JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 3, November 2006

Engineering Spacecraft Mission Software using a Model-Based and Safety-Driven Design Methodology

Kathryn Anne Weiss*
Jet Propulsion Laboratory, Pasadena, CA, 91109

Nicolas Dulac,† Stephanie Chiesi,‡ Mirna Daouk,† David Zipkin,§ and Nancy Leveson*¶
Massachusetts Institute of Technology, Cambridge, MA, 02139

This paper outlines an integrated approach to system, software, and safety engineering for today's complex, safety-critical systems. Intent Specifications, component-based systems engineering, and a new hazard analysis technique based on the Systems Theoretic Accident Modeling and Process (STAMP) are all combined in a seamless, safety-driven design methodology. This integrated approach to system software development is demonstrated through an example application of the techniques on a low-Earth orbiting satellite. The results of performing the system modeling, hazard analyses, and model simulations of the spacecraft as a whole are also presented. Finally, the approach is compared to other, current model-based systems engineering and hazard analysis techniques.

Nomenclature

SpecTRM	Specifications Tool and Requirements Methodology
SpecTRM-RL	SpecTRM-Requirements Language
SpecTRM-GSC	SpecTRM-Generic Spacecraft Component
STAMP	Systems Theoretic Accident Modeling and Processes
STPA	STAMP-Based Hazard Analysis

I. Introduction

THE design of system modeling, analysis, and visualization theories and tools to assist in the design and operation of safer systems with greater capability is becoming increasingly important as systems become more complex and the goals of these systems more far-reaching. The main objectives of developing such theories and tools involves providing ways to stretch the limits of complexity and intellectual manageability of modern, engineered systems with reasonable resources, with confidence in their expected behavior, and with particular emphasis on safety and mission success. To accomplish these goals, a systems approach to engineering is needed that includes building technical foundations and knowledge and integrating these with the organizational, political, and cultural aspects of system construction and operation.

Received 18 April 2006; revision received 28 August 2006; accepted for publication 22 September 2006.

* Flight Software Engineer, Flight Software and Data Systems Section, 4800 Oak Grove Drive M/S 301-225, AIAA Member.
† Ph.D. Candidate, Department of Aeronautics and Astronautics, 77 Mass. Ave. #33-407, AIAA Student Member.
‡ Masters Student, Department of Aeronautics and Astronautics, 77 Mass. Ave. #33-407, AIAA Student Member.
§ Master's Student, Engineering Systems Division, 77 Mass. Ave. #33-407, AIAA Student Member.
¶ Professor, Department of Aeronautics and Astronautics, 77 Mass. Ave. #33-336, AIAA Member.

This article presents an overview of how to apply an integrated, safety-driven design methodology to create safer spacecraft mission software, using an example of a low-Earth orbiting, imaging satellite. This integrated design methodology is comprised of a systems engineering development environment that supports the use of intent specifications; formal analysis of executable specifications; a new hazard analysis technique based on the STAMP accident model; and iterative, component-based system development. Section II presents the approach including a brief background of the techniques employed—intent specifications, component-based systems engineering, a new model of accident causation (STAMP) based on systems theory, and a new hazard analysis technique based on STAMP called STPA, as well as how they combine to form an integrated approach to safety-driven design. Section III contains a description of the example used to illustrate the integrated approach as well as the models generated for the example. Sections IV and V describe the results of the design process and an analysis of these results respectively. The final section outlines conclusions that can be drawn from this experience, the limitations of this demonstration, and possible future work to improve upon the methodologies.

II. The Approach

The approach described in this paper focuses on developing techniques to support the building and operation of today's complex systems, many of which include large amounts of software, have humans in the loop and are safety- and/or mission-critical. Consequently, an integrated approach to systems development and operation must take into consideration the special needs of software-intensive systems, human factors, and safety analyses. The integration of these engineering disciplines into a design for safety or a safety-driven design approach includes three main concepts: (1) intent specifications, (2) Component-Based Systems Engineering, and (3) STAMP and STPA. These topics are described in the following subsections. The final subsection describes how these processes are used in parallel to provide a framework for designing software-intensive systems while considering safety from the beginning of development.

A. Intent Specifications and SpecTRM-RL

Specifications are critical in complex system development. Intent specifications provide a framework for organizing system development and operational information in a hierarchical structure that models various views of the system through different abstractions at each level of the hierarchy.[1] Each specification is structured as a set of models that describe the system from various viewpoints; complete traceability is provided between all the models. This hierarchical structure was designed to facilitate (1) system-level requirements and design constraints tracing down into detailed design and implementation, (2) assurance of various system properties (such as safety) in the initial design and implementation, and (3) cost reduction for implementing changes and reanalysis. This approach was also based on research on how to enhance human problem solving and therefore aids engineers in the processing and use of specifications to perform system design and evolution activities.

Figure 1 depicts the seven levels of an intent specification as well as the types of information recorded at these levels.

It is important to note that levels do not represent refinement, as in other, more common hierarchical structures, but instead the levels represent completely different models that characterize various views of the system. In other words, each model or level presents a complete view of the system, but from a different perspective. This type of abstraction is termed a "means-end" hierarchy. In a means-end abstraction, each level represents a different model of the same system. At any point in the hierarchy, the information at one level acts as the goals (the ends) with respect to the model at the next lower level (the means). Thus, in a means-ends abstraction, the current level specifies what, the level below how, and the level above why. A change of level involves both a shift in concepts and in the representation structure, as well as a change in the information suitable to characterize the state of the function or operation at the various levels. However, the models within each level may be described in terms of a different set of attributes or language. Refinement and decomposition occurs within each level of the specification, rather than between levels.

1. Intent Specification Levels

The top level (Level 0) provides a project management view as well as insight into the relationship between the plans and project development. Level 1 of an intent specification is the customer view and assists system engineers and customers in agreeing on what should be built and whether that has been accomplished. It includes system goals,

	Environment	Operator	System and Components	V&V
Level 0	Project management plans, status information, safety plans, etc.			
Level 1 System Purpose	Assumptions Constraints	Responsibilities Requirements I/F Requirements	System Goals, High-level Requirements, Design Constraints, Limitations	Hazard Analysis
Level 2 System Principles	External Interfaces	Task Analyses Task Allocation Controls, displays	Logic Principles, Control Laws, Functional Decomposition and Allocation	Validation Plans and Results
Level 3 Blackbox Models	Environment Models	Operator Task and HCI Models	Blackbox Functional Models, Interface Specs	Analysis Plans and Results
Level 4 Design Rep.		HCI Design	Software and Hardware Design Specs	Test Plans and Results
Level 5 Physical Rep.		GUI and Physical Controls Designs	Software Code, Hardware Assembly Instructions	Test Plans and Results
Level 6 Operations	Audit Procedures	Operator Manuals Maintainance Training Materials	Error Reports, Change Requests, etc.	Performance Montitoring and Audits

Fig. 1 Intent specification hierarchy.

high-level requirements, design constraints, hazards, environmental assumptions, and system limitations. The second level, System Design Principles, is the system engineering level and allows engineers to reason about the system in terms of the physical principles and laws upon which the system design is based.

The third, or Blackbox Behavior level, enhances reasoning about the logical design of the system as a whole, its interactions with elements in its environment and the functional state of the system. This level acts as an unambiguous, yet implementation independent model of the system that can assist various types of domain engineers in communicating and reviewing blackbox behavioral requirements and reasoning about the combined behavior of individual components through informal review, formal analysis and simulation. The language used at this level, SpecTRM-RL, has a formal foundation so it can be executed and subjected to formal analysis while still being readable with minimal training or expertise in discrete math.

Level 3 of the intent specifications contains a formal, blackbox model of the component's externally visible behavior. The formal models, which are based on state machines, are specified using a language called SpecTRM-RL that was designed with reviewability and ease of learning as goals. Experience in using the language on industrial projects shows that engineers can learn to read SpecTRM-RL models with about ten to 15 minutes of training. Different visualizations of the underlying state machine allow users to facilitate the creation of a mental model of the system's functioning.

The next two levels (Level 4 and 5) provide the information necessary to reason about individual component design and implementation issues. Finally, the sixth level provides a view of the operational system. Each level is mapped to the levels above and below it. These mappings provide the relational information that allows reasoning across the hierarchical levels and tracing from high-level requirements down to implementation and vice versa.

Intent information represents the design rationale upon which the specification is based. This design rationale is integrated directly into the specification. Each level contains information about underlying assumptions upon which the design and validation is based. Assumptions are especially important in operational safety analyses. When conditions change such that the assumptions are no longer true, then a new safety analysis should be triggered. These assumptions may be included in a safety analysis document (or at least should be), but are not usually traced to the parts of the implementation they affect. Thus even if the system safety engineer knows that a safety analysis assumption has changed (e.g., the pacemakers are now being used on children rather than the adults for which the device was originally designed and validated), it is a very difficult and resource-intensive process to figure out

which parts of the design used that assumption. Interface specifications and specification of important aspects of environmental components are also integrated into the intent specification, as are human factors and human interface design. The separation of human-automation interface design from the main system and component design can lead to serious deficiencies in each. Finally, each level of the intent specification includes a specification of the requirements and results of verification and validation activities of the information at that specification level.

2. Using Formal, Blackbox Specifications

SpecTRM models are both formally analyzable (for such properties as robustness, consistency and completeness) and they are executable. Individual models can be executed in isolation and multiple models can be executed in an environment in a system simulation environment. SpecTRM currently provides two analyses that can be performed on the individual intent specifications, completeness and determinism. Completeness and Determinism are two criteria desirable for safe software systems. A software system is incomplete if the system or software behavior is not specified precisely enough because the required behavior for some events or conditions is omitted or is ambiguous (is subject to more than one interpretation). Mathematically, from a state machine perspective, this means that for any given system state and set of inputs, there must be a transition out of that state. Determinism, on the other hand, refers to the property of the software system that for any given system state and set of inputs, there should be only one way to transition out of each state. These analyses allow the system engineers to eliminate all inconsistencies and incompleteness before the simulation is run. The Level 3 models can be checked automatically for these properties.

There are many benefits to using executable specifications. First, simulations allow system developers to observe the results of interactions between components and the functionality of the subsystem specification and model. Testing blackbox behavior is especially important at this stage in the development lifecycle, because errors in the requirements specifications and/or the blackbox model can be uncovered before any code has been implemented.

Second, after the spacecraft has been created and deployed, changes will need to be made to the on-board software. Code maintenance comprises nearly 70% of the software lifecycle and changes to the software can be costly.[2] An executable state machine provides the software maintainers with the ability to incorporate changes to the code from the formal requirements specification and to simulate the effects those changes will have on the rest of the system, again before any code has been implemented.

Third, executable blackbox models can assist in performing trade-off analyses. Engineers can simulate alternative design strategies and determine which approach is most suitable given the constraints and requirements of the system. Having a formal, executable, blackbox model of the system that is easily read with minimal training provides engineers with the variety of benefits that aid in the proper implementation of a development approach. The following section describes our methodology for reusing intent specifications between various missions by creating generic models and then tailoring those models to the particular characteristics of the mission.

