by switching to backup Xmitter2, but transmission rate is unchanged.

5. Failure: Transmission is not completed in specified time.

The engineer updates the noise countermeasure to inhibit operation of the Thermal Control System during transmission. An example abstract control model with a new inhibit mode is shown in Figure 7. Simulation can be used again to evaluate this change.

This new information can be fed back into the risks and requirements tools via a tool integration framework.

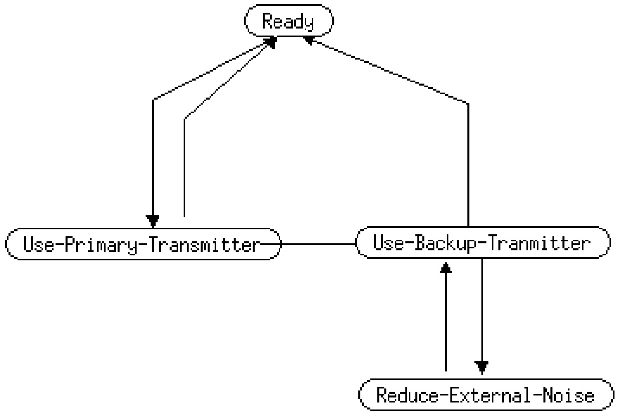


Figure 7 – Control model with mode for external noise

6. Conclusions and Recommendations

We have presented methods for transforming and integrating requirements and risk data so that it can be used in model-based analysis and simulation to explore system interactions and evaluate mitigations. This will be important for connecting a variety of tools that use different methods for capturing system and software functions, architectures and risks. We have shown how to use lightweight simulation of operations in early analysis, to explore system effects on risks and performance problems and impacts of these problems on the system and subsystems. There is much more to be done in this area. With these types of analysis, consideration of system health management needs can be an integral part of early design.

Currently, our requirements transformations focus on information about functional architecture. The methods, ontology and model library can be expanded to transform and include more types of requirements data in the future. The Reconciler approach should also make it possible to incorporate data from failure modes, effects and criticality analysis (FMECA) and software safety analysis [9] into the models.

Lightweight simulation of system effects in operations, in the context of functional hierarchies, should be applicable to automating generation of event trees and fault trees. We hope to further explore using the lightweight simulation approach to help the developers of event trees and failure effects analysis [10].

While exploring the spacecraft case, we discovered an interesting ambiguity used by designers when considering function, implementation, operation and behavior. We think our tools can benefit from permitting this ambiguity, and we have begun experimenting with providing linked functions, operating modes and implementing agents.

In the future, sub-functions could be identified that are services that prepare (and repair) participants in the function: functional agent, operand, environment and resources (supplies, energy, control). For example, the Power agent (subsystem) may need services to supply it with power, store the power and control the distribution. The destination, the Radar, may need to be enabled to receive power and cooled while powered.

In the future, we also want to further explore the interactive use of generic risk and countermeasure information that can be associated with entities in the HIT model library. Defaults can be used to aid and automate the process of enumerating and discovering hazards, vulnerabilities and countermeasures for types of subsystems, components and agents (human and software).

Acknowledgement

This work has been funded by the System Reasoning and Risk Management thrust area in the NASA Engineering for Complex Systems program. Thanks to Steven Cornford and Leila Meshkat for developing the generic spacecraft model case that has guided this work. Thanks to Martin Feather and Nancy Leveson and MIT graduate students for providing requirements and risk data for the case study.

References

[1] J. T. Malin, D. R. Throop, L. Fleming and L. Flores, “Computer-Aided Identification of System Vulnerabilities and Safeguards during Conceptual Design,” 2004 IEEE Aerospace Conference Proceedings, March 6-13, 2004.

[2] D C. Hendershot, R. L. Post, P. F. Valerio, J. W. Vinson, D. K. Lorenzo and D. A. Walker, "Putting the 'OP' Back in ‘HAZOP’," MAINTECH South '98 Conference and Exhibition, December 2-3, 1998.

