Information Extraction for System-Software Safety Analysis

Calendar Year 2008 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 and simulation to identify and evaluate possible paths from hazard sources to vulnerable entities and functions, in nominal and anomalous system-software configurations and scenarios; and 4) identify resulting candidate scenarios for software integration testing. There has been significant technical progress in model extraction from Orion program text sources, architecture model derivation (components and connections) and documentation of extraction sources. Models have been derived from Internal Interface Requirements Documents (IIRDs) and FMEA documents. Linguistic text processing is used to extract model parts and relationships, and the Aerospace Ontology also aids automated model development from the extracted information. Visualizations of these models assist analysts in requirements overview and in checking consistency and completeness.

Information Extraction for System-Software Safety Analysis

Calendar Year 2008 Year-End Report

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

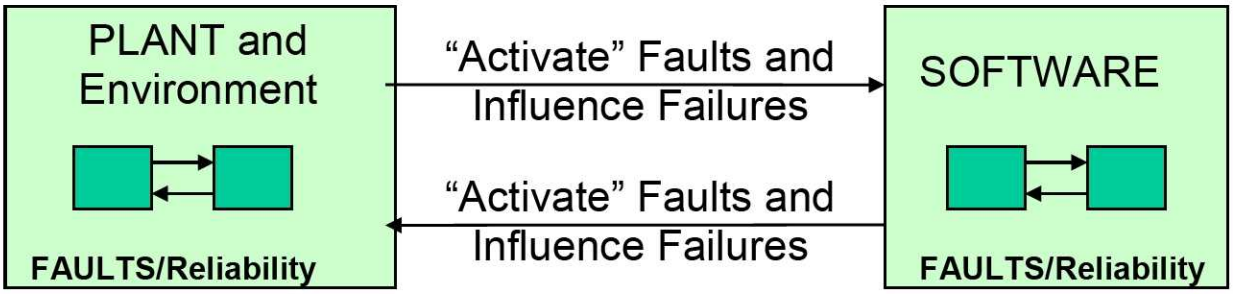
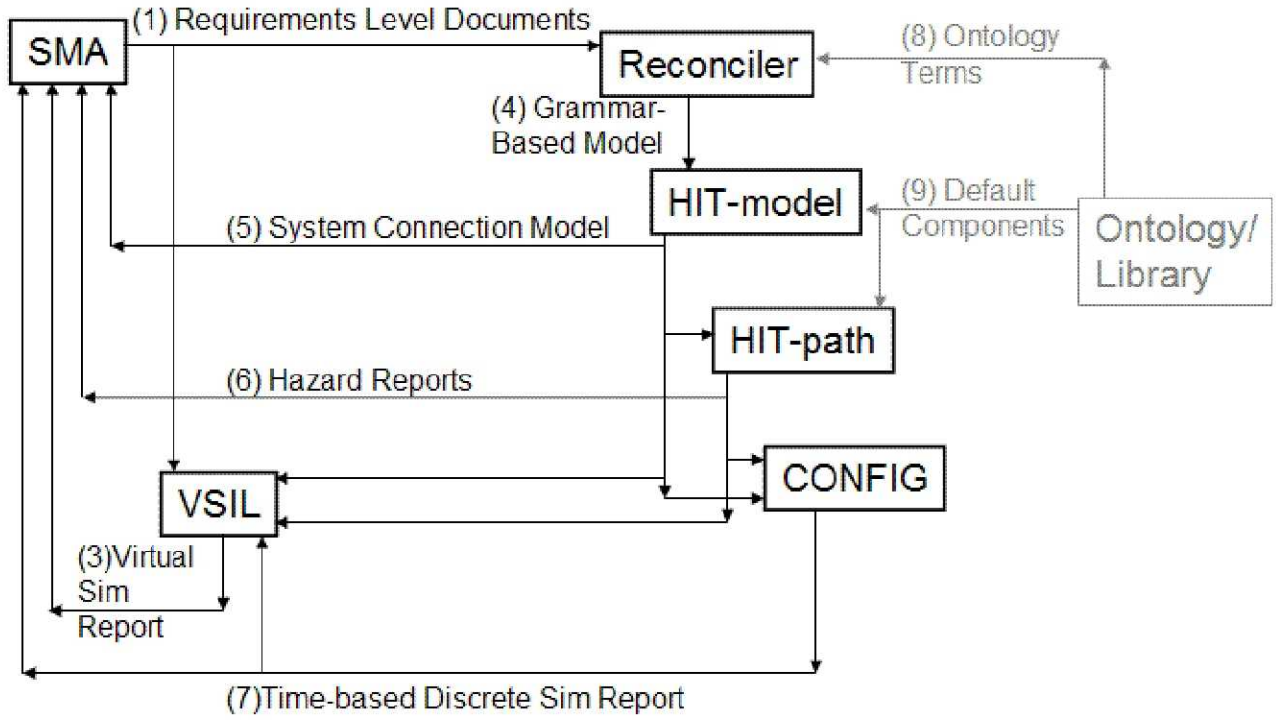


Figure 1: Concept of system-software interactions related to hazards.

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 safety and mission assurance (SMA) analysts.



Extraction → Modeling → Path Analysis → Simulation → Testing

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 will also be extracted.

The Hazard Identification Tool (HIT) Modeler module is used to semi-automatically develop system architecture models from the extracted model information by using model structures that map to the types of extracted information. This module uses the Aerospace Ontology to screen sentence interpretations for the types of extracted entities and weed out spurious mappings to model structures. Libraries of component types can 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]

Technical Challenges

Documents containing requirements, design, hazard analysis, and Failure Modes and Effects analyses (FMEAs) have proven to be incomplete and preliminary in the period prior to the project Preliminary Design Review. Needed model information can be missing, and model information can be redundant or internally inconsistent within a single document. This makes it difficult to perform simulation or path analysis without significant human intervention. On the other hand, model visualizations can help analysts reconcile and complete architectural models of the controlled system. These visualizations can also help analysts find discrepancies and inconsistencies in requirements and safety analyses.

Goals and Progress

Summary

There has been significant technical progress in model extraction from Orion program text sources, architecture model derivation (components and connections) and documentation of extraction sources. Models have been derived from Internal Interface Requirements Documents (IIRDs) and FMEA documents. Linguistic text processing is used to extract model parts and relationships, and the Aerospace Ontology also aids automated model development from the extracted information. Visualizations of these models assist analysts in requirements overview and in checking consistency and completeness. Types of interface components (e.g., Remote Interface Unit [RIU]), types of carriers and types of distributors (e.g., manifold) are represented in the model and the visualization. For example: A carries energy to B; A provides power to B; C receives command data from B; A distributes energy to Bs and Cs.