B. Component-Based Systems Engineering and SpecTRM-GSCs

Reuse is critical in decreasing the cost of spacecraft development. However, when reuse is implemented incorrectly, its effectiveness is decreased and in some cases catastrophic accidents occur. The Mars Climate Orbiter, Ariane 5 and SOHO mishaps are all examples of the possible results of improperly implemented reuse. Examining the Mars Climate Orbiter (MCO) loss is instructive. The loss of the MCO involved minor changes to software that was being reused from the Mars Global Surveyor (MGS) spacecraft.[3] According to the developers (but surprisingly absent from the accident report), the original software contained a conversion from imperial to metric units, but that conversion was not documented and was inadvertently omitted when a new thruster equation had to be used because MCO had a different size Reaction Control System (RCS) thruster. "... the 4.5 conversion factor, although correctly included in the MGS equation by the previous development team, was not immediately identifiable by inspection (being buried in the equation) or commented in the code in an obvious way that the MCO team recognized it".[3]

The most problematic reuse has been attempted at the code level, but reuse may be more effective and safe by going back to an earlier development phase and beginning the reuse from that phase (i.e., reuse the work performed up to that phase). Other approaches to software reuse involve reusing commonality-based, or product line, core assets in which engineering artifacts are developed with planned variation points across a family of systems with significant engineering similarities.[4] While very successful in the software engineering realm, software commonality product line principles are just now beginning to become infused into the realm of safety-critical, real-time, embedded applications

such as the spacecraft industry.[5] Because coding is such a small part of the software engineering process, particularly for real-time embedded control software, and coding for these applications can even be partially automated from requirements specifications, the cost of repeating the coding step is insignificant compared to the potential cost of revalidating the reused code. In addition, safely changing code is easier when the changes are made at an earlier development phase.

Furthermore, accidents involving computers are usually the result of flaws in the software requirements, not coding errors.[2] Consequently, applying systems engineering principles during these early development stages, aids engineers in recognizing software interconnections at every stage of the lifecycle, including requirements specification. Thorough documentation, trade-off analyses, testing and the recognition that each software component is a part of the system-as-a-whole increases the quality of requirements specifications by uncovering problems early in the lifecycle and thereby decreasing the cost of correcting these mistakes. Using SpecTRM-RL or another executable specification language allows requirements specifications to be executed and analyzed before any code is ever written or any hardware implementation is completed.

The first step in creating SpecTRM-GSCs (Generic Specification Components) is to decompose the system into manageable chunks. Depending on the properties of the system, this decomposition may be a functional, physical or logical. Functional decomposition is a natural approach for spacecraft, which are composed of components that are grouped into subsystems based on the functionality they provide to the system-as-a-whole. Figure 2 illustrates an example subsystem-based decomposition of a typical spacecraft.

After system decomposition, generic models of the various components are built and assembled to form subsystems. The subsystems are then, in turn, assembled to form the system as a whole. These models are developed as intent specifications with the formal, executable portion at Level 3.

This methodology decreases the time it takes to develop a new spacecraft because it does not start from scratch, one of the main advantages of using a component-based development approach. Because spacecraft engineers develop the specification components, they are specific to the domain of spacecraft engineering, which aids engineers on other projects to easily incorporate these components into their projects. In addition, the entire process of developing the components and subsystems of the spacecraft with the principles of systems engineering and the assembly of the subsystems and controller is reused. Details specific to the spacecraft and its mission can be easily added to the generic components, making the design suitable for the wide range of spacecraft applications.

The third component of the methodology is a new approach to safety-driven design known as STAMP-Based Hazard Analysis (STPA), which provides the safety analysis necessary to support each component model. The

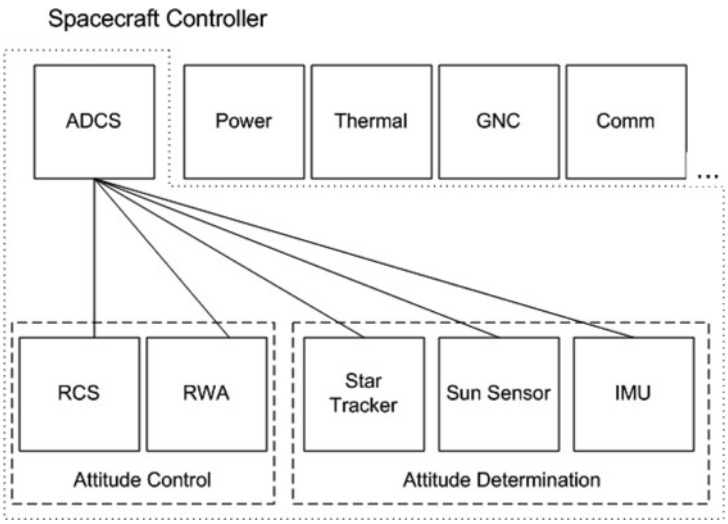

Fig. 2 Example spacecraft decomposition.

hazard analysis parallels the systems engineering efforts and is integrated into each model. STPA is based on a new model of accident causation called STAMP.

C. STAMP and STPA

STAMP, which stands for Systems Theoretic Accident Modeling and Process, is an accident causality model in which accidents are conceived as resulting not from component failures, but from inadequate control or enforcement of safety-related constraints on the design, development and operation of the system.[6] Instead of viewing accidents as the result of an initiating (root cause) event in a series of events leading to a loss, accidents are viewed as resulting from interactions among components that result in a violation of the system safety constraints. The most basic concept in STAMP is not an event, but a constraint. Safety is viewed as a control problem: accidents occur when component failures, external disturbances, and/or dysfunctional interactions among system components are not adequately handled. The control processes that enforce these constraints must limit system behavior to the safe changes and adaptations implied by the constraints.

While events reflect the effects of component failures, dysfunctional interactions and inadequate enforcement of safety constraints, the inadequate control itself is only indirectly reflected by the events—the events are the result of the inadequate control. Therefore, the control structure itself must be examined to determine why the controls were inadequate in maintaining the constraints on safe behavior and why the events occurred—for example, why the hot air gases were not controlled by the O-rings in the Challenger field joints, why the designers arrived at an unsafe design, and why management decisions were made to launch despite warnings that it might not be safe to do so. This control structure includes both the technical and social aspects of the system. An example of a safety control structure is depicted in Fig. 3.

Preventing accidents requires a control structure design that encompasses the entire socio-technical system and will enforce the necessary constraints on system development and operations. Systems evolve over time in order to accomplish changing objectives and to adapt to environmental pressures and disturbances. Often times, accidents result from a migration of the system toward an unsafe or unstable state where small deviations can cascade into catastrophes.[7] In STAMP, systems are viewed as interrelated components that are kept in a state of dynamic equilibrium by feedback loops of information and control. A system is not treated as a static design, but as a dynamic process that is continually adapting to achieve its ends and to react to changes in itself and its environment. The original design must not only enforce appropriate constraints on behavior to ensure safe operation, but it must continue to operate safely as changes and adaptations occur over time.

Furthermore, any controller—human or automated—must contain a model of the system being controlled. Figure 4 shows a typical control loop where an automated controller is supervised by a human controller, both of which have a model of the controlled process.

The model of the process (the plant, in control theory terminology) at one extreme may contain only one or two variables (such as that required for a simple thermostat) while at the other extreme it may require a complex model with a large number of state variables and transitions (such as that needed for air traffic control). Whether the model is embedded in the control logic of an automated controller or in the mental model of a human controller, it must contain the same type of information: the required relationship among the system variables (the control laws), the current state (the current values of the system variables), and the ways the process can change state. This model is used to determine what control actions are needed, and it is updated through various forms of feedback. When the model does not match the controlled process, accidents can result.

Accidents, particularly system accidents, often result from inconsistencies between the model of the process used by the controllers (both human and automated) and the actual process state: for example, the software does not know that the plane is on the ground and raises the landing gear or the pilot does not identify an object as friendly and shoots a missile at it, or the pilot thinks the aircraft is in mode *A* but the computer has changed the mode to *B* and the pilot issues inappropriate commands for the actual mode.

3. Benefits of STAMP

Unlike traditional accident analysis techniques, STAMP takes into consideration the technical, organizational, and dynamic factors that characterize today's complex systems. STAMP has been used in analyzing accidents such

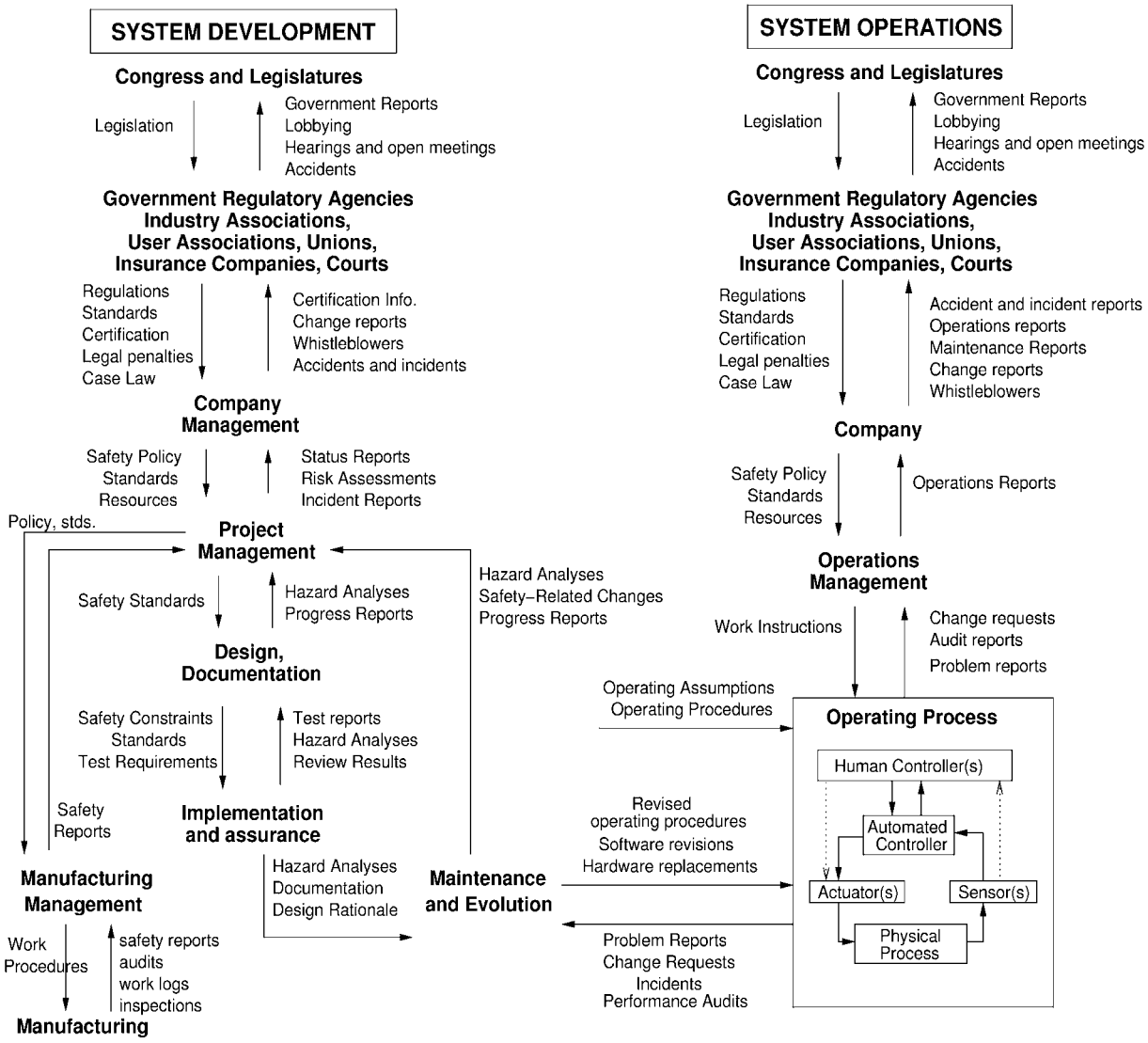

Fig. 3 Example safety control structure.

as a friendly fire Blackhawk helicopter shoot-down over Northern Iraq,[8] a water contamination accident in Ontario, Canada,[9] and several spacecraft losses.

