Information Extraction for System-Software Safety Analysis

Calendar Year 2007 Year-End Report

Jane T. Malin, Principal Investigator
Project: Automated Tool and Method for System Safety Analysis

Abstract This annual report describes work to integrate a set of tools to support early model-based analysis of failures and hazards due to system-software interactions. The tools perform and assist analysts in the following tasks: 1) extract model parts from text for architecture and safety/hazard models; 2) combine the parts with library information to develop the models for visualization and analysis; 3) perform graph analysis on the models to identify possible paths from hazard sources to vulnerable entities and functions, in nominal and anomalous system-software configurations; 4) perform discrete-time-based simulation on the models to investigate scenarios where these paths may play a role in failures and mishaps; and 5) identify resulting candidate scenarios for software integration testing. This paper describes new challenges in a NASA abort system case, and enhancements made to develop the integrated tool set.

Information Extraction for System-Software Safety Analysis

Calendar Year 2007 Year-End Report

Jane T. Malin, Principal Investigator
Project: Automated Tool and Method for System Safety Analysis
Automation, Robotics and Simulation Division, NASA Johnson Space Center
jane.t.malin@nasa.gov 281-483-2046

Problem Statement

Unsafe system-software interactions are a major concern in software validation and demonstrating software safety.[1] A unified, systematic, and automated approach is needed to validate system requirements and identify failures and hazards that NASA flight software is designed to handle. Early evaluation of software requirements and design will reduce system-software integration risks. It is important to identify requirements gaps and robustness issues early and often because relevant factors in complex controlled systems are easily overlooked. It is also important to assess system failures and anomalous conditions that may challenge software in system integration testing. As shown in figure 1, operations and stresses in software can ‘activate’ faults and influence failures in the controlled system (the “plant”) or the environment. Likewise, operations and stresses in the controlled system or the environment can ‘activate’ faults and influence failures in the software. Interacting cascades are possible.

Uniform automated methods are needed for extracting early information from requirements specifications, for system modeling for validation and safety analysis. Without these methods, quality is inconsistent from one project to the next. Probability increases that requirements-induced errors and hazards will propagate to subsequent development phases. In addition, excessive amounts of time can be consumed in reanalyzing modified or added requirements as projects progress. Semi-automated information extraction, model generation, and analysis save labor and schedule by using data extracted from documents. Automated information extraction can improve the efficiency, consistency, repeatability, and comprehensiveness of modeling and analysis, and it can reduce the time spent reanalyzing when specifications and designs change.

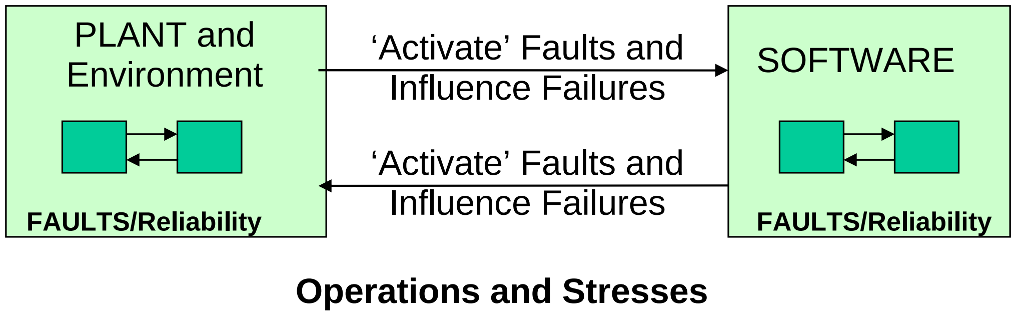


Figure 1. Concept of system-software interactions related to hazards.

Technical Approach

The goal of the research is to enable early model-based system-software failure and hazard analysis during requirements and design phases. The approach is to integrate and enhance previously developed prototype tools for text information extraction, system architecture modeling, simulation, and analysis. The feasibility of this integrated approach has been demonstrated.[2,3] Figure 2 shows a diagram of the relationships among the prototypes in the project. This diagram is taken from the Concept of Operations document for the project,[4] which provides a functional description of the proposed system and its components and defines operational scenarios for tasks supported by the system. Products of the system support personnel in the safety and mission assurance (SMA) organization.