[3] J. T. Malin, L. Fleming and D. R. Throop, “Predicting System Accidents with Model Analysis during Hybrid Simulation,” *Proceedings of Business and Industry Symposium, Advanced Simulation Technologies Confer.*, Simulation Councils, pp. 155-160. April 2002.

[4] J. T. Malin, L. Fleming and D. R. Throop, “Hybrid Modeling for Scenario-Based Evaluation of Failure Effects in Advanced Hardware-Software Designs,” Model-Based Validation of Intelligence, Technical Report SS-01-04, AAAI Press, Menlo Park, CA, 2001.

[5] D. Throop, “Reconciler: Matching Terse English Phrases,” Proceedings of 2004 Virtual Iron Bird Workshop, NASA Ames Research Center, April, 2004.

[6] M. Katahira and N. Leveson, “Use of SpecTRM in Space Applications”, 19th International System Safety Conference, Huntsville, Alabama, September 2001.

[7] S. L. Cornford, M. S. Feather, and K. A. Hicks, “DDP – A Tool for Life-cycle Risk Management,” 2001 IEEE Aerospace Conference Proceedings, March 2001.

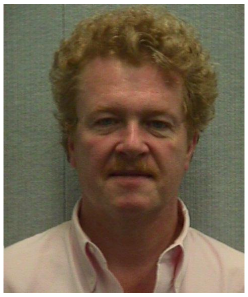
[8] R. D. Coblin, “Fault Management in Communications Satellites,” MILCOM ’99 Conference Proceedings, IEEE, 1999.

[9] N. Dulac and N. Leveson, “An Approach to Design for Safety in Complex Systems,” Intl. Conference on System Engineering (INCOSE '04), Toulouse, June 2004.

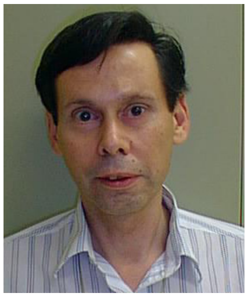
[10] D. R. Throop, J. T. Malin and L. Fleming. 2001. “Automated Incremental Design FMEA,” 2001 IEEE Aerospace Conference Proceedings, March 2001.

BIOGRAPHY

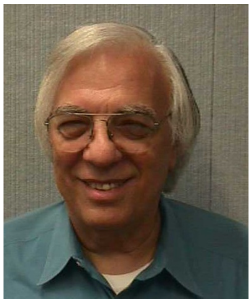
Jane T. Malin *is Senior Technical Assistant in the Intelligent Systems Branch, Automation, Robotics and Simulation Division, Engineering Directorate, NASA Johnson Space Center, where she has led intelligent systems research and development since 1984. She has led development of the CONFIG hybrid simulation tool. She has led research on intelligent user interface and intelligent agents for control of space systems, and on teamwork tools for anomaly response teams. Her 1973 Ph.D. in Experimental Psychology is from the University of Michigan.*



David R. Throop *has been an Artificial Intelligence Specialist with The Boeing Company since 1992. He provides engineering software support in the Intelligent Systems Branch in the Automation, Robotics and Simulation Division in the Engineering Directorate at NASA Johnson Space Center. He oversaw development of FMEA modeling software and its use for the International Space Station. His 1979 Bachelors of Chemical Engineering is from Georgia Tech. His 1992 Ph.D. in Computer Science is from the University of Texas, with a dissertation on Model Based Diagnosis.*



Land D. Fleming *is a Computer Systems Specialist supporting the NASA Johnson Space Center Automation, Robotics, and Simulation Division since 1990. He has been involved in both the development of computer simulation tools and their application to space systems. His 1987 M. S. in Computer Science is from De Paul University.*



Luis Flores *is a systems software engineer supporting the NASA Johnson Space Center Automation, Robotics, and Simulation Division since 1985. He has been involved in design and development of software using knowledge-based, intelligent control and computer simulation tools for space systems applications. His 1967 Ph.D. in Physics is from Texas A&M University.*