All hazard analysis techniques rest on an underlying model of how accidents are assumed to be caused. Most traditional hazard analysis techniques are based on a model of accident causation as a chain of failure events. In these event-based techniques, hazard analysis consists of identifying the failure events that can lead to a hazard, usually putting them into event chains or trees. Two popular techniques to accomplish this are Fault Tree Analysis (FTA) and Failure Modes and Effects Criticality Analysis (FMECA). Because of their basic dependence on failure events, neither does a good job of handling software or system accidents where the losses stem from dysfunctional interactions among operating components rather than failure of individual components.

The STAMP accident model is useful not only in analyzing accidents that have occurred but in developing methodologies to prevent accidents. Hazard analysis can be thought of as investigating an accident before it occurs. STAMP-Based Hazard Analysis (STPA) goes beyond component failure and is more effective than current techniques in protecting against system accidents, accidents related to the use of software, accidents involving cognitively

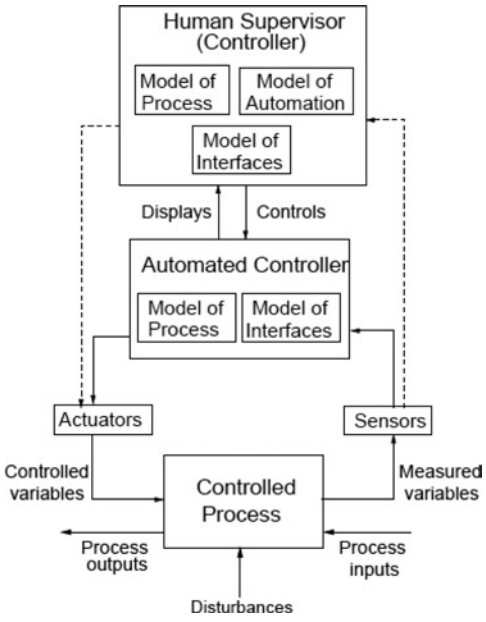

Fig. 4 A typical process control loop.

complex human activities, and accidents related to managerial, organizational, or societal factors. The analysis starts with identifying the constraints required to maintain safety and then goes on to assist in providing the information and documentation necessary for system engineers and system safety engineers to ensure the constraints are enforced in system design, development, manufacturing and operations. STPA has proven very useful to perform a hazard analysis on an existing aircraft collision avoidance system.[10] It was also used as part of a risk analysis performed to identify and prioritize risks associated with the implementation and functioning of NASA's new Independent Technical Authority (ITA).[11]

As previously mentioned, in STAMP, the cause of an accident, instead of being understood in terms of a series of failure events, is viewed as the result of a lack of constraints imposed on system design and operations. The role of the system safety engineer in this model is to identify the design constraints necessary to maintain safety and to ensure that the system design and operation enforces these constraints. This view is closer to the classic system safety approach than the reliability-oriented models often applied today.

4. STAMP-Based Hazard Analysis

STPA (**STAMP A**nalysis) has the same general goals as any hazard analysis: (1) identification of the system hazards and the safety constraints necessary to ensure acceptable risk and (2) accumulation of information about how those constraints could be violated to use for eliminating, reducing, and controlling hazards in the system design and operations.

The first step in STPA is to identify the system hazards. Once the hazards are identified, as in any system safety process, the system-level safety-related requirements and constraints must be identified. STPA utilizes these system-level requirements and constraints to design safety into the system, and as the analysis progresses, new requirements and constraints will be identified and traced to individual components. The next step in the analysis involves defining the basic control structure of the system. After the general (or candidate) control structure has been defined, how the controlled system can get into a hazardous state is defined. In general, a controller can provide four general types of inadequate control:

1) A required control action is not provided.
2) An incorrect or unsafe control action is provided.
3) A potentially correct or adequate control action is provided too late (at the wrong time).
4) A correct control action is stopped too soon.

1. Inadequate Control Actions (enforcement of constraints)
    a. Design of control algorithm (process) does not enforce constraints
    b. Process model are inconsistent, incomplete, or incorrect
        i. Flaw(s) in creation process or updating process (e.g., asynchronous evolution)
        ii. Inadequate or missing feedback
            1. Not provided in system design
            2. Communication flaw
            3. Inadequate sensor operation (incorrect or no information provided)
        iii. Time lags and measurement inaccuracies not accounted for
    c. Inadequate coordination among controllers and decision makers (e.g., boundary and overlap areas)
2. Inadequate Execution of Control Action
    a. Communication
    b. Inadequate "executor" operation
    c. Time Lag

Fig. 5 Classification of control flaws.

Control actions may be required to handle component failures, environmental disturbances, or dysfunctional interactions among the components. Incorrect or unsafe control actions may cause dysfunctional behavior or interactions among components. Hazardous control actions can be identified for each of the other system components. STAMP provides a taxonomy of inadequate control actions to use in understanding why an accident occurred and how it could have been avoided. Figure 5 contains a list of the possible inadequate control actions that may lead to a hazardous state.

The second half of the STAMP taxonomy depicted in Fig. 5 (Inadequate Execution of the Control Algorithm) is used to identify ways in which well-designed control actions can be inadequately executed, thus suggesting additional sub-constraints. A control action can be inadequately executed due to faults in the following building blocks of the loop: (1) the feedback on which the controller bases its actions may be corrupt or may not be present, i.e. there is a problem with the sensors; (2) the control action may not be applied, i.e. there is a problem with the actuators; (3) the controller itself may be defective and (4) the controlled process itself "fails."

D. An Integrated Approach to Design for Safety

The previous subsections describe the three components of the safety-driven methodology described in this document for creating software-intensive, safety-critical systems: intent specifications, component-based systems engineering and STAMP-based hazard analysis. The use of these three techniques in tandem provides a seamless approach to safety-driven design. Systems and safety engineering efforts proceed in parallel and provide information to one another, give and receive feedback for design and work together to ensure that safety is a priority from the beginning of the development lifecycle. Figure 6 presents the typical steps in STPA and how each of those steps in the process interacts with a typical system engineering process throughout system development to create an integrated approach to design for safety.

Although the first steps of the STPA are similar to those performed in other hazard analysis techniques, the later steps either deviate from traditional practice or provide a guiding framework for doing what is traditionally done in an ad-hoc manner. Just as a system design process, the activities involved in the safety analysis process do not occur in a well-defined sequence. The process is highly iterative and should not be seen as a perfectly ordered steps or sequence of events. In fact, safety analyses need to be revisited every time changes occur or as new information is made available.

During the concept development stage of system development (Step 1 in the system safety process), the system engineering group begins development by identifying high-level requirements, goals, concepts and assumptions as well as the first pass at a set of functional requirements. This information is captured using intent specifications and component-based system engineering. Simultaneously, the hazard analysis team begins the safety engineering process with the identification of high-level hazards. The hazard analysis has to be augmented to take into account any

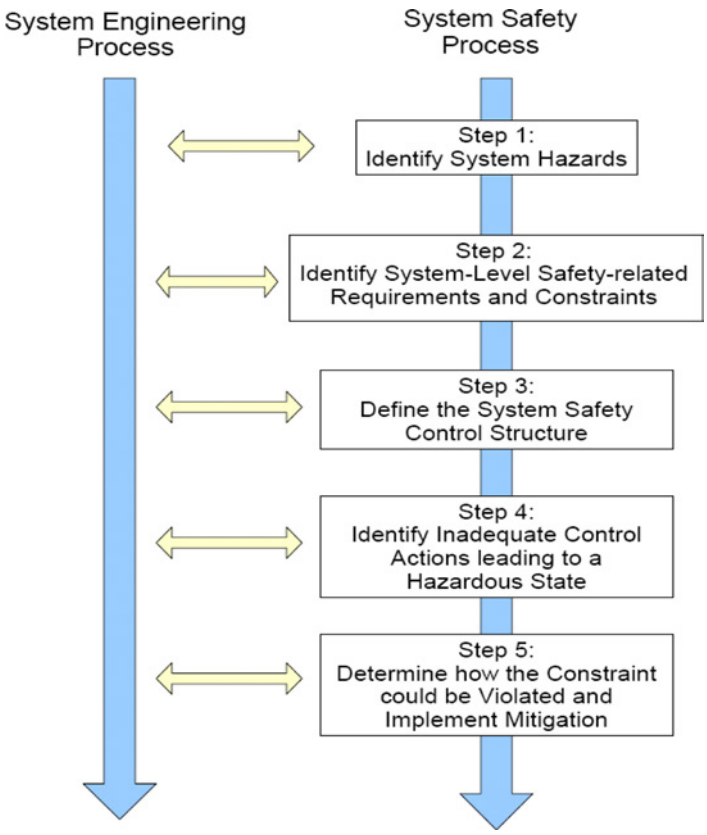

Fig. 6 Integrated approach to design for safety.

new hazards identified. By performing the hazard analysis in parallel with the system design, safety considerations can be accounted for throughout design more easily and at a lower cost. Furthermore, these safety concerns as well as the safety-related requirements and constraints that result from the analysis are also captured in the intent specifications, or SpecTRM-GSCs.[12]

Once the system hazards have been identified, a preliminary set of associated system-level safety requirements and constraints is created, as seen in Step 2 of Fig. 5, which augment and complement the requirements and constraints concurrently developed by the system engineers. It should be noted that the safety requirements and constraints will be refined and augmented as the hazard analysis is refined. The various requirements and constraints are assigned to the appropriate subsystems that comprise the overall spacecraft system.

In Step 3, the various subsystem engineers then discuss possible control actions and possible feedback channels. Selected control actions and feedback channels are integrated in the analysis and the safety analysis team proceeds with the constraint refinement process. The constraints, control actions, and feedback channels are also integrated by the design teams into the various subsystems, at the appropriate level(s) of the Intent Specifications. The constraints and design decisions can be traced back to the hazard analysis through hyperlinks in the specification.

Step 4 involves identifying possible inadequate control actions that are associated with each of the preliminary requirements and constraints. As the system and subsystem intent specifications become more detailed, and the engineers identify the system and subsystem control structures and the various process models for the system components, the models are then evaluated with respect to these inadequate control actions to determine possible control flaws. Additional safety-related requirements and constraints are then added to ensure that these flaws in the control structures and process models do not lead to a hazardous state.

The next section contains an example of the methodology using a low-earth orbiting satellite and its ground support system. The ground system design is augmented by a human-centered approach that includes a focus on human behavior and human–computer interaction.

III. The Example and Models

The functional goal of the example spacecraft described in this paper involves collecting and delivering to the Principal Investigator (PI), or Client, images of selected sites on Earth at specified times and resolutions to monitor deforestation. The pictures are obtained by a low-Earth orbiting spacecraft carrying a camera as payload. Figure 7 depicts the components and boundaries of the spacecraft mission system's satellite, ground facilities and organizational structure. These system elements are modeled in greater detail using the various techniques described in the previous section.

The ground operators use a decision support tool, referred to as the Plan Generator, to generate a plan that specifies (in a time-ordered, constraint-checked list) the location (longitude, latitude), time, and resolution desired for each picture. This plan is developed so as to meet the Client's preferences while taking into account the technical and operational constraints and limitations. The operators then uplink the plan to the spacecraft.