Some work has been done on simulation and path analysis that uses the extracted models, but the bulk of that work will be done next year. This work is dependent on extraction of functions and failures from FMEA documents. Extraction of these types of information is in progress.

Advice and feedback from Flight Software Engineering and Orion Software Safety and Mission Assurance have successfully guided the project work.

NASA Orion Launch Abort System Case

A NASA expert on Orion Launch Abort System (LAS) software helped identify LAS document parts and formats that would be the best sources of the modeling information. This led to a focus on IIRDs and Preliminary FMEAs. These were both available and had the type of architecture and scenario information that is needed for modeling. The expert suggested that the best candidates were the inertial sensors or the paths and interactions for firing pyrotechnic separation mechanisms during the Pad Abort sequence. Figure 3 shows a notional Pad Abort Sequence for Orion. No direct command feedback was expected in the design. The Crew Module (CM) software and computers connect to the Pyrotechnic Event Controllers (PECs) in the RIU, which connect to the LAS Pyrotechnics, The PECs control ignition of the LAS motors, including the Abort Motor (AM), Jettison Motor (JM) and Abort Control Motor (ACM). The RIU also contains PEC Control Output and PEC Power Supply Shield modules. Figure 4 shows the LAS pyrotechnics, which includes PECs, NASA Standard Initiators (NSI), and the NASA Standard Detonators (NSD) that are ignited by currents controlled by the RIU PEC outputs. Three commands are required to ignite a NSI: ARM; Fire-1; and Fire-2.

This year, the project has focused on modeling and developing visualizations of the pyrotechnics architecture. Primary extraction sources have been the Orion and Crew Exploration Vehicle (CEV) FMEAs, including the CEV-S-009 Avionics FMEA, as well as the CEV-T-035101, the CEV IIRD. The IIRD describes required interfaces between the three modules within the CEV elements: the Service Module (SM); the CM; and the LAS.

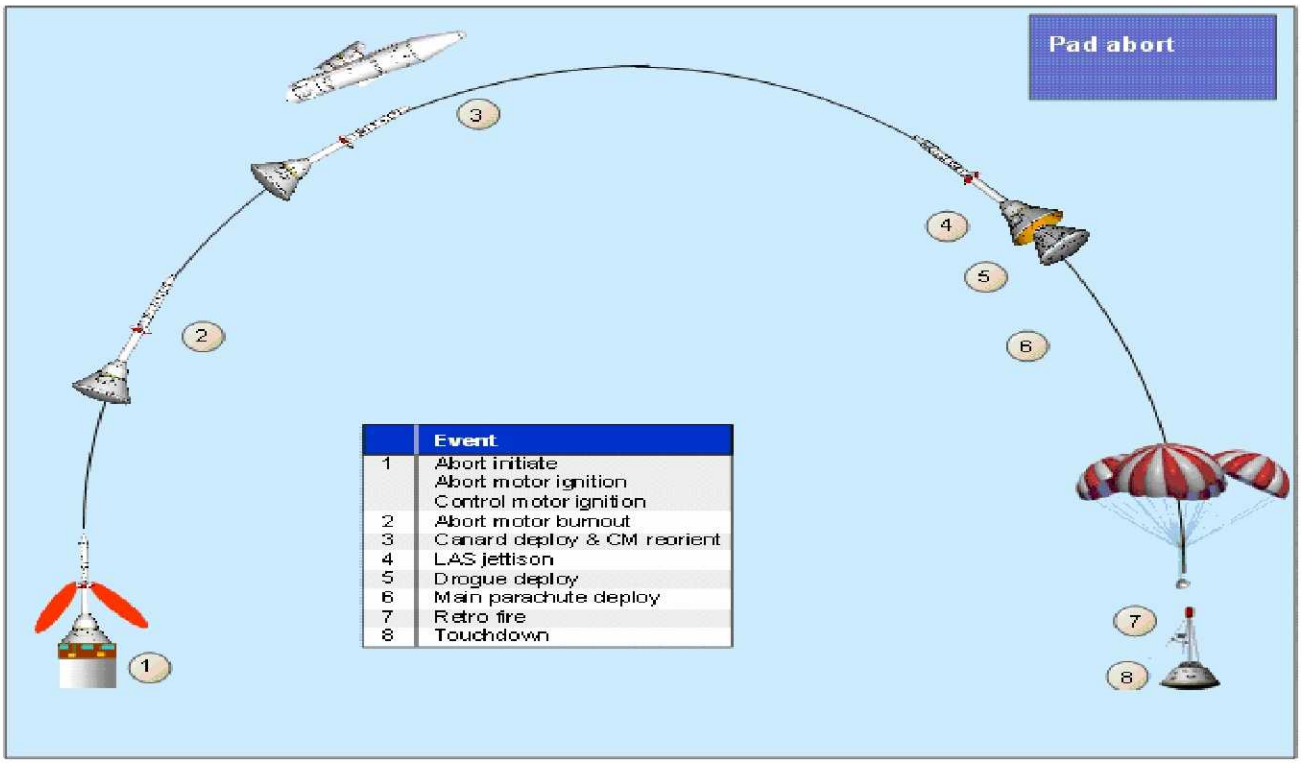


Figure 3: CEV Pad Abort Sequence – notional.