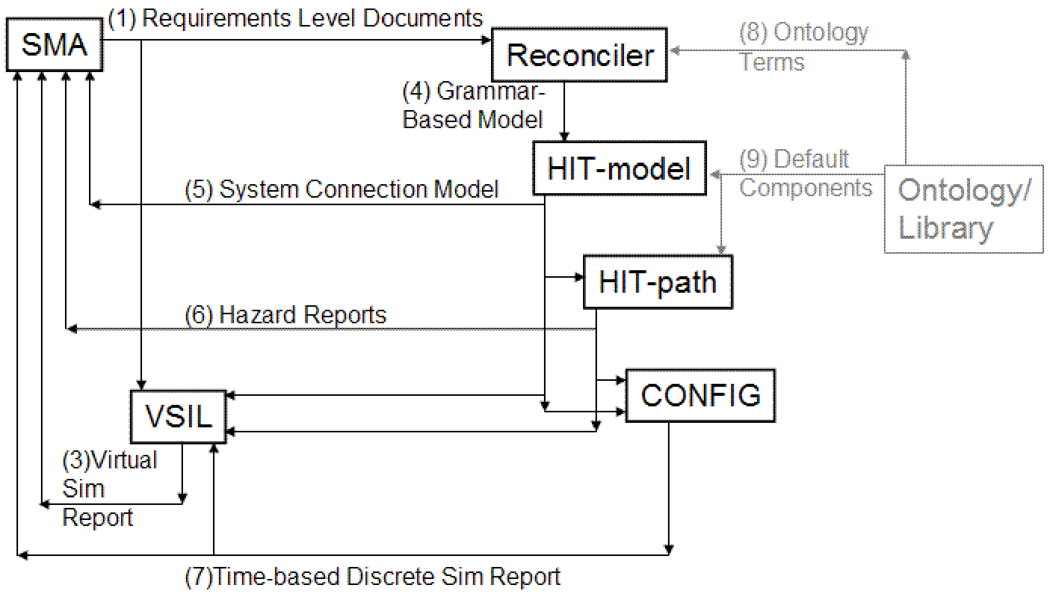


Figure 2. Component diagram of model-based system-software safety analysis.

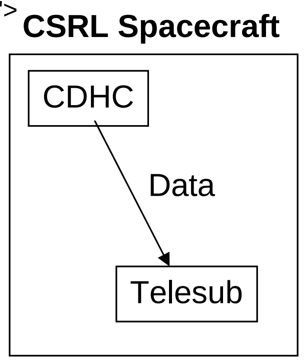
Reconciler[5] and the Aerospace Ontology[6] are used together for semantic analysis and extraction of models from requirements and design texts. Parts of abstract physical architecture models are extracted to model and analyze the controlled system architecture. Systems, subsystems and components and their interfaces are extracted. Operational modes and functions, constraints, sensitivities, hazards, and risks are also extracted.

The Hazard Identification Tool (HIT) Modeler module is used to develop system architecture models from the extracted model information by using model structures that map to the types of extracted information. Libraries of component types are used to fill in missing information with defaults. HIT Modeler also provides a visualization for inspecting the architecture implied by the requirements and design documents.

The HIT Path Analyzer module[7] and the CONFIG hybrid simulator[8] are used to analyze hazard paths and simulate risk propagation in the system during operational and off-nominal scenarios. CONFIG has been used previously for validation testing of intelligent control software for gas storage and transfer in a manned life support test. Among other things, deficiencies in the software requirements were identified. Analysis and simulation is used to identify possible hazard paths in test scenarios for a virtual system integration laboratory (VSIL) for software testing.[9]

Initial Model Feasibility and Integration Case. The first project case used a portion of a generic spacecraft model to evaluate modeling feasibility, using XML formats for communicating model parts and models. The HIT Modeler was re-implemented to use CONFIG model building and model display capabilities. Figure 3 shows a data connection between two spacecraft subsystems as it was extracted, modeled, and visualized. The models of the subsystems or components can also contain information on hazards and sensitivities to potential hazards.

```
<MODEL name="CSRL Spacecraft" type="SYSTEM-UNIT">
  <COMPONENTS>
    <COMPONENT name="CDHC" type="SYSTEM-UNIT">
    </COMPONENT>
    <COMPONENT name="TELESUB" type="SYSTEM-UNIT">
    </COMPONENT>
  </COMPONENTS>
  <CONNECTIONS>
    <CONNECTION>
      <FROM_DEVICE name="CDHC">
      </FROM_DEVICE>
      <TO_DEVICE name="TELESUB">
      </TO_DEVICE>
      <ENTITY-TRANSFERRED name="DATA">
      </ENTITY-TRANSFERRED>
    </CONNECTION>
  </CONNECTIONS>
</MODEL>
```



CDHC: Command and Data Handling Computer
Telesub: Telemetry subsystem

Figure 3. Simple spacecraft architecture model: HIT model output.