Upon receipt of the plan, the onboard system generates the commands necessary to set up the camera and repositions the spacecraft by using the Reaction Control System (RCS) or by gimbaling the camera. Once the images are taken, the data is processed and briefly stored on board. It is then down-linked to the operators on Earth when the spacecraft comes in view of the Ground Station. Once the receipt confirmation message sent by the ground operators is received by the onboard system, the data can be erased from memory to clear space for future images. The RCS can also be used to point the satellite in the appropriate direction for data transmission and reception.

The spacecraft has an onboard Diagnoser. If a fault occurs on the spacecraft, the system transitions to Safe Mode or Diagnosis Mode. The spacecraft also downlinks housekeeping data to the Ground Station (the amount and type of data transmitted depends on the system's mode of operation). The operators use this data for monitoring and, when a fault is detected, for diagnosis purposes. The operators can manually switch the spacecraft to Safe Mode, Diagnosis Mode, Maintenance Mode or back to normal operations when needed.

The focus in this paper is on the design of the Telecommunications Subsystem, CDHC (Command and Data Handling Computer), Camera Payload, and some aspects of the Ground Segment, which is simplified to include only an Operation Center and a Ground Station with an individual Controller. Moreover, the camera is the only payload package aboard the spacecraft, and housekeeping sensors are limited to those on the two spacecraft Transmitters. The following subsections contain the engineering artifacts and descriptions of those artifacts that were developed as a part of the system and safety engineering processes described in the first part of the paper.

A. Preliminary Hazard Identification

The following high-level hazards were identified:

H.1. Desired images (correct place, correct time, correct resolution) cannot be obtained by client when needed.[#]

H.2. Transmission from spacecraft corrupts transmissions from other spacecraft.

H.3. Spacecraft collides with other spacecraft in Earth orbit.

H.4. Spacecraft re-enters atmosphere in an uncontrolled manner resulting in possible damage, injury, and loss of life.

H.5. Spacecraft otherwise adversely affects other spacecraft.

In the remainder of this analysis, we consider only the first hazard, that of the client not receiving the desired images. Analysis of the other hazards is performed in a similar manner.

B. Identification of High-Level Constraints

Based on the high-level system design and the hazard that the client cannot obtain the desired images (H.1), the following system-level constraints can be derived, as shown in Fig. 8.

[#] A very general definition of safety is used here that includes loss of mission as a hazard.

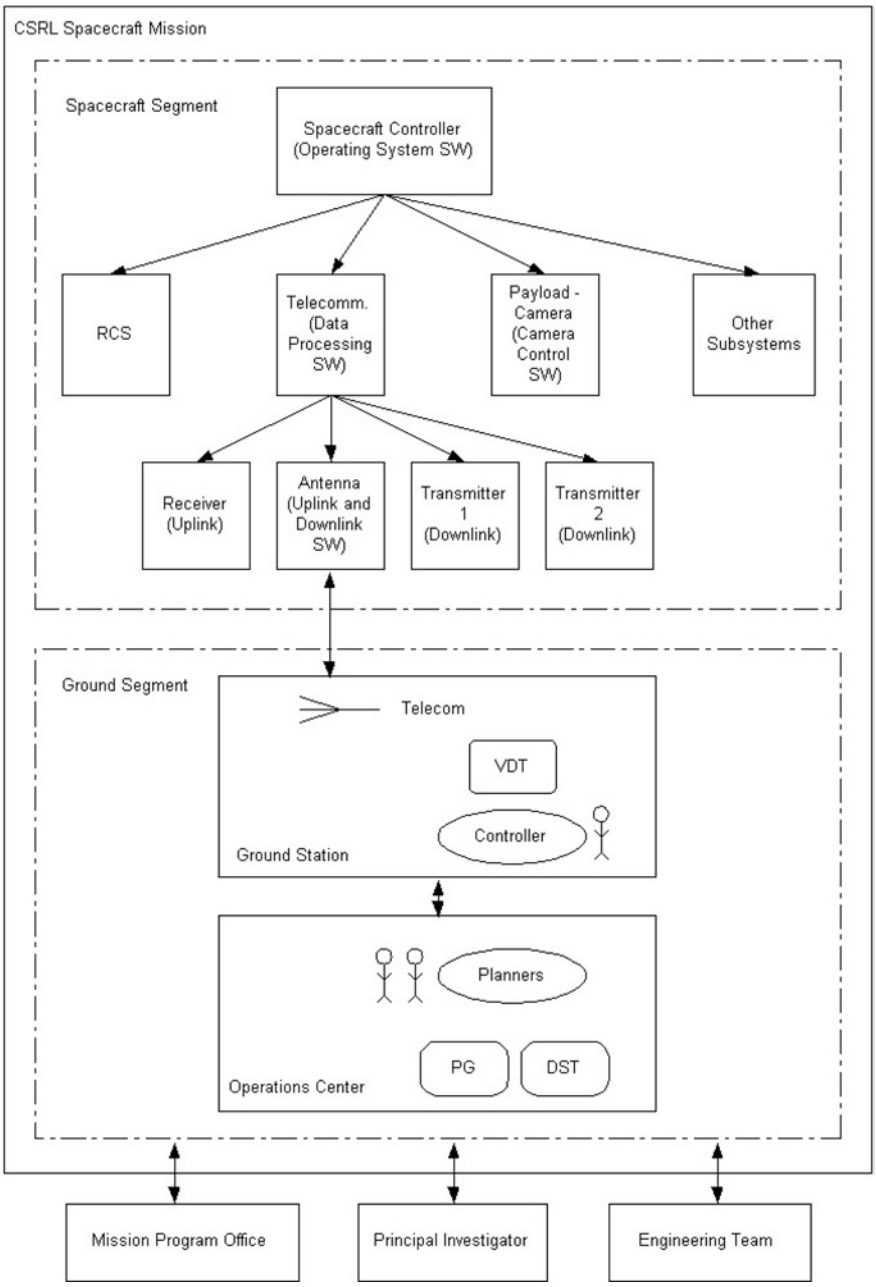

Fig. 7 System components and boundaries.

C. Define the System Control Structure

Once the system level hazards and constraints have been identified, the next step is to define the hierarchical control structure that will be used to enforce the constraints. Each component of the system control structure plays a role in enforcing the system-level constraints. However, enforcing system-level constraints usually requires many system components to perform certain specific control actions. The problem boils down to allocating appropriate responsibilities for each system component that will together result in enforcement of the system safety constraint

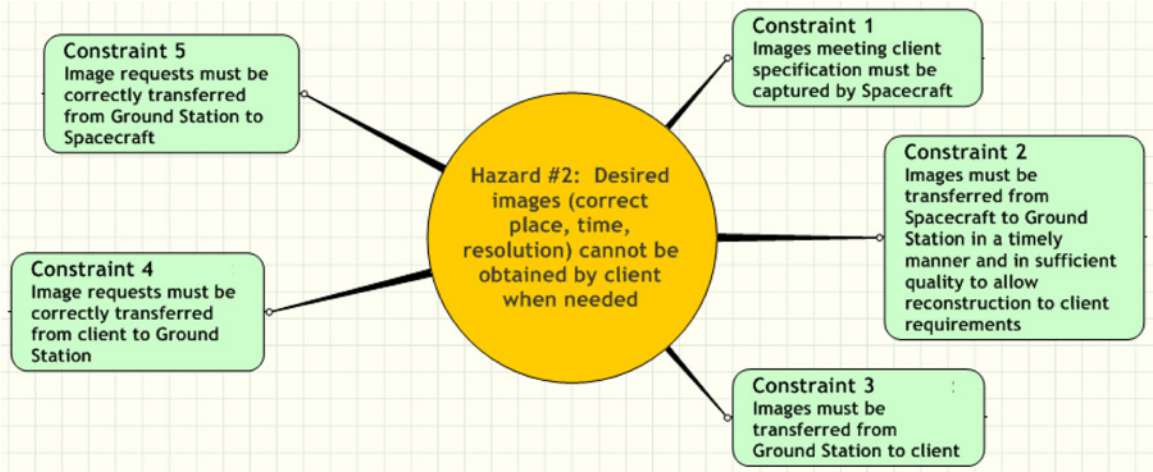

Fig. 8 Selected hazard and system level constraints.

and then analyzing how and why each component might not satisfy its responsibilities. The information from the analysis can be used to design the system to prevent or reduce the likelihood of inadequate hazard control.

Figure 9 provides an overview of the control structure of the system. According to the chosen control structure, the Spacecraft (Sc) communicates only with the Ground Station (Gs), exchanging scientific data, telemetry, and control commands. The Ground Station also communicates with the Central Location Spacecraft Operations Center (Cl).

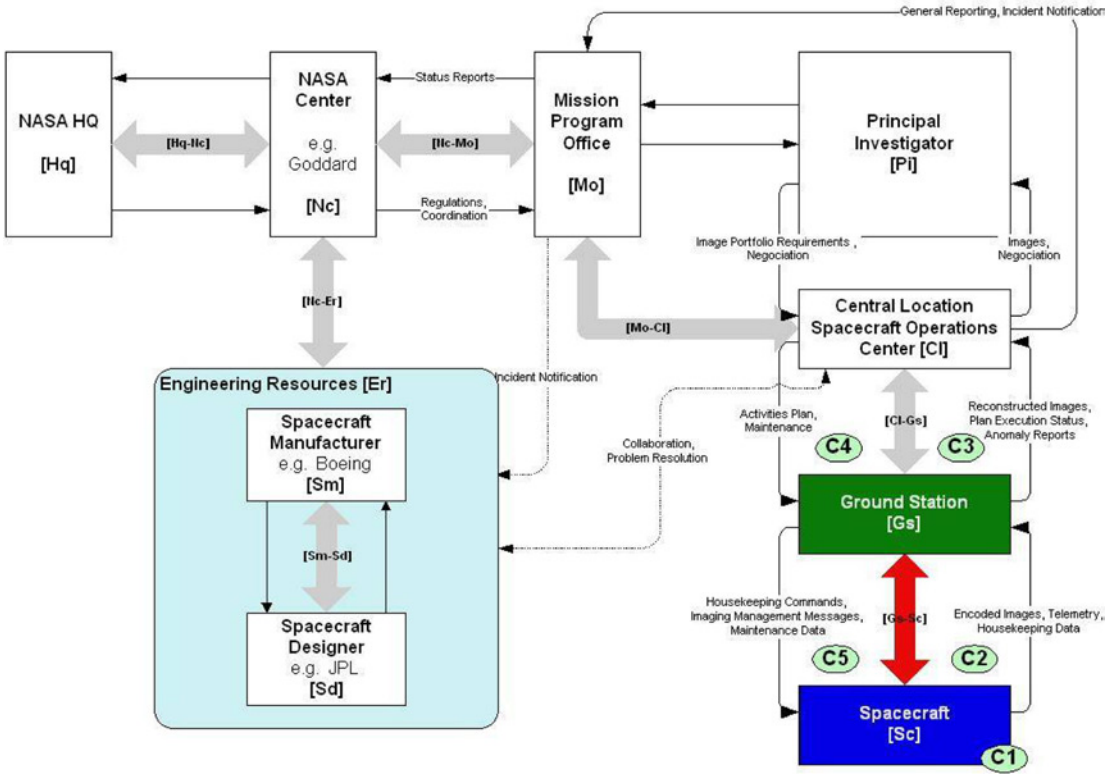

Fig. 9 Mission system control structure.