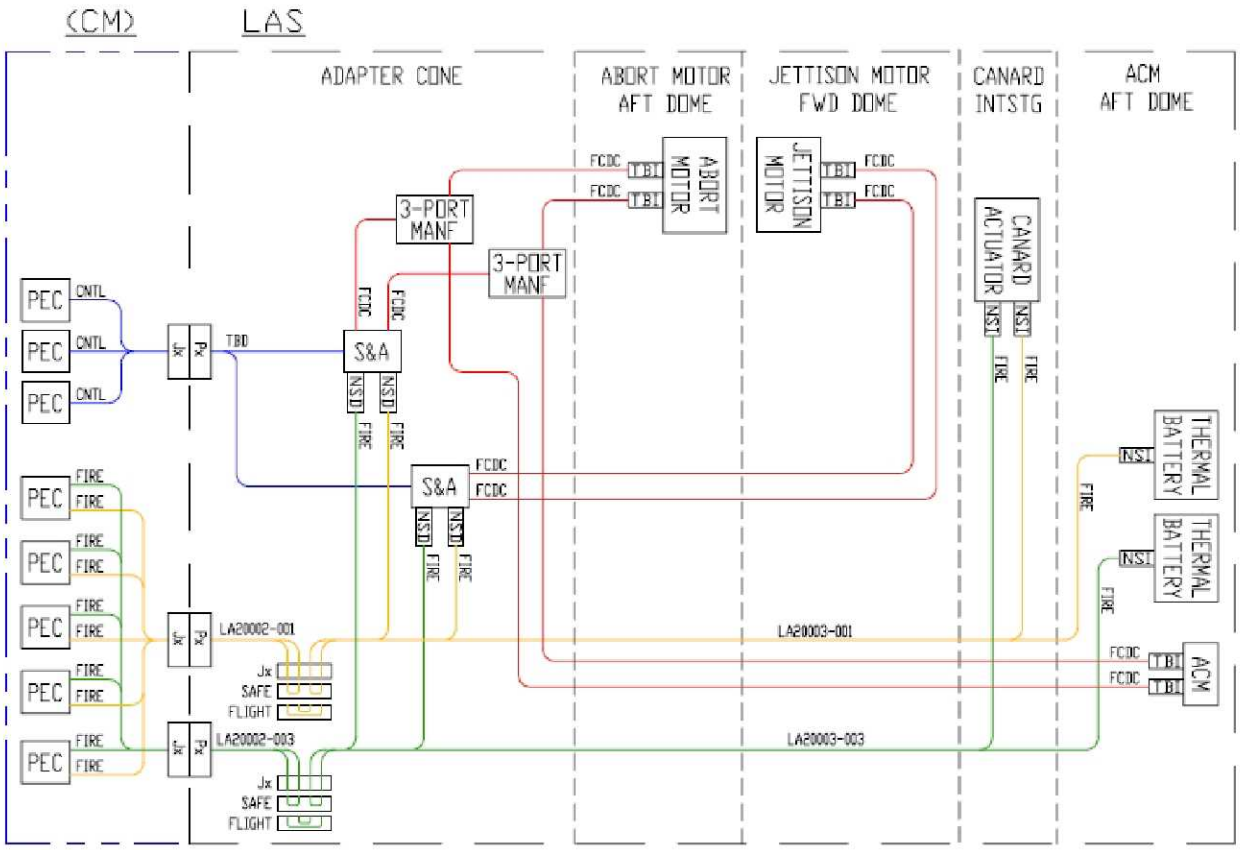


Figure 4: LAS Ordnance Subsystem Schematic.

Model Information Extraction

Model information extraction is accomplished by converting documents to structured text and extracting model parts from selected relevant sections. The software that performs linguistic processing of relevant sentences has been named the Semantic Text Analysis Tool (STAT). STAT output in xml format shows part-of and connection relationships among model components. A detailed procedure is shown in Figure 5 and described further in the sections that follow.

- Select documents to serve as source for models
- Convert documents into extractable formats
- Select relevant sections within the documents
- Specify
    - The fields to be extracted from sections
    - Which fields are text to be parsed as English sentences
    - Input / output directories and other run parameters
- Run extraction software, which
    - Reads in specifications and source files
    - Builds Perl data structures corresponding to document indenture
    - Reads Aerospace Ontology and knowledge base files
    - Calls parser (Reconciler or Charniak) on English text
    - Tags relationships
        - Tag *subject / verb / object*
        - Tag part-of and connection relationships
        - Annotates the Perl structure with these tags
    - Translates Perl structure to xml; writes xml structure report
    - Logs difficulties, unparsable text, unknown words, etc
- Pass xml structure report to HIT software

Figure 5: Procedure for model information extraction.

Document-structure specification grammar and extraction software

FMEA documents and Requirements documents are the main sources of model information. These documents have been generated from a database. For contractual and programmatic reasons, there is not timely, direct access to the database, or even its table structure. There is timely access to the documents. These documents are in proprietary Microsoft Word or Adobe pdf (Acrobat) formats. Several public-domain text extraction modules were tried, but proved to be error-prone. The ”save as text” capabilities of Word and Acrobat have worked the best.

Each document has a readily apparent regular structure of numbered indentured document sections. Within a single section, there is additional regular structure. For example, within a FMEA Appendix:

- Each FMEA worksheet has an item-name, author, item-part-number, and a numbered set of Failure Modes.
- Each Failure Mode has a worksheet number, date, and a numbered set of Causes.
- Each Cause has a name, phase, and description.

A grammar is used to specify document structure. The grammar, coded in Perl, is used to break apart the document and organize its parts into structured text. A full listing of the grammar is not in the scope of this report, but Figure 6 shows a sample fragment of a specification, with comments. Specifications are written out in detail for families of documents such as FMEAs, hazard analyses, or interface requirements.

The substructure varies slightly from one FMEA or Requirement document to the next. Subfields and numbers of columns in tables change across documents. The same information can appear once on a

single line, or may be broken across several lines. The specification for an individual document references the family specification, and adds (or overwrites) specification details particular to it.

When a document is analyzed, the extraction software builds a hierarchical Perl structure that corresponds to the extracted document structure. For this project, structured text has been extracted from five Requirement documents and three FMEA documents. Extraction is accomplished by using the grammar and associated software modules. Using the same grammar and software, an analyst on a related project has extracted requirements and parent-child relationships from over 50 Orion software requirements documents.

```
                    # First line is 'Appendix M1' and a title.
FMEA_Worksheets
   => {recognizer  => qr/^(appendix \s+ M\d*) \s*\-+ \s* (.+\D)\s*$/xi,
       nameFromRec => [qw(name title)],
       subsections => [qw(FMEA)], # Has individual worksheets within it
       hasText     => 0},

FMEA    # Capture name from first line of a single FMEA worksheet
   => {recognizer => qr/^FMEA Number: (.+)/,
       nameFromRec  => [qw(name)],
       allowedAttributes => 1,
       subsections  => [qw(FailureMode ItemFunction)],
       itemizer     => qr/^ ([:]+): \s+ (\S (:? .*\S)?) \s* $/x,
       # Look for pairs of fields on a single line.
       sharedFields => {'Prepared by Reliability Engr' => ['Date'],
                  'Concurred by Design Engr'     => ['Date'],
                  'FMEA Revision'           => ['Date'],
                  'Drawing Number'    => ['Ref Des']}}
```

Figure 6. Excerpt from an FMEA worksheet structure specification.

Sentence parsing and tagging

Some substructures contain sentences that describe the hierarchical parts structure of the vehicle components or the connections between components. These are selected for parsing and tagging. STAT separates sentence clauses and pulls out syntactical relationships (subject, verb, object, indirect-object) from the clauses. Consulting the Aerospace Ontology, STAT adds tags to the syntactical structure:

- Verbs that indicate connecting relationships – sends, supplies, transfers
- Verbs that indicate part-of or other structural relationships – contains, consists of, comprises

Tags are added as annotations to the Perl structure. This structure is passed in xml format to the HIT software, for semi-automatic model development.