NASA Orion Launch Abort System Case. To challenge and extend the technology, a large-scope space system case has been selected: analysis of system-software interactions and hazards for a pad abort test of the Orion spacecraft launch abort system (LAS). NASA will perform a series of six abort flight tests at White Sands, New Mexico, starting in the fall of 2008 and ending in 2011. In the two pad abort tests, the LAS will be used to safely propel the crew module (CM) boilerplate mockup away from the launch pad. The pad abort happens in two stages. First, abort and attitude control motors are used to propel the launch abort vehicle (LAV), which is made up of CM and LAS, from the Ares launch vehicle or the launch pad, toward a suitable landing area. Second, the LAV reorients for landing, the LAS is jettisoned, fuel is dumped, and parachutes are deployed for safe reentry and landing of the CM. Software-related test objectives of the first pad abort test include demonstrations of telemetry transmission from the CM through the LAS boost protective cover, abort events sequencing, parachute and landing sequencing, and functional performance of the ground support command, control, and monitoring system.

There are new types of subsystems, functions, and hazards in the LAS domain. The information to be extracted is rich, varied, and distributed throughout many requirements and design documents in various formats. These include flight test objectives, requirements, failure mode and effects analysis (FMEA), risk analysis, interface requirements documents, and interface control documents. The sentence structures in rationales and descriptive introduction sections can be very complex. Much of the nomenclature in the LAS domain is NASA-unique.

Reconciler Information Extraction. Reconciler has been used to parse text in aerospace database records, including space shuttle problem reporting and corrective action (PRACA) and space station reliability block diagrams, in space station documents, and in requirements specifications in SpecTRM[10] and Cradle[11] requirements specification tools. For architecture modeling, Reconciler should process text to extract several types of information (when available): 1) system-subsystem-component decomposition relationships; 2) interfaces and connections; 3) functions or actions of devices; 4) operational and failure modes of devices; 5) variables and events being controlled by the system devices and software; and 6) enablers, disablers, limits, hazards, sensitivities, risks and failures, and their controls or mitigations.

A complex sentence from an overview of the Orion launch abort system illustrates the content and the challenges.

> The LAS consists of a nose cone, a canard section which enables the LAS to reorient the CM for parachute deployment following an abort, three propulsive motors (attitude control, jettison, and abort), an adapter cone that provides the structural interface to the CM, and a boost protective cover (BPC) sized for ascent heating to protect CM thermal protection system (TPS) coatings.

An ambitious text processing goal is to build an XML structure expressing an indentured structure like the one in figure 4. The structure in figure 4 is a human-generated performance benchmark for Reconciler, to measure the amount of model information Reconciler extracts compared to model information available in a piece of text. Modules have been added for extracting text from Adobe document format and extracting acronyms. To extract some of the structure in figure 4, new parsing functionality has been added to extract acronyms, part-of relationships, and functional assignments and their rationales.

Aerospace Ontology. This standard nomenclature for hardware, software, and human systems defines hierarchies of types of subsystems, functions, entities, hazards, and failures. It provides sets of mapping words so that many synonyms are accommodated. The nomenclature and class hierarchies in the Aerospace Ontology should reflect the LAS domain to support information extraction and modeling, and they should map to the classes and library items that are needed for HIT modeling and analysis. Classes and associated vocabulary words have been added to the Aerospace Ontology to accommodate LAS things and problems, vulnerabilities, and threats. Some of these additions are shown in figure 5.

Top Level: LAS
- Function: reorient the CM
  - Agent: LAS
  - Action: reorient
  - Operand: CM
- Function: control attitude
  - Agent: LAS?
  - Action: control
  - Operand: ?
    - Variable: attitude
- Function: jettison
  - Agent: LAS?
  - Action: jettison
- Function: abort
  - Agent: LAS?
  - Action: abort
- Function: deploy parachute
  - Agent: LAS?
  - Action: deploy
  - Operand: parachute