The Central location receives image requests from the Principal Investigator, who does not enforce constraints on the system, but has much influence on the spacecraft mission design and assignments. The Central Location operates within guidelines from the Mission Office (Mo), which in turn, operates according to guidelines from the specific NASA center coordinating the mission. The NASA center (Nc) reports to NASA HQ (Hq) and operates within the Agency's regulations. The Engineering Resources required to maintain, update and fix the Spacecraft and Ground Station operate according to the NASA Center guidelines and are activated upon request from the Mission Office. The color-coding used in some parts of the control structure matches the constraint refinement scheme provided in the next section.

D. Level 1 and 2 Intent Specification Development

This section contains a description of the intent specification corresponding to the following portions of the spacecraft described above: High-Level Spacecraft Mission and CDHC, Ground Segment, Telecommunication Subsystem (TeleSub), Transmitter, Receiver and Antenna. Figure 10 contains the intent specification hierarchy picture described earlier in this paper. The portions of the entire intent specifications (Level 1–3) that are the focus of this paper are in red text and enclosed in curly braces. For example, the Spacecraft Mission was modeled at all three levels (1–3) while the ground segment was only modeled at Levels 1 and 2.

Using the SpecTRM-GSC approach to spacecraft decomposition and specification, a component-based intent specification is produced for the entire spacecraft segment. At the highest level, the Spacecraft Mission and CDHC document describe the project as a whole in addition to the detail of the onboard spacecraft control. At the Subsystem-Level, the Telecommunication Subsystem (TeleSub) was modeled. Finally, at the component-level, Levels 1 and 2 of Transmitters, a Receiver and an Antenna were defined. These component-level specifications interact with their parent Subsystem, the TeleSub. The TeleSub interacts with the CDHC.

	Environment	Operator	System and Components	V&V
Level 0: Program Management	Project Management Plans, Status Information, Safety Plans			
Level 1: System Purpose	{Spacecraft Mission and CDHC} {Transmitter}	{Ground Segment} {Receiver}	{TeleSub} {Antenna}	
Level 2: System Design Principles	{Spacecraft Mission and CDHC} {Transmitter}	{Ground Segment} {Receiver}	{TeleSub} {Antenna}	
Level 3: Blackbox Models	{Spacecraft Mission and CDHC}		{TeleSub}	
Level 4: Design Representation		HCI Design	Software and Hardware Design Specs	Analysis Plans and Results
Level 5: Physical Representation		GUI and Physical Controls Design	Software Code, Hardware Assembly Instructions	Test Plans and Results
Level 6: Operations	Audit Procedures	Operating Manuals Maintenance Training Materials	Error Reports, Change Requests, etc.	Performance Monitoring and Audits

Fig. 10 Spacecraft intent specification overview.

Environmental Assumptions and Constraints

[EA.1] The CDHC is operating within the low-Earth orbiting satellite.
Assumption: This intent specification describes the CDHC that will operate within this particular spacecraft.

[EC.1] There is only one Ground Station for spacecraft communications. The Ground Station is located at Goddard Space Flight Center in Greenbelt, Maryland, USA. [L.1] [DP.1.3] [TeleSub EC.2] [Transmitter EA.3] [Antenna EA.3] [Receiver EA.3] [Ground L.1]
Rationale: The Goddard Ground Station handles many small Earth-orbiting science satellites and can therefore provide the ground control necessary for this LEO spacecraft.

System Functional Goals

[FG.1] The spacecraft provides scientists with high-resolution images of selected sites on Earth's surface in order to monitor deforestation. [FR.3] [FR.4] [DP.3.2] [DP.4] [C.1.1] [C.2.2]

[FG.2] The spacecraft CDHC coordinates the actions of the spacecraft's onboard components. [FR.1] [FR.2] [FR.5] [DP.2.1] [DP.2.2] [DP.4] [DP.5.5] [DP.5.6] [C.1.2] [C.2.2] [C.2.5] [C.3.1]

High-Level Functional Requirements

[FR.1] The spacecraft CDHC shall manage the interactions between the subsystems. [FG.2] [DP.2.1] [DP.5.1] [DP.5.6] [C.1.2] [C.2.1] [C.2.2] [C.2.5] [C.3.1] [Ground EA.4]
Assumption: All interactions between the subsystems are channeled through the spacecraft controller.

[FR.2] The spacecraft CDHC shall determine the overall operating mode of the spacecraft. [FG.2] [DP.2.2] [DP.3] [DP.3.1] [DP.3.7] [DP.5.5]
Assumption: The spacecraft CDHC is aware of the states of all its subsystems

Fig. 11 Example information provided in level 1 intent specification.

This section focuses on the details of Levels 1 and 2 of the intent specifications to demonstrate requirements traceability throughout the SpecTRM-GSCs. The ground segment intent specification and STPA is also described to demonstrate the human-centered development process.

One of the main aspects of spacecraft engineering involves the tradeoffs that occur between the various subsystems. These tradeoffs are documented using intent specifications and realized in SpecTRM models. The models described in this paper illustrate the process of making high-level design decisions and how those decisions impact the tradeoffs made at the lower levels of design.

The example provided in the TeleSub intent specification below involves the design of Transmitter, Receiver and Antenna hardware components. Elements of their design stem from high-level mission decisions concerning the spacecraft's orbit altitude and inclination. This information is provided in Level 1 of the intent specification (see Fig. 11).** As seen in Level 2 (System Design Principles) of the Spacecraft Mission Intent Specification (Fig. 12), the orbit altitude and inclination impact the average and best communication windows, when the spacecraft is in view of the selected Ground Station.

The orbit altitude and inclination impacts the data rate, transmission frequency and bandwidth, power and size of the Transmitter, Receiver and Antenna components. The designs of these components can be found in Level 2 of their respective intent specification. Figure 13 illustrates the design principle in the transmitter intent specification that

** Note: A functional goal is a high-level objective of the system while a high-level functional requirement is a testable "shall" statement.

[DP.1] Spacecraft Orbit [TeleSub EC.1] [Transmitter EA.2] [Antenna EA.2] [Receiver EA.2] [Transmitter DP.1]

[DP.1.1] The altitude of the deforestation satellite is 1,000 km above the Earth's surface. Using Equation 1, at this altitude the period P of rotation of the spacecraft around is 105 minutes, where 6378.14 is the radius of the Earth. [L.2]
Rationale: The payload of the spacecraft is a camera that is designed to operate in a low-Earth orbit, in particular at an altitude of 1,000 km.

Equation 1.

$$P = 1.658669 * 10^{-4} \cdot (6378.14 + 1000)^{3/2}$$

[DP.1.2] The orbit inclination of the spacecraft is 60 degrees. [L.2]
Rationale: The orbit inclination was chosen in order to provide coverage for all types of forests around the globe up to the Boreal Forest in Northern Canada and Russia.

[DP.1.3] The maximum time in view of the Ground Station for the spacecraft is 15 minutes per pass. The average time in view of the Ground Station for the spacecraft is approximately 12 minutes per pass. [EC.1] [FR.4] [L.1]
Rationale: See Equation 2, where Eta-min is 5 degrees based on the Ground Station's altitude and geography, Re is the radius of the Earth and H is the spacecraft altitude.

Equation 2.

$$\eta_{max} \quad \sin^{-1}(\sin\rho\cos\varepsilon_{min}) \quad 59.4° \quad where\ \rho \quad \sin^{-1}\left(\frac{R_E}{R_S + H}\right)$$

$$\lambda_{max} = 90° - \varepsilon_{min} - \eta_{max}$$

$$T_{max} = P\left(\frac{\lambda_{max}}{180°}\right) = 15\text{min}$$

$$T_{ave} = 0.8 \cdot T_{max} = 12\text{min}$$

Fig. 12 Sample level 2 system design principle.

flows down from the orbit calculations performed in the spacecraft mission intent specification. One of the benefits of using SpecTRM is that rationale is captured after every design decision is made. As can be seen in the documents, all principles have an associated rationale that describes why the decision was made. Furthermore, linking (underlined hyperlinked text located between square brackets) is available within the document as well as between documents, allowing users to link high-level design decisions to subsystem and component designs and vice versa.

[DP.1] The Downlink Data Rate for sending the scientific data to the Ground Station is 80Mbps. [FR.1.1] [Mission DP.1]
Rationale: In accordance with the amount of data that needs to be transferred during each pass, and based with the orbital parameters that define the average time the spacecraft will be in view of the ground station during each pass, the required communication data rate was calculated to be at least 76.8 Mbps (see Equation 1). Based on these calculations, a data rate of 80 Mbps was chosen.

Equation 1.
R = Dr*M/(F*Tmax-Tini) Dr = 256Mbps Tmax = 15min Tini = 2min M = 3 F = 0.8

Fig. 13 Sample level 2 component design principle.

Similarly, specifying the ground segment design and operations begins with identifying the high-level functional goals for the system and the assumptions made about its environment (the assumptions are justified by tracing them to the section in the other intent specifications that validates them). Operator tasks are considered to be within the system because the tasks must be designed together with the other parts of the system.

After the high-level functional requirements are defined and the preliminary hazard analysis (PHA) is performed, a preliminary task analysis (PTA) is performed. The PTA consists of cognitive engineers, human factors experts, and operators together specifying the goals and responsibilities of the operators, the task allocation principles to be used, and operator task and training requirements. Automation can be introduced in a system to provide some assistance to the human operator; it can also, however, have a much more essential role, replacing some key perceptual and cognitive functions of the operator. The involvement of the operators at this stage is very important, as they are the final users of the system and because they can provide a description of their needs for assistance to the system designers, engineers, or managers. The PTA and the PHA go hand in hand, and several iterations are necessary as the system designers acquire a better understanding of the operators' responsibilities and of the different human factors to take into consideration.

A PTA process was used to define the operators' goals and responsibilities and to allocate high-level functional requirements. Requirements are then defined for each component of the Ground Segment. It should be noted that the automation requirements are derived from the operator task analysis, not vice versa (the more common design approach).

At Level 2, using the system requirements and design constraints as well as the other information that has been generated to this point, system design principles are generated and tasks are allocated to the system components (including the operators) to satisfy the requirements, task allocation principles, and operational goals. It is important to note that this process will involve several iterations as the results of analysis, experimentation, review, etc. become available.

More precisely, the results of the analyses made in Level 1 are traced down in Level 2 into an operator or user task specification and user interface design principles. The system design principles are iteratively derived from the Level 1 requirements and the Level 2 user task specifications. Simulations, experiments, engineering analyses, and other verification and validation procedures can be used. The findings are then used to refine and improve the automation and the HMI designs and the operators' tasks and procedures, eventually allowing the development of a complete user manual (described in Level 6 of intent specifications). As in Level 1, operator input is an essential part of the documentation and manual creation process.

In order to identify a set of acceptable human task principles and system and HMI design principles, a series of simulations, experiments, engineering analyses, and other verification and validation (V&V) procedures are needed. This observation implies a strong involvement of the operators as well as the engineers. Ideally, the simulation would start early to assess the different design options individually. This process is nonetheless very costly, and in some cases a very simplified simulation or a survey of the end users' opinions is preferred. This problem can be solved by using a blackbox model of the automation as a prototype that can be run on actual HMIs. The blackbox models are formal and executable, and do not need any physical implementation. Those formal and executable models are constructed in the third level of our methodology, based on system design principles and controller tasks defined in Level 2 of the Intent Specifications. A preliminary design is created for the automation and the HMI, including a set of controller tasks, then the appropriate blackbox model of the automation are built and executed with the HMI. The results of the simulation are then used to review the automation and the HMI design, and the controller's tasks and procedures.