The following long sentence from a FMEA document introduction illustrates the richness of this text. It has been used to provide a challenge test to our extractions.

> 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), a bi-conic adapter which 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.

The sentence describes connectivity, three levels of parts hierarchy, and two acronyms. The goal is to extract the following information:

Toplevel: LAS
Component: Nose cone;
Component: Canard section,
Function: reorient the CM,
Rationale: parachute deployment following an abort;

Component: Three propulsive motors
Component: attitude control motor,
Component: jettison motor,
Component: abort motor
Component: A bi-conic adapter,
Function: provide the structural interface to the CM,
Connection: (biconic adapter to CM, type structural)
Component: Boost protective cover
Acronym: Boost protective cover = BPC
Rationale: sized for ascent heating
Function: protect CM thermal protection system coatings
Acronym: thermal protection system = TPS

Currently, most of this information can be extracted, but there are some errors on the parts hierarchy implied in the sentence. Some of this parts information is also available the in FMEA worksheets.

Improved parsing and tagging

Experience with Orion documents has led to numerous additional extraction improvements.

***Abbreviations*.** Text often mixes abbreviations with spelled-out forms. STAT starts with extensive lists of aerospace and NASA abbreviations. STAT also reads abbreviation tables from individual documents, and it can recognize and learn abbreviations when encountering them in text – e.g., in "These are candidates for Design for Minimum Risk (DFMR.)"

The model generation software uses the acronym information to transform all variations into one standardized identifier. For example, an "abort motor through bulkhead initiator" could be referred to as "AM through bulkhead initiator," "LAS abort motor TBI," or "AM TBI." Any multi-word name that has an officially recognized acronym is replaced by that acronym before a new component or connection is created. "Launch Abort System Abort Motor Through Bulkhead Initiator" would be replaced by "LAS AM TBI." The acronym "LAS" is removed because the AM TBI is a part of the LAS. The standardized identifiers help eliminate duplication in the model. The list of existing model components is examined to determine if a component named "AM TBI" already exists or should be created.

***Tests and requirements*.** Terms denoting problems often turn up as adjectives in the names of tests or other engineering activities or artifacts. For example, "corrosion testing" should not be tagged as an occurrence of a corrosion problem. STAT recognizes such phrases and tags them properly as engineering activities or artifacts.

***Inflected forms*.** Any of several related words such as "acquire," "acquiring," or "acquirable" can denote a concept such as "aquire." Typically, these inflections have been aggregated by wildcard stemming – *acquir**. This is often too broad or too narrow. The wildcard *acqui** tags both *acquisition* (correct) and *acquittal* (wrong), but *acquir** tags neither. STAT solves this problem by using in-depth morphological analysis and a knowledge base of irregularly inflected forms.

Some inflected forms shouldn't be tagged. For example, "blocked" and "blockages" denote obstructions but "block" commonly denotes a section of text in a regulation. By default, STAT tags all inflections with a concept from the ontology. This tagging can be adjusted for individual inflections, and differently for different domains.

***Parsing and styling*.** Problems are often denoted by some desirable concept term that is negated or modified, for example, "not aligned," "arrived too late," or "difficult to complete." Standard keyword tagging fails to distinguish between these occurrences and corresponding positive concept terms ("aligned," "arrived," "complete"). The modifying words are called styles. To assign tags, STAT recognizes and combines the positive terms and the styles. STAT has seven groups of style types: negation, excess, insufficient, bad, difficult, early, and late. The ontology provides up to several dozen modifying terms (words or phrases) for each group.

Combining positive terms and styles is straightforward when styles are adjacent to concepts, for example, “incorrectly aligned.” However, styles and concepts can be separated by intervening text, for example, “Neither pin was aligned correctly.” To correctly tag text, STAT performs a full-natural language parse to determine if concept terms are within the linguistic scope of style words.

Complex Sentences and Scoping. When long sentences contained several clauses, the original simple parser usually correctly extracted no more than the first subject/verb/object triple. Integration of a new parser into STAT has substantially increased the number of correctly parsed sentences.[10] The new statistical parser performs a complete syntactical analysis. It also prunes extraneous information before parsing and identifies the semantic roles of verb complements.

Semi-Automated Model Development and System Visualizations

HIT software has been developed to generate component-connection models from the tagged sentence data structures extracted by STAT. A capability has also been added to automatically lay out visualizations of the models.

HIT Component Extraction Methodology

For each extracted sentence in a document (FMEA or IIRD), HIT searches the Aerospace Ontology to determine if there are ontology concepts that match the sentence parts of speech and that indicate that a component connection is being described. The rules for identifying sentences representing component connections are as follows:

- The subject of the sentence must map to a concept in the Aerospace Ontology that is a kind of component or artifact.
- There must be an indirect object that maps to a concept in the Aerospace Ontology that is a kind of component or artifact.
- The direct object of the sentence must be an entity that maps to an Aerospace Ontology concept for something provided by one component to another (e.g., power or information).
- The sentence verb must map to a concept that denotes the act of transmission (e.g., “send” or “receive”).

An “indirect object” may be identified by prepositional phrases, where the preposition denotes that the object of the phrase is the source or destination of the entity that is identified as the sentence’s object (e.g., “from” or “to”).

Figure 7 shows an example of one such sentence and the generated data structure that satisfies the rules for a component connection. The concept-class names following the word “MATCHES” indicates the concept that satisfies one of the rules for a component connection.

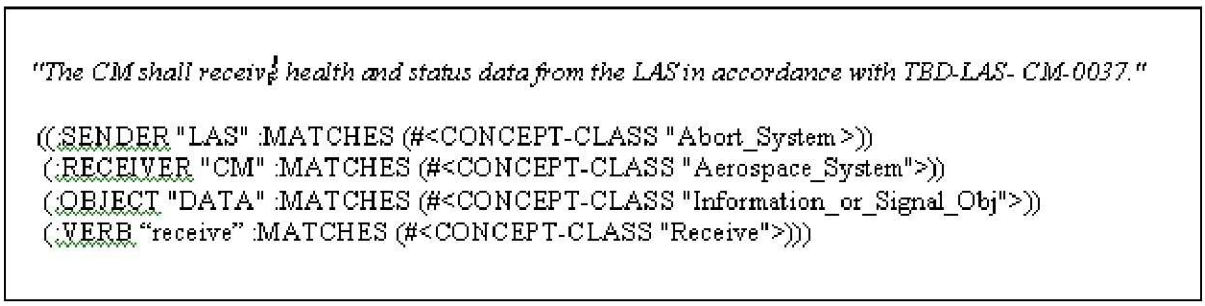


Figure 7: Sentence extracted from the CEV IIRD and the data structure supporting the existence of a connection from the LAS to the CM.