LAS Components
- **Component:** Nose cone
- **Component:** Canard Section
  - Function: ?
  - Component Enables Function: LAS: reorient the CM
    - Function Enables Function: LAS?: parachute deployment (following an abort)
- **Component Group:** Three propulsive motors
  - **Component:** attitude control motor
    - Enables Function: LAS?: control attitude
  - **Component:** jettison motor
    - Enables Function: LAS?: jettison
  - **Component:** abort motor
    - Enables Function: LAS?: abort
- **Component:** An adapter cone
  - Connection
    - Connector: adapter cone
    - From: LAS
    - To: CM
    - Type: Structural
- **Component:** Boost protective cover
  - Acronym: Boost protective cover = BPC
  - Design parameter: size
    - Determined by: ascent heating
  - Function: protect CM thermal protection system coatings
    - Agent: Boost protective cover
    - Action: protect
    - Operand: CM thermal protection system coatings

Other: CM
- **Component:** Thermal protection system
  - Acronym: thermal protection system = TPS

Figure 4. Model information manually extracted from a complex LAS overview sentence.

Entities

- Entity Flows (for connection classes)
  - Data/Signal: data signal, data, signal, telemetry, video, voice, image, audio, command
  - Electrical/Data Noise: electrical noise, EMI, radio frequency interference, RFI, static, electromagnetic noise, bleed over…
  - Propellant: hypergolic fuel, hypergolic oxidizer, solid propellant, oxidizer, LOX, nitrogen tetroxide, (monomethyl) hydrazine, MMH…
  - Energy/Power: energy, thermal energy, acoustical energy, chemical…
    - Mechanical Energy: hydraulic energy, power, force, load, flow…
    - Radiant Energy: radiant energy, illumination, lighting, light, heat…
    - Electrical Energy: electrical energy, current, alternating current…
- Subsystems and Equipment
  - Subsystems: Launch Site Operation; Abort Systems; Avionics Subsystems; Communications…
  - Fluid Storage; Information Storage; Pyrotechnic Equipment; Electrical Equipment (generate, store…), Motors, Control Jets; Seals; Fittings; Joints
  - Instrumentation: Sensors, Controllers, Actuators
  - Safety/Prevention Equipment for Mechanical, Electrical, Software
    - Software Safety/Prevention Equipment: firewall, anti-virus, packet filter, proxy server, screening router, access control list…

Problems, Vulnerabilities and Threats (to capability, integrity/reliability, performance timing and quality, or controllability)

- Action/Functions: Not Robust: vulnerable, sensitive, brittle, not stable...
- Objects: Vulnerable: Not Robust to Damage/Impairment Source
- Software Threat: spyware, spam, virus, malware…
- Software Vulnerability: dangling pointer, format string vulnerability, code injection…
- Bad Software Structure
- Missing Software Structure

Counteractions

- Maintain (good state): Preserve; Ensure
- Prevent (bad state): Avoid; Withstand; Decrease Sensitivity; Guard
- Minimize
- Rectify (from bad state): Neutralize; Safe: Restore (Substitute, Repair, Renew); Rework (redo); Undo; Reset

Figure 5. New classes and nomenclature for the LAS case.

<u>Hazard Identification Tool Modeler and Library.</u> The new version of the HIT modeler uses the graphical modeling elements of CONFIG modeling and simulation tool for automatic and manual model construction and visualizing the architecture models. HIT models represent objects that are system components or things that are processed. Models represent function types for: 1) Placing: transporting, transferring, and changing connectedness; 2) Processing objects: transforming (e.g., phase change) or changing a parameter; and 3) Processing information: monitoring, controlling, communicating, regulating, commanding, assisting, counteracting/mitigating. Models also represent influences on functionality and problem types for hazard analysis: 1) Damage and impairment sources: burdens/shocks, threats, and material hazards; and 2) Vulnerabilities or lack of robustness in functions and objects.

Deleted: <u>Aerospace Ontology.</u> This standard nomenclature for hardware, software, and human systems defines hierarchies of types of subsystems, functions, entities, hazards, and failures. It provides sets of mapping words so that many synonyms are accommodated. The nomenclature and class hierarchies in the Aerospace Ontology should reflect the LAS domain to support information extraction and modeling, and they should map to the classes and library items that are needed for HIT modeling and analysis. Classes and associated vocabulary words have been added to the Aerospace Ontology to accommodate LAS things and problems, vulnerabilities, and threats. Some of these additions are shown in figure 5.¶