With respect to the human-automation interface, it is important to ensure that the information is easily and quickly accessible by the human user and that it is displayed in a way that matches his/her working behavior, methods and decision-making; no additional perceptual or cognitive workload should be introduced. The PHA, PTA, former accidents and incidents, and controllers' preferences are used again here to determine what information is needed and what part of this information is supposed to be provided by the automated tool.

The next step in the process involves validating the system design and requirements and performing trade studies required to select from among a set of design alternatives. The operator tasks are modeled using our state-machine language and formally analyzed and executed with the specification of the automation behavior simultaneously. Using a combination of simulation and formal analysis, the combined operator task model and blackbox automation

behavior models can be evaluated for potential problems, such as task overload and automation design features that can lead to operator mode confusion. The analysis performed at this stage of system design and development is not meant to replace standard simulator analysis, but rather to augment it. The formal analysis and automated execution of specifications focus on safety, rather than usability. Formal analysis is indeed better able to handle safety while usability is better tested in simulators. The analysis can also assist in designing the simulator scenarios needed to evaluate safety in the more standard simulator testing.

E. Level 3 Intent Specifications

As the Level 1 and 2 intent specification models are refined and augmented by the results of the hazard analysis, Level 3 formal models of the externally visible behavior of the system, subsystems and components written in SpecTRM-RL are developed. For example, Fig. 14 illustrates a visualization of the formal model of the system-level blackbox behavior.

The graphical model depicts the control loop of the spacecraft mission and how the various subsystems interface with the controller. The shaded part of the model describes the required behavior of the Spacecraft Mission CDHC. There are three main parts of this description: the supervisory mode specifies the current controller of the component (in case there are multiple controllers); the current control mode for the component (Startup, Idle, Repositioning, Data Collection, Maintenance, Safe and Diagnosis); and, to the right of the solid line, the controller's current state model of the controlled components. At any time, a controller only has a model, inferred from inputs and other information, about the real state of the components. In the example below, the inferred state model has three state variables, representing information about each of the three subsystems that were modeled as SpecTRM-GSCs and interact with the spacecraft controller. The graphical notation also shows the possible values for these state values; for example, the "PayloadState" state variable can have the values "Unknown," "Idle," "Gimbaling" or "Taking Picture." The boxes external to the gray-shaded area are devices external to the system, in this case the four subsystems that interact with the spacecraft controller (ADS, RCS, Payload and TeleSub). In a complete specification, the devices external to the CDHC would also be modeled using SpecTRM-RL and the models could all be analyzed and executed together.

The behavior of the spacecraft controller and the various spacecraft subsystem and components, i.e. the logic for sending output commands to the various external devices and changing the inferred system state, is specified using a tabular notation called AND/OR tables. The rows of the tables indicate AND relationships, while the columns represent ORs. Figure 15 shows transition conditions required for the TeleSub Control Mode to take the values "Startup," "Idle," "Transmitting" and "Receiving." Using the example AND/OR tables in Fig. 14, the TeleSub Control Mode element will transition to a new mode if any of the columns in the transition table evaluate to true. In other words, if the either of the transmitters is operational, there is a compressed picture or telemetry in the queue

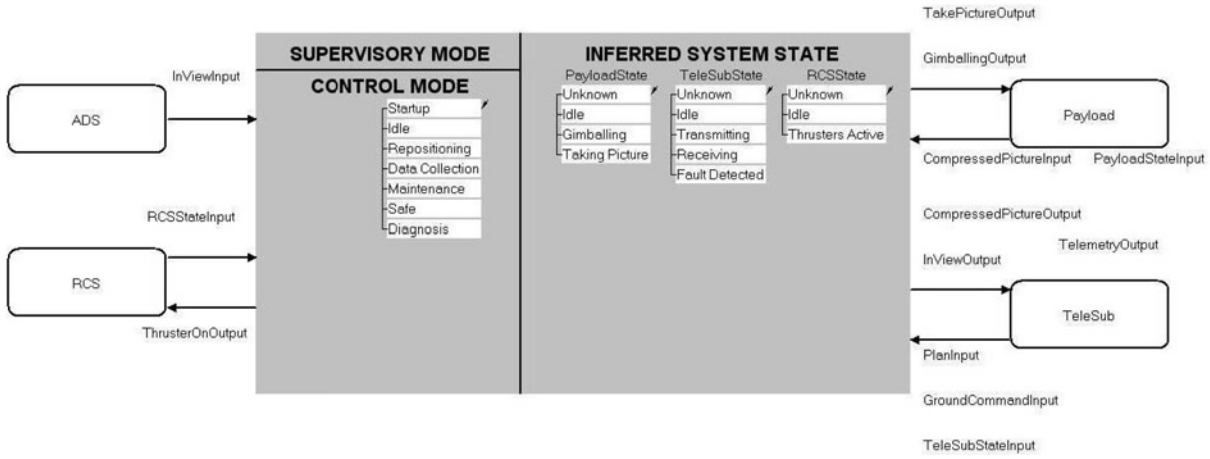

Fig. 14 System-level blackbox model.

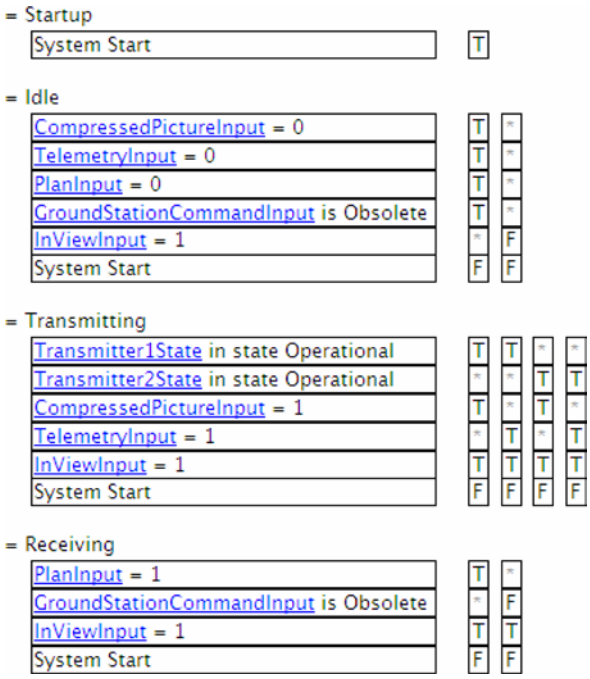

Fig. 15 Level 3 and/or table.

and the spacecraft is in view of the Ground Station, then the TeleSub controller will transition to "Transmitting" mode. Furthermore, the TeleSub will transition into "Receiving" mode if the spacecraft is within view of the Ground Station and the Ground Station has a plan to upload to the spacecraft.

As seen in the three transition tables, there are several statements with an asterisk in the OR column. This represents a "don't care" condition. In the example in Fig. 16, the TeleSub control mode will transition to "Transmitting" if one of the Transmitters is currently in the operational state. In this case, the subsystem "doesn't care" which of the Transmitters is operational as long as one of them can transmit the compressed picture or telemetry.

Functions in SpecTRM-RL can be written by either using an Ada-like programming script or external software packages such as the Matlab/Simulink environment. Functions allow users to calculate intermediate values from the inputs. SpecTRM uses these values as inputs to AND/OR tables that define either output triggering conditions, control mode and state variable transitions or macros. Macros allow users to abstract common logic and to increase readability. A macro takes a piece of an AND/OR table from another part of the model and gives it a name. This name can be substituted in for that table portion elsewhere in the model. The macro definition is a single AND/OR table. This table will evaluate to true or to false. When the table is true, the macro as a whole evaluates to true where it is used in other model elements. Similarly, if the table is false, then the macro evaluates to false.

The elements of the SpecTRM blackbox model include output commands, output values, modes, states, macros, functions, command inputs and input values. Other information contained at Level 3 includes communication channels, operational procedures, user models and the results of performing various analyses and simulations on the blackbox model.

IV. Analysis Results

After Levels 1–3 of the Intent Specifications, SpecTRM models and STAMP-Based Hazard Analysis and Risk Management strategy are completed, a simulation of the spacecraft SpecTRM models can be performed to help visualize the performance of the system. Simulations in SpecTRM can be visualized through the animation of the diagram that represents the blackbox model of the system. The animation includes highlighting the current values of the state and mode elements in yellow and displaying the current values of inputs and outputs in blue text under the element names. Obsolete data values are shown in green text under the element names. Time is shown in the upper

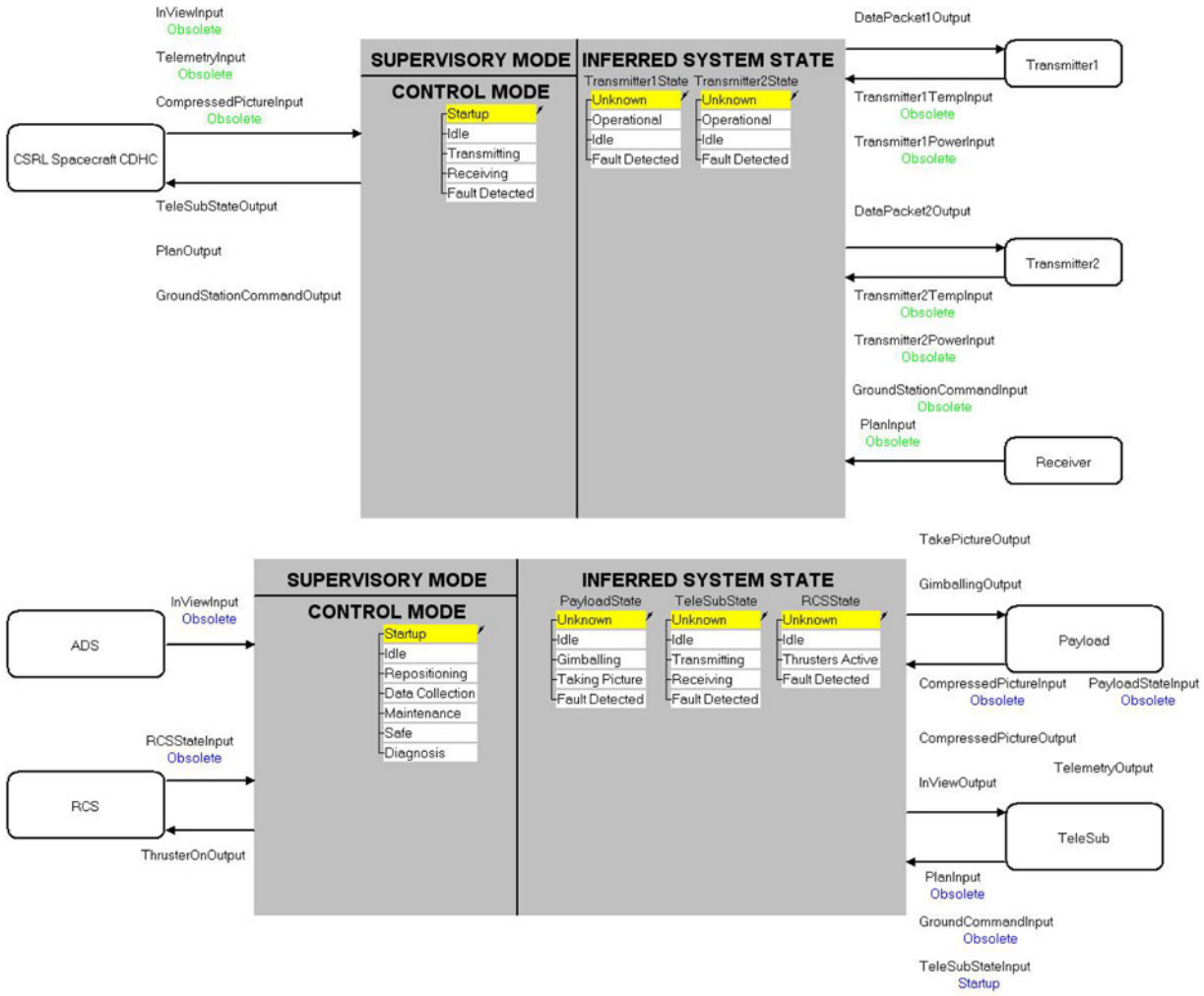

Fig. 16 Screen capture of spacecraft simulation in SpecTRM at startup.

left hand corner of the visualization window. As the simulation time progresses, the input, output, state and mode values update to reflect the new data. A side bar lists all the element values of all the models in the simulation. The bar located on the bottom of the visualization provides an event log showing detailed timing information.

The user is also able to set the time resolution of the simulation. Since the spacecraft state updates every 10 milliseconds, the simulation updates every 5 milliseconds, where 5 milliseconds in simulation time correspond to one half a second (or 500 milliseconds) in real time. In the simulation, each pass and blackout takes 10 seconds in real time.

Two SpecTRM models are used in the example simulation, the Spacecraft CDHC (the bottom blackbox model) and the Telecommunication Subsystem (TeleSub) (the top blackbox model). These models interact with one of another, exchanging data in the form of inputs and outputs. The simulation considers several scenarios of the spacecraft operation as described below:

1) The Spacecraft begins in Startup mode when the simulation starts. Startup modes are always included in system definitions so that the system does not start in an unsafe state. see Fig. 16.
2) After the Spacecraft has been powered up, the Attitude Determination Subsystem (ADS) notifies the CDHC that the Spacecraft is in view of the Ground Station. The CDHC notifies the TeleSub to create a connection

with the Ground Station. The Ground Station then uploads a plan to the spacecraft during so that it may begin taking pictures of the selected sites described in the plan.

3) The CDHC is then notified by the ADS that it is not in view of the Ground Station. The Spacecraft enters Idle Mode.
4) During the blackout period, the Plan Executer sends a command to the camera to take a picture and the CDHC enters Data Collection Mode. see Fig. 17.
5) After finishing the plan, the Spacecraft reenters Idle Mode. The Spacecraft is still out of view of the Ground Station.
6) For purposes of demonstration, the next "In View" time segment lasts the remainder of the simulation. Therefore at t = 15.5 seconds real time and t = 310 milliseconds spacecraft time, the ADS sends a message to the CDHC that the spacecraft is in view of the Ground Station. Because the Ground Station is in view and a picture resides in memory, the TeleSub transitions to Transmitting Mode and sends the picture file to the Transmitter for downlink to Earth.
7) As the simulation progresses, the temperature of the sensor associated with Transmitter 1 is slowly increasing. At t = 22 seconds real time, the sensor detects the temperature of Transmitter 1 to be 50°. This information is always transmitted to the TeleSub and therefore the TeleSub registers the problem with Transmitter 1 and switches to the redundant Transmitter 2. see Fig. 18.

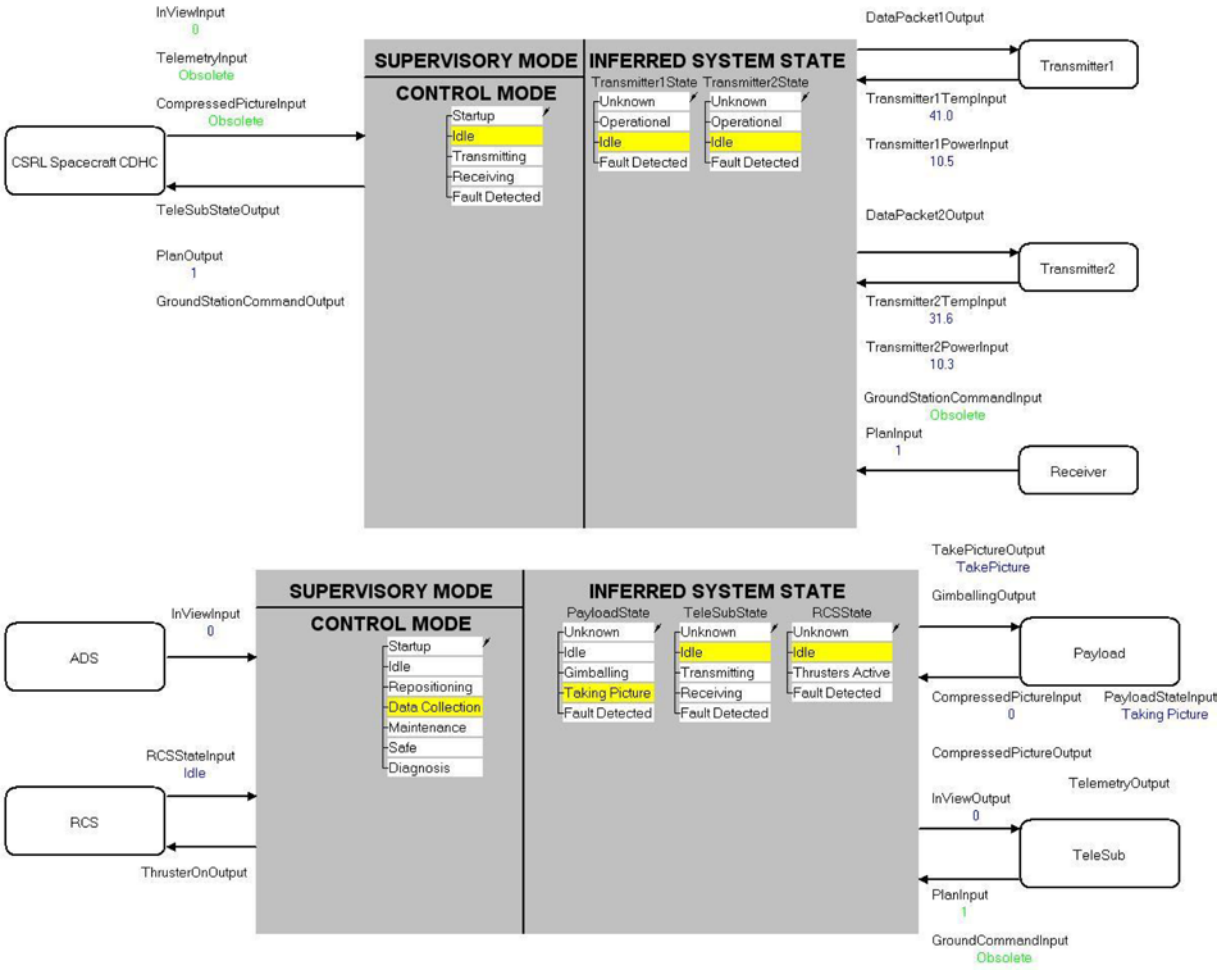

Fig. 17 Screen capture of spacecraft simulation in SpecTRM during data collection.

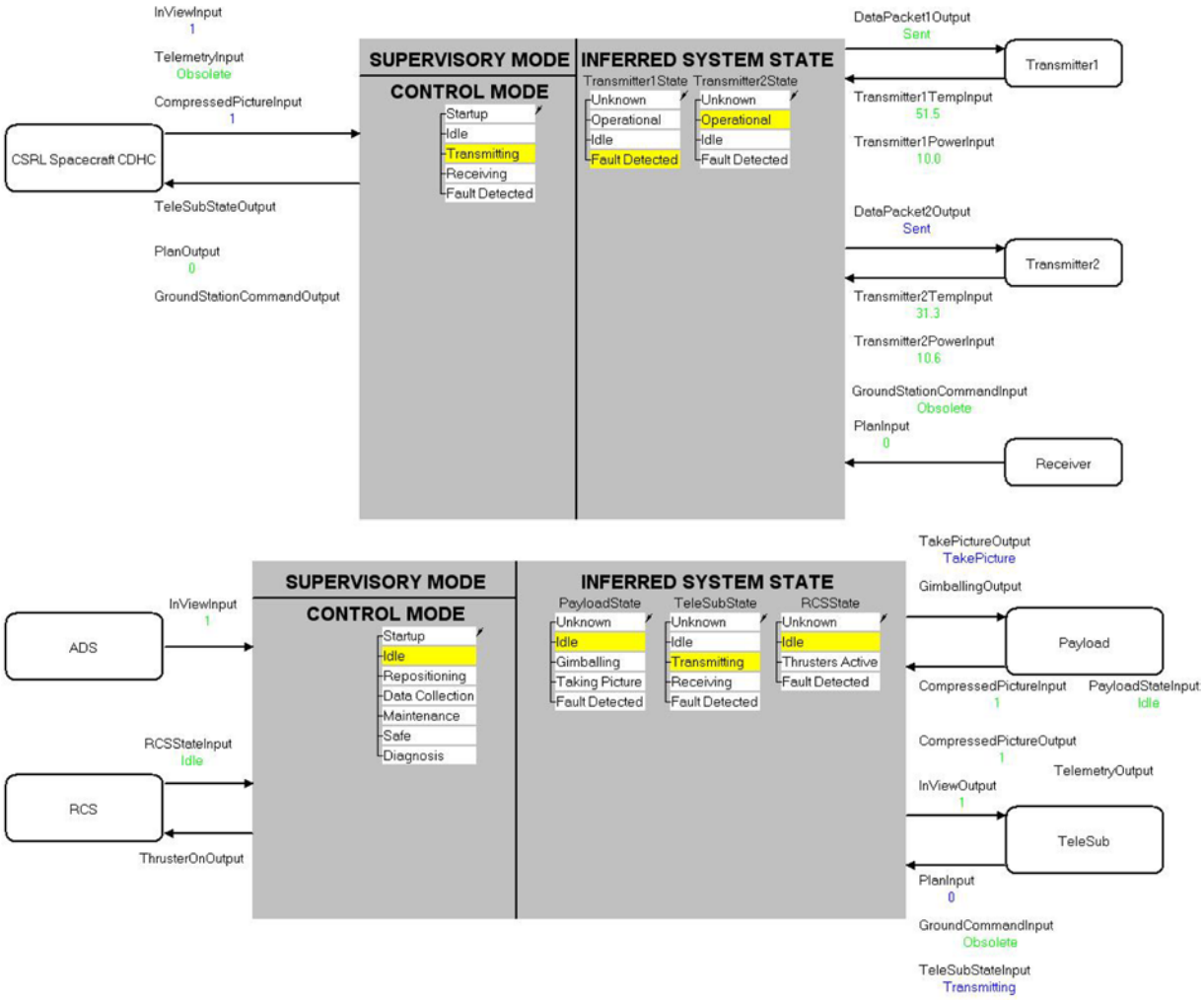

Fig. 18 Screen capture of spacecraft simulation in SpecTRM with transmitter 1 failure.

8) Again, throughout the first part of the simulation, Transmitter 2 is experiencing a gradual increase in the power through it's power sensor. At t = 30 seconds real time, Transmitter 2 reaches 11 Watts. The TeleSub recognizes that the threshold has been reached and sends a message to turn off the second transmitter.
9) Because both transmitters are now outside their nominal operating temperature and power ranges, the TeleSub enters Fault Detected Mode. Because each subsystem's state is always transmitted to the CDHC, the CDHC becomes aware that there is a failure in the TeleSub. Consequently, the CDHC enters Safe Mode and the simulation ends. see Fig. 19.

Through simulation scenarios such as these, the logic of the system can be tested before any software or hardware implementation has been performed. The simulation visualizations aid the engineers in identifying behaviors of the system that are unexpected or unwanted. Furthermore, these simulation scenarios can be written to reflect possible failures that may occur during operations so that the diagnosis approaches can be tested.

In addition to simulation, the state machine model underlying the SPECTRM-RL specification can be formally analyzed for such properties as completeness, non-determinism, robustness, and potential for causing operator mode confusion.

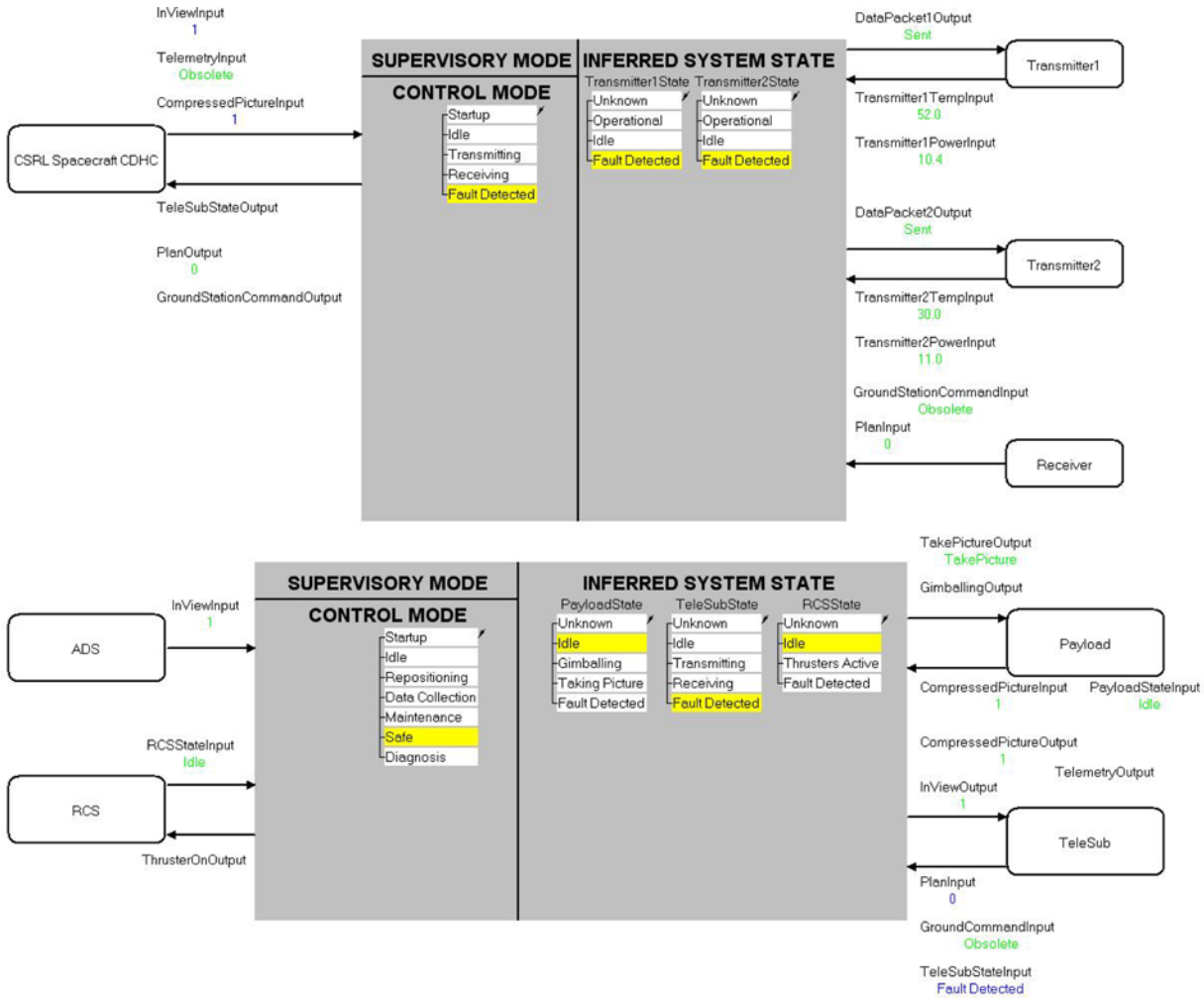

Fig. 19 Screen capture of spacecraft simulation in SpecTRM in safe mode.

V. Comparison to Other Current Approaches

Several other approaches can be used to perform piecewise safety analyses of complex systems. While these approaches may be useful for parts of the analysis necessary to develop and operate safe systems, none of these approaches provide a seamless integration of design for safety techniques with the system engineering process throughout the entire lifecycle of complex systems. For example, traditional top-down hazard analysis approaches, such as Fault Tree Analyses (FTAs), work well for simple electro-mechanical systems where the relationships between events can be defined by direct causal relations that are easily expressed using logical AND and OR statements. Similarly, bottom-up approaches such as Failure Modes and Effect Analysis also depend on direct relationships between events, which are usually available in the case of electro-mechanical systems in an advanced design stage. However, for modern systems that include digital systems, software, and complex human-automation interactions, it becomes difficult to model the relationships between system components directly. Moreover, FTAs and FMEA-type analyses require a fairly complete or advanced design to be effective. As such, they are very difficult to use during the early phases of system development where most of the benefits of an integrated design for safety approach can be reaped. Alternative approaches exist to handle the difficulties associated with the lack of detailed information available during the early system architecture definition phases.[13]

When a detailed design is available at a later stage of development, FTAs and FMEAs can be performed to assess some safety characteristics of the system. Very often, a probabilistic risk assessment (PRA) will be performed by assigning failure probabilities to different individual component failure modes, which allows analysts to combine probabilities and obtain a probabilistic estimate of a system-level failure mode. For certain, carefully-chosen systems, these estimates may be relatively accurate, for complex systems with little operational experience, there is likely to be much difficulties associated with these estimates.[14]

In addition to difficulties in handling software and HCI intensive systems, traditional methods are also limited to understanding the contribution of "softer" organizational factors affecting system safety, such as management pressures, limited resources, and independence of safety decision-making. Event-based methods are sometimes extended to include organizational factors. For example, researchers in the PRA field[15,16] have included some human and management factors into their methods for determining system failure probabilities. However, these attempts (apart from being based on difficult to validate assumptions about the mapping from factor to event) are static in nature, and thus will not capture the boiled frog phenomena of slowly increasing system risk. The STAMP/STPA approach does capture the dynamics associated with the migration of complex socio-technical systems towards states of higher risk and allows the design of more robust systems and better monitoring systems to detect and prevent this migration process.[11,17]

Apart from safety-centric analysis methods, other tools and techniques allow the blackbox specification of requirements for digital systems that should be used in an integrated design for safety process. The SCR (Software Cost Reduction) tools for example, describe software requirements in terms of their externally visible required behavior in a manner similar to SpecTRM-RL. However, these tools do not include the SpecTRM intent specification methodology that allows the recording and tracing of requirements and design rationale across different abstraction levels. Moreover, for large digital systems, the notation used to specify system requirements may limit the scalability of SCR-type models. Experiments have shown that simpler AND/OR tables were easier to understand by domain experts, thus increasing their ability to review specifications of large systems.[18] [Dulac 2002].

Another approach to safety analysis, used primarily in Europe, is called safety-cases, often referred to in the US as dependability cases by the Software Engineering Institute (in fact, safety-cases are considered a subset of the overall dependability case). A safety-case is a demonstration of software product dependability. The demonstration typically takes the form of a presentation of evidence with argumentation linking the evidence to the claim of software safety. Safety-cases have a post-design focus, which is the opposite of the STPA approach. The emphasis of a safety-case is on proving that the current design is safe, therefore considerable emphasis is placed on creating a proof of system safety whether the design is safe or not. In the STPA approach, safety is designed into the system from the beginning; in this paradigm, the safety-case becomes trivial.

In summary, the benefits of the proposed approach over current approaches include the following:

- Ease of directly modeling the relationships between system components
- Support for safety analysis during the early phases of system development
- Not limited to PRA or event-based analysis to assess system safety
- Captures system dynamics and how a socio-technical system can migrate to states of higher risk
- Use of a formally analyzable specification language that has shown to be easily readable and reviewable

VI. Conclusion

This paper presents a new design for safety methodology that provides engineers with a seamless approach to integrating safety analysis with typical system engineering design activities. The methodology is demonstrated through modeling and analyzing a complex, safety critical spacecraft mission. The approach is based on systems theory and integrates the technical components with the human factors and the social and organizational aspects of complex system. It includes the use of Intent Specifications, SpecTRM, SpecTRM-GSCs, and STAMP-Based Hazard Analysis and Risk Management.

As demonstrated throughout this paper, this systems theoretic approach to safe system design helps engineers to stretch the limits of complexity and intellectual manageability of the systems they build while maintaining confidence in their expected behavior, particularly safety and mission accomplishment. Through the use of the methodology presented in this paper, we believe engineers can create safer, more reliable and cost effective solutions to the variety

of new endeavors on the horizon for the aerospace industry. Automated tools have been developed to support the methodology.

Acknowledgments

This work was partially supported by the NASA Engineering for Complex Systems program (NAG2-1843) and NSF ITR grant (CCR-0085829).

References

[1]Leveson, N.G., "Intent Specifications: An Approach to Building Human-Centered Specifications," *IEEE Trans. on Software Engineering*, January 2000.

[2]Leveson, N.G., *Safeware: System Safety and Computers*, Addison-Wesley Publishing Company, Reading, MA, 1995.

[3]Stephenson, A.G. *et al*., "Mars Climate Orbiter Mishap Investigation Board," NASA Archives, November 10, 1999.

[4]Clements, P., and Linda, N., *Software Product Lines: Practices and Patterns*, Addison-Wesley, Boston, MA, 2002.

[5]Weiss, K.A., "Incorporating Modern Development and Evaluation Techniques into the Creation of Large-Scale, Spacecraft Control Software," Doctoral Dissertation, Massachusetts Institute of Technology, Jan. 2006.

[6]Leveson, N.G., "A New Accident Model for Engineering Safer Systems," *Safety Science*, Vol. 42, No. 4, 2004.

[7]Rasmussen, J., "Risk Management in a Dynamic Society: A Modeling Problem," *Safety Science*, Vol. 27, No. 2/3, 1997.

[8]Leveson, N.G., "The Analysis of a Friendly Fire Accident using a System Model of Accidents," *International Conference of the System Safety Society*, Denver, 2002.

[9]Leveson, N.G., Daouk, M., Dulac, N., and Marais, K., "Applying STAMP in Accident Analysis," *Workshop on the Investigation and Reporting of Accidents*, 2003.

[10]Leveson, N.G., "A New Approach to Hazard Analysis for Complex Systems," *International Conference of the System Safety Society*, 2003.

[11]Leveson, N.G., Dulac, N., Cutcher-Gershenfeld, J., Carroll, J.S., Barrett, B., and Friedenthal, S., "*Risk Analysis of the NASA Independent Technical Authority*", available at http://sunnyday.mit.edu/ITA-Risk-Analysis.doc

[12]Weiss, K