High-Level model extraction from Internal Interface Requirements Documents

The IIRDs provide general information on connections between the LAS, CM, and SM. The top-level CEV model generated from the IIRD for the LAS, CM, and SM is shown in Figure 8. Each connection from the CM to the LAS represents several IIRD requirements. One connection represents power transmission from the CM to the LAS. The second connection represents various types of commands to be transmitted from the CM to the LAS. Despite the inclusion of the SM in the IIRD title, there is no further SM connection information. The fact that the model extraction process shows a lack of connections to the SM from the other two modules could be potentially useful information to analysts evaluating the completeness and correctness of the IIRD.

Documentation and source information in visualizations

The existence of a given model element (i.e., a component or a connection between components) may be supported by more than one sentence in more than one document. As part of model generation, all sentences that justify the creation of a model element are identified. These sentences and the titles of the source documents from which they were extracted are attached to the model data structures. This information can be displayed by a mouse click in the graphical display, over the arrow representing a connection or the icon representing a component.

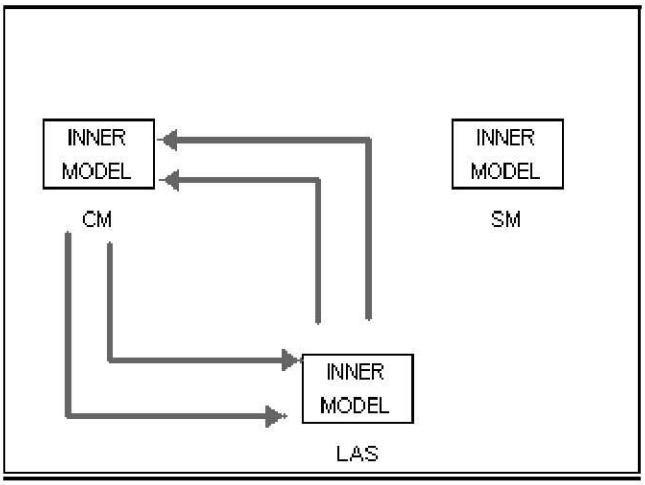


Figure 8: Top-level model generated from an IIRD document.

Figure 9 shows the Documentation display for one of the two connections from the LAS to the CM that are shown in Figure 8. Note that the second IF.CM.LAS requirement is the sentence matched to a component connection in Figure 7. The “type of thing sent” entry is the concept in the Aerospace Ontology that matches the sentence object and that satisfied the rule that a sentence must reference some entity that could be transmitted from one component to another. In this case, the entity “command” matches the Information_or_Signal_Obj concept in the Ontology.

Detailed Component Connection Models Extracted from FMEAs

In Figure 8, the label “INNER MODEL” inside the CEV module icons indicates that each module can be decomposed into a lower level of components and connections. The HIT model generator extracts these lower-level connections from the text of the CEV FMEA document, which provides more detailed information about component connections.

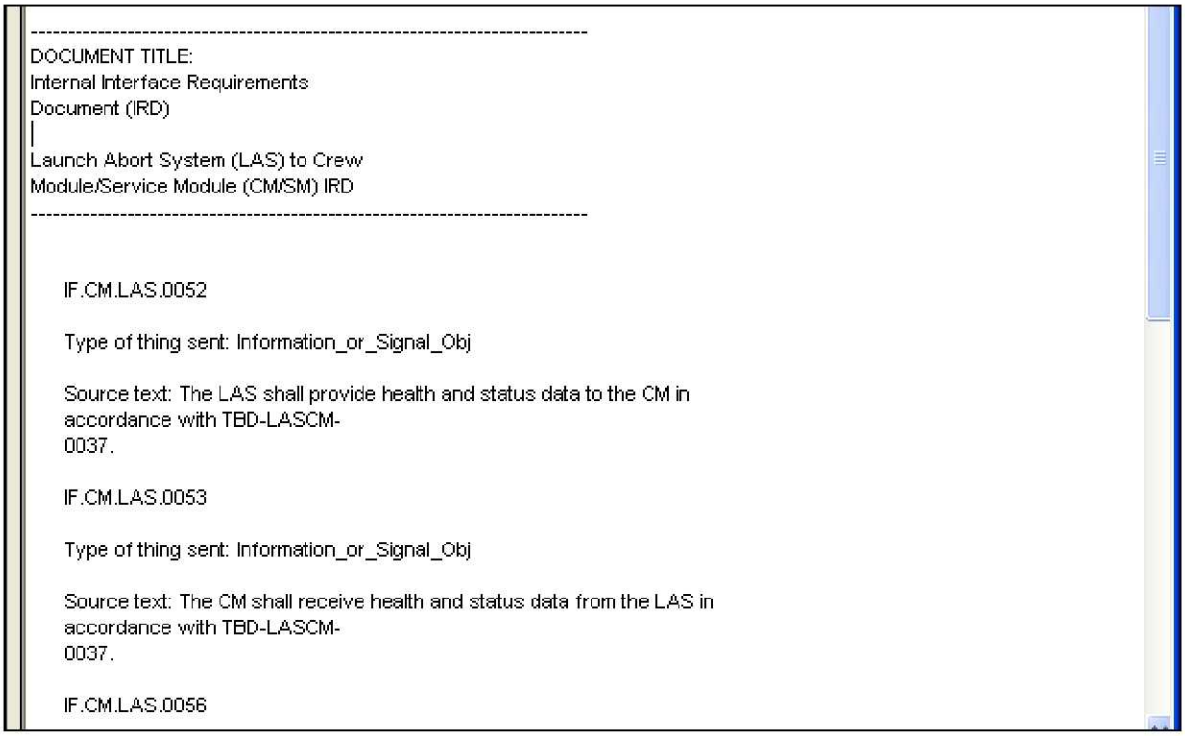


Figure 9: Documentation display for a LAS-CM connection extracted from an IIRD.

A portion of the FMEA worksheet text for the abort motor through bulkhead initiators (TBIs) is shown in Figure 10. The Indenturements section at the top of FMEA worksheet describes the part relationships. The HIT model generator uses this information to assign components to the correct CEV inner model. The FMEA also states the quantity of components of the type covered by the FMEA item. The FMEA Item Function description often describes connections to other components. This connectivity information is used to create the connections between model components. Figure 11 shows the component connection data structure that is generated from the FMEA in Figure 10.