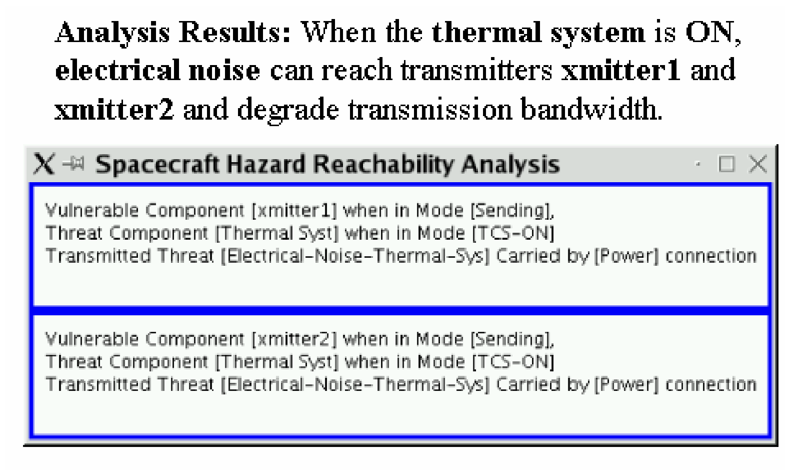
The HIT Modeler should provide model classes with attributes that map to the extracted information types, to enable semi-automated development of component-connection models with embedded hazard information and functions. Some of those attribute structures are shown in figure 6. The HIT Modeler should also provide a library of component types for the LAS domain that can be used for defaults when the information is not available to be extracted. Default hazards, sensitivities and enablers, disablers, or influences on performance can be used to create plant models with embedded safety issues, for integration testing.

Class Structures for Library for Objects, Functions, Influence Relationships, Mitigations

- Component/Subsystem classes with attributes for:
  - Interfaces and Input/Output Parts/Constituents, Parameters
  - Modes/States (with associated Functions)
  - Problems, Hazards, and Sensitivities
- Function/Action classes with attributes for:
  - Agents, Operands/Patients, Origins/Destinations, Variables
  - Functional Dependency Relations, associated modes
  - Problems, Hazards, and Sensitivities
- Influence Class with attributes for:
  - Positive-Negative (signed) relation to influenced variable or problem
  - Importance (degree of worst-case impact)
  - Likelihood (probability of occurrence of factor)

Figure 6. Some new draft domain library classes with types of attributes.

Hazard Identification Tool Path Analyzer. The HIT Path Analyzer is a simple, early form of automated hazard analysis that can be refined as more design information becomes available. Figure 7 shows the output from a demonstration of the HIT Path Analyzer when search is dependent on a dynamic configuration in a scenario. Simpler analyses find matching hazard-vulnerability component pairs, or they perform static path analysis on connections or dependencies. Analysis data include cause-condition-vulnerability paths and cascades (hazards and failure effects), as well as effectiveness (severity) scores. Severity scores are computed during graph analysis, based on ratings of local effectiveness of hazards, connections, and mitigations. The analysis helps identify scenarios for software integration testing. The module should be capable of handling scenarios that address the types of LAS problems that interact with the software.


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 7. Output from path analysis demonstration case.

The Aerospace Ontology class hierarchy has been reorganized for compatibility with basic actions and functions that are important for path analysis, as shown in figure 8.

- Place/Arrange
  - Move + *EntityOperand* + *Path*
    - Transport + *SourcePlace* + *DestinationPlace*
  - Change "Owner"
    - Transfer + *EntityOperand* + *Source* + *Sink*
    - Input/Output + *EntityOperand*
      - Output: Emit (Active-Output); Release (Passive-Output)
      - Take-In: Input (Active Take-In); Receive (Passive Take-In)
- Process
  - Transform + *EntityOperand* + *Parameter*
    - Phase change, change in composition…
  - Change Position on a Scale + *EntityOperand* + *Parameter*
    - Increase
    - Decrease
- Control
  - Regulate + *EntityOperand* + *Parameter*

Figure 8. Action primitives for path analysis.

CONFIG Hybrid Simulator. When HIT has identified potential hazard scenarios, possibly including mitigations, CONFIG can be used to analyze them in a time-based numerical simulation that accommodates design information that is abstract or approximate. It combines discrete event and discrete time simulation technology with a capability to dynamically change configuration and underlying models during simulation. CONFIG can provide lightweight analysis of cascades of events and of the plausibility and severity of potentially hazardous scenarios. Reuse of CONFIG for software validation testing can uncover both traditional software problems and some issues concerning software requirements. The most interesting issue in the life support evaluation involved a complex interaction between elements of the crew chamber and the plant-growth chamber. It is not likely that this type of software problem would have been found during conventional software testing because the failure response involved a "surprising" sequence of interactions among multiple devices and controllers in the system. This sequence would be difficult to conceive of or emulate in conventional software testing.