Program: Constellation
Project: Orion
Element: Crew Exploration Vehicle (CEV)
Module: Launch Abort System
System: Launch Abort System (LAS)
Sub-System: Ordnance

Item Name: LAS Abort Motor TBI
Item Part Number: PENDING-2103

Drawing Number: **Ref Des:**
Qty: 2
LCN:
Item Function: The Abort Motor Through Bulkhead Initiators provide initiation energy to the abort motor igniter during an abort.

Figure 10: FMEA stating that the LAS contains two TBIs connected to abort motor igniters.

```
ID          "FMEA-004075"
SENTENCE    "The Abort Motor Through Bulkhead Initiators provide initiation energy to the
            abort motor igniter during an abort."
SENDER      (:WORD "INITIATOR"
                    :CAT-MATCHES (#<OWL::CONCEPT-CLASS "Igniter">)
                    :QUALIFIERS ("AM" "THROUGH" "BULKHEAD")
                    :COMPONENT-DATA (:QTY 2 :ITEM-NAME "AM TBI"
                                            :SUPER-PART #<LAS COMPONENT-10003>))
RECEIVER    (:WORD "IGNITER"
                    :CAT-MATCHES (#<OWL::CONCEPT-CLASS "Igniter">)
                    :QUALIFIERS ("AM") :COMPONENT-DATA
                    (:ITEM-NAME "AM IGNITER"
                                :SUPER-PART #<LAS COMPONENT-10003>))
OBJECT-SENT (:WORD "ENERGY"
                    :CAT-MATCHES (#<OWL::CONCEPT-CLASS "Energy_or_Power">)
                    :QUALIFIERS ("INITIATION"))
CONNECT-VERB (:WORD "PROVIDE"
                    :CAT-MATCHES (#<OWL::CONCEPT-CLASS "Provide">)
              :DIRECTION :FROM-SUBJ)
```

Figure 11: A component connection data structure derived from the FMEA for TBIs in Fig. 10.

Information provided by model visualizations

Figure 12 shows the inner model of the LAS pyrotechnics that was automatically created from several FMEA worksheets. Unlike the connections in the top-level CEV model, many of the connections from the FMEAs represent direct physical connections between components. Even the model connections that do not represent direct physical connections indicate some form of dependency.

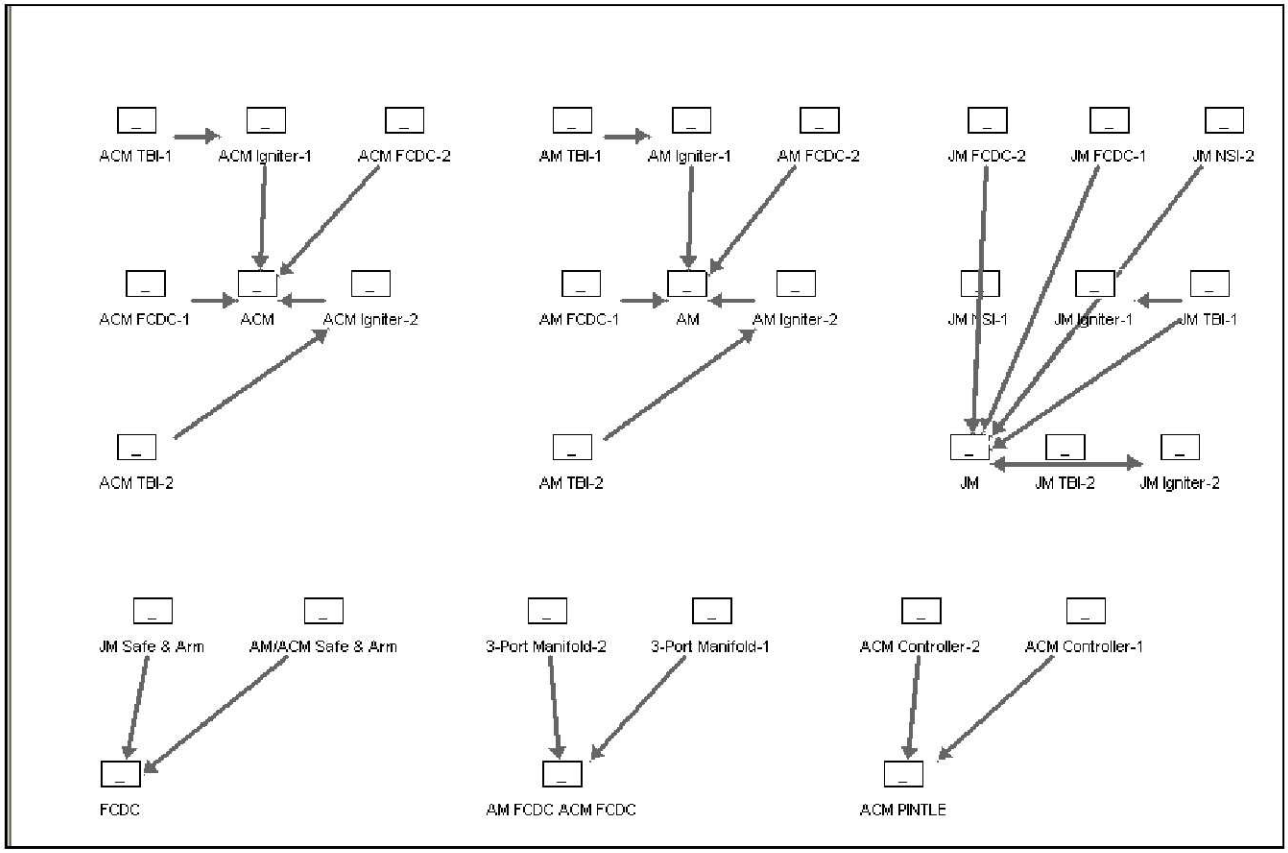


Figure 12: Inner model of the LAS extracted from FMEA document.

The component at the head of each connection arrow depends on the functions of the component at the tail. For example, the model shows components labeled “FCDC” (“flexible contained detonator cord”) connected directly to the ACM and AM. The FCDCs are actually physically connected to the TBIs, which are the components in direct contact with the motor bulkheads in the pyrotechnic system. However, the successful operation of both motors depends on the proper operation of the FCDC cord system. The FCDC system is analogous to “wiring” or “network.”

In Figure 12, the structurally identical networks of components connected to the AM and ACM in the LAS model reflect the identical connectivity for the two motors in the schematic of the pyrotechnic system in Figure 4. Although not all of the connections to the motors in the model represent physical connections in the system schematic, the visualization indicates that the FMEA worksheets account for dependencies among components associated with the two motors.

Comparison of the LAS model with the schematic also shows that there are components in the FMEAs that do not appear in the schematic (e.g., Jettison Motor NASA Standard Initiators labeled “JM NSI”). Likewise, there are components in the schematic not mentioned in the FMEAs. For example, the parts labeled “NSD” in the schematic are not in the FMEAs. They are apparently detonators activated by the pyrotechnic event controllers labeled “PEC.” The visualization makes these discrepancies in the FMEA document much easier for an analyst to find.

References to a component or connection found in a document during model generation are added to the information display for that component or connection. Examination of these collected references reveals that, for some components, there are multiple FMEAs describing different failures modes rather than a single FMEA describing all component failure modes. For example, one FMEA for the abort motor described a “fires prematurely” failure mode and a second described a “fails to fire” mode. Both of these failures could have been included in a single FMEA item. The visualization makes such anomalies in the information easier for SMA analysts to find.

Sorting our multiple components and connections

When an FMEA states a quantity of like components greater than 1, HIT creates the same number of components with the same name but different numerical suffixes, as shown in Figure 12. When a sentence indicates a connection with sets of components, a modeling problem arises. For example, “The Abort Motor Through Bulkhead Initiators provide initiation energy to the abort motor igniter …”

Because there are equal quantities of TBIs and igniters in this case, HIT makes pair-wise connections between TBIs and igniters having the same suffix. In other words, AM TBI-1 is connected to AM Igniter-1 and AM TBI-2 is connected to AM Igniter-2. If the quantities of two sets of connected components differ, cross-strapping is assumed (i.e., each component in one set is connected to every component in the other set).

Aerospace Ontology

The screening rules for model development use Aerospace Ontology classes. The Aerospace Ontology has been enhanced to support these rules for identifying sentences representing component connections and screening out spurious mappings to model structures. To support the screening rule for transmitted things, distinctions between the concepts of Electrical Power and Mechanical Power have been sharpened. Senders can be distinguished from Receivers in the architecture by using the action verb categories Output and Take-In. This determines whether the indirect object or the subject is the Receiver or Sender.

A modeling rule requires components to be descendants of either the Artifact_or_Device category or the Physical_Structure category. The Aerospace Ontology was enhanced to include more domain concepts and mapping words (including acronyms) that are used to describe launches and

components of rockets, pyrotechnic components, and launch abort systems. Terms were also added to describe LAS hazards and problem detection, including types of test and inspection and corrosion factors.

A Channel type and several subtypes were added to the ontology, to support identification of types of carriers of entities moving along connections. This set of new types is shown in Figure 13. Each category name is followed by a list of mapping words and phrases that indicate that category if they are found in the text. These channel types often denote a supporting structure such as a network of wires or pipes. Thus, they can be used to recognize components that may not be enumerated in full in a FMEA because they serve as connection network infrastructure.

Channel (passive transport): channel, pipe, conduit, tube, duct, pathway, course, passage

1. Signal_Carrier: bus, data bus, data channel, network, broadband, fiber optic cable, local area network, LAN, intranet, internet, extranet, world wide web, WWW, communication loop…
2. Wireless_Signal_Carrier: radio wave, radio frequency, S-band, Ka-band, Ku-band, L-band, X-band, AM, FM, radio frequency, RF, ultrasonic, short-wave, wireless, WIFI, Wi fi…
3. Electrical_or_Power_Carrier: optical fiber, wire, bus wire, stub wire, ground wire, line, lead, wiring, conduit, tube, raceway, wireway, busway, wiring duct, circuit, circuitry, cable…
4. Pyro_Carrier: detonating cord, flexible confined detonating cord, FCDC, shielded mild detonating cord, SMDC, fuse, ignition cord, igniter cord
5. Fluid_Carrier: line, fluid line, loop, fluid loop, pipe, pipette, duct, tube, capillary tube, tubing, hose, flex hose, siphon, syphon, wick, culvert, fluid jumper, conduit
6. Domain_Fluid_Carrier: pipeline, water line, hydraulic line, coolant line, gas line, steam line, propellant line, water loop, hydraulic loop, gas loop, steam loop, high pressure line…

Figure 13: Types of channel in the Aerospace Ontology.

An interface can be a connection, carrier, or channel, but it may also be a component that performs an interface type of function. A hierarchy of Interface structure types and subtypes was created, as shown in Figure 14. This information can be used in future model generation to distinguish types of interface components and structures.

1. Information_Interface: information connection, information interface
   - 1.1. Message: message, communication, command, confirmation, indication, notice, notification, e-mail, email
     - 1.1.1. Comment : comment, annotation, callout, remark
     - 1.1.2. Problem_Message: alarm, caution, warning, error indicator, error flag, error message, complaint, hazard indicator
   - 1.2. Human_Interface: user interface, UI, human interface, display
   - 1.3. Software_Interface: software interface, avionics interface, software driver, driver, API
     - 1.3.1. Command_or_Data_Interface: data interface, command interface, remote interface unit, RIU, data port, access point, USB, FireWire, Ethernet, ATA/IDE, SCSI, PCI …
2. Physical_Interface_Component: physical interface, adapter, umbilical, umbilical cable
   - 2.1. Connection: connection, connector, coupler, coupling, cross coupling, chain, strap, cinch, socket, dynatube fitting, key, flange connector
     - 2.1.1. Fastener: fastener, anchor, bolt, eyebolt, screw, self-tapping screw, nut, locknut, safety nut, lug, lug nut, nail, washer, pin, crimp pin, linchpin, rivet, blind rivet …
       - 2.1.1.1. Separation_Fastener: separation bolt, separation nut, disconnect, quick disconnect, release mechanism, quick release, pyrobolt, explosive bolt …
     - 2.1.2. Distributor: distributor, multiplexer, splitter, manifold, 3-way junction, T junction, Y junction, demultiplexer
   - 2.2. Mechanical_Interface; mechanical interface, mechanical connection

2.2.1. Mechanical_Closer: isolator, safe and arm device
2.2.1.1. Plug: plug, cork, stopper, bung
2.2.1.2. Valve: valve, poppet valve, poppet, relief valve, safety valve, glove valve, gate valve, check valve, piston valve, ball valve, needle valve …
2.2.2. Mechanical_Outlet: vent, orifice, nozzle, faucet, hose bib, drain
2.3. Electrical_or_Power_Interface: power interface, power connection, electrical interface, electrical connection
2.3.1. Electrical_or_Power_Connector: plug, fuse, pin, contact, terminal contact, connector, connector plug, interconnect
2.3.2. Electrical_Closer: switch, switchgear, DIP switch
2.3.3. Electrical_Outlet: socket, terminator
2.4. Structural_Interface: structural interface, structural connection
2.4.1. Spacer: stand off, standoff, separator, spacer, edge spacer, shim spacer, spacer block
2.4.2. Structural_Opening: airlock, door, window, porthole, viewport, hatch
2.5. Biological_Interface: biological interface, biological connection

Figure 14: Types of interfaces in the Aerospace Ontology.

Simulation Scripts

CONFIG scripting capabilities were enhanced to support a script for a launch abort scenario after second-stage ignition. The script emulates some of the software operations controlling a sequence of rocket firings and stage separations. A new DELAY option was added to script specifications so that conditional events can be caused to occur after a time delay rather than immediately at the time the condition is present.

A function detection model and a logging specification support detecting whether specific system-wide functions are present during a CONFIG simulation. The presence or absence of a system-wide function may be dependent on the states of multiple system components. The function modeling utility was expanded to include the capability to specify the time when a particular function has succeeded. This was determined to be useful due to the nearly instantaneous nature of some critical functions in Constellation, such as the separation of rocket stages.

This work on simulation scripts is exploratory, leading up to enhancing HIT and CONFIG for an LAS hazard analysis case.

Virtual System Integration Laboratory Simulation Model

An interconnected system structural framework simulation model of the LAS has been developed for the VSIL. This comprises the functional blocks that are involved in the LAS decision logic and execution of a launch abort sequence. VSIL parts include CEV and LAS avionics, pyrotechnic separators for LAS and CM, avionics connections to the pyrotechnics, AM, ACM, JM, thrusters for AM, ACM and JM, valve assembly and control system, igniter assemblies, canard, fairing, spacecraft adapter, ground systems, mission systems, and Constellation Communication Adaptor network node.

Papers and Publications

Jane T. Malin, David R. Throop, Land Fleming and Carroll Thronesbery, Information Extraction for System-Software Safety Analysis, 26th International System Safety Conference, Vancouver, Canada, August 2008.

This paper 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.

Calendar Year 2009 – 2010 Plan and Milestones

April 2009	Extract LAS Pyrotechnic System Hazard Reports and FMEA failure modes information.
May 2009	Complete Orion LAS model extraction case and evaluation of visualization.
July 2009	Complete Orion LAS hazard path analysis, with test case definition.
Aug 2009	Complete analysis evaluations.
Aug 2009	Complete preliminary methods document.
Sep 2009	Complete first version of enhanced and integrated tool suite.
Sep 2009	Software Assurance Symposium Presentation.
Oct 2009	Submit conference paper on integrated tool suite.
Nov 2009	Submit new technology disclosure on integrated tool suite.
Dec 2009	Complete annual report.
Jan 2010	Manage and compare model versions when source documents change.
Feb 2010	Complete model extraction for new Orion case, including model versions and changes.
Mar 2010	Complete new Orion hazard analysis case, comparing changes due to updates.
Jun 2010	Complete enhanced version of integrated tool suite.
July 2010	Complete evaluations and final methods document.
Aug 2010	Complete project analysis results files or reports.
Sep 2010	Software Assurance Symposium Presentation.
Sep 2010	Complete project Final Report.
Sep 2010	Complete project software source files and documentation and deliver on CDs.

January – September 2009 Technical Plans

Work in 2009 will continue to focus on enhancements to STAT, HIT, and the Aerospace Ontology for Orion and LAS model extraction and development. The SMA project participant has requested that extractions be made from LAS Pyrotechnic System Hazard reports and compared to FMEA extractions, to “close the loop” between FMEAs and hazard analyses. This work will be added to model extraction tasks. Path analysis is dependent on extraction of functions and failures from FMEA documents. Extraction of these types of information is in progress.

Progress has been made on generating architecture models from text that implies component connections, and parsing capabilities are being substantially enhanced. This progress will enable extraction improvements in the following areas:

- ***Extracting and Structuring Information on Component Failures and Functions.*** Parsing progress will enable extraction of additional information on component functions and failures, for use in hazard path analysis. Failure mode descriptions in the FMEAs can describe input failures and internal functional failures of components. These can describe inputs that disable functions and additional component functions that may fail. This is the type of failure and vulnerability information needed for hazard path analysis.
- ***Eliminating Spurious Components and Connections.*** As can be seen in Figure 12, the model extraction methods developed to date do not guarantee that all the components and connections created are valid. For example, the connections of a “JM Safe&Arm” and an “AM/ACM Safe&Arm” to a common “FCDC” is not correct. FCDC cords are themselves

connectors used throughout the pyrotechnic system, and the two Safe&Arm devices are not connected to the same FCDC segments. Also, the components labeled "ACM FCDC-1," "AM FCDC-1," etc. are actually the connectors referred to by the sentence concerning Safe&Arm devices. One objective of future work is to address such problems. The new Carrier concepts in the Aerospace Ontology (Figure 13) can be used to help identify the FCDC connectors.

- ***Extracting Connections from More Complex Sentences.*** Some sentences refer to connections generally while using singular forms. They actually describe multiple connections between more than two components. One FMEA example is the sentence:

  "The 3-port manifold distributes energy from the Safe&Arm to the AM FCDC and ACM FCDC for abort."

  This sentence describes the actual physical connections for a manifold, one of the Safe&Arm components, and the FCDCs of the two motors. There are actually two sets of components, each consisting of a manifold, a Safe&Arm, and three segments of FCDC. Improvements to STAT parsing should make sentences such as this useable for component-connection extraction.

HIT path analysis CONFIG simulation software will be enhanced to use the extracted LAS models to identify hazardous configurations, scenarios, and test cases. New HIT capabilities are also planned to support multiple model versions. This will support reuse and analysis to evaluate requirements and design changes. A first version of the integrated tool suite will be completed, and a preliminary methods document for its use will be written. The project has not begun extracting operational and failure scenarios. FMEA effects information is a potential source for parts of failure scenarios.

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.

During 2009, Triakis will continue developing the VSIL by coding the behavior of simulator system elements to match the requirements given. This will be done to the extent necessary to facilitate testing of hazards identified by the research team. Triakis will also explore the viability of using the outputs from STAT or HIT, to automatically generate tests that may be used to verify identified hazards in the VSIL.

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]Malin, J. T., Throop, D. R., Millward, C., Schwarz, H. A., Gomez, F. and Thronesbery, C. "Linguistic Text Mining for Problem Reports," submitted to IEEE SMC Conference.