HIT provides a specification for a CONFIG system model and also provides a hazard scenario to CONFIG. CONFIG user interface now provides the basis for implementing HIT models and visualizations. Therefore, evolving from HIT models to more detailed CONFIG models is easier than in the previous implementation. It will continue to rely on correspondence between component types in the HIT and CONFIG libraries.

Calendar Year 2008-2009 Milestones

Mar 2008:	Complete Orion model extraction and analysis case
Jun 2008:	Comparative evaluation of capabilities complete, preliminary methods document
Sep 2008:	Complete first version of integrated tool suite
Jan 2009:	CEV requirements and design data extraction complete
Mar 2009:	HIT modeling and analysis complete
Jun 2009:	CONFIG modeling and analysis complete, test definition complete
Sep 2009:	Complete enhanced version of integrated tool suite
Dec 2009:	Evaluation and documentation complete

Calendar Year 2008 Technical Plans

Work in 2008 will continue to focus on enhancements to Reconciler, HIT, and the Aerospace Ontology for LAS model extraction, development, and analysis. Tasks include identifying and documenting methods for performing extraction and analysis, as well as identifying LAS document parts and formats that are the best sources of the modeling information. User interface improvements will be made for use by SMA personnel.

The scope and performance of the tools for extraction, model development, and analysis will be evaluated: 1) the mix of automated and manual tasks and the manual time and effort required; and 2) the relevance and usefulness of the outputs to SMA personnel and to VSIL. Software performance will be evaluated: 1) comparison of the amount of model information extracted automatically with the amount that could be extracted manually; 2) comparison of the hazards and failures identified by graph analysis with those found with standard hazard analysis; and 3) comparison of ease of current test generation methods with test generation by graph analysis and simulation.

There will be a number of new technical challenges. One is extraction of scenarios for graph analysis and simulation. Another is use of visualizations and model verification to identify requirements gaps and inconsistencies. These visualizations are unique because they combine success and failure spaces, and system and operation/event spaces. Another is reuse and versioning of models and analysis to evaluate requirements and design changes. New tool capabilities are planned to support multiple model versions, check changes, and report results of comparisons.

Deleted: will

Conclusion

The overall approach permits application of graph analysis and abstract simulation to perform model-based quick-look investigations of the implications of specifications for safety. Thus, it is useful for evaluating completeness and consistency of requirements and risk. Reliability and probability of events could be used to enhance these analyses.

This type of architecture model is unique because it combines success and failure spaces, and system and operation/event spaces. In addition to the planned uses, this modeling approach could be used to demonstrate and evaluate safety cases. This approach is also promising for validating models and analyses of failure paths, FMEAs, and fault trees.

Deleted: ¶

References

[1]Heimdahl, M. P. E., “Safety and Software Intensive Systems: Challenges Old and New.“ International Conference on Software Engineering 2007 Future of Software Engineering, 2007, 137-152.

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

[3]Malin, J. T., Throop, D. R., Fleming, L., and Flores, L., “Transforming Functional Requirements and Risk Information into Models for Analysis and Simulation,” 2005 IEEE Aerospace Conference Proceedings, March 2005.

[4]Software Concept of Operations: For the System Models for Software Safety Analysis System. NASA Johnson Space Center, Automation, Robotics and Simulation Division.

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

[6]Malin, J. T. and Throop, D. R., “Basic Concepts and Distinctions for an Aerospace Ontology of

Functions, Entities and Problems," 2007 IEEE Aerospace Conference Proceedings, March 2007.

[7]Malin, J. T. and Fleming, L., "Vulnerabilities, Influences and Interaction Paths: Failure Data for Integrated System Risk Analysis," 2006 IEEE Aerospace Conference Proceedings, March 2006.

[8]Malin, J. T., Fleming, L., and Hatfield, T. R., "Interactive Simulation-Based Testing of Product Gas Transfer Integrated Monitoring and Control Software for the Lunar Mars Life Support Phase III Test," Proceedings of SAE 28th International Conference on Environmental Systems. SAE Paper No. 981769, 1998.

[9]Bennett, T. L. and Wennberg, P. W., "Eliminating Embedded Software Defects Prior to Integration Test," CROSSTALK: The Journal of Defense Software Engineering, December 2005.

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

[11]Cradle Web site: http://www.threesl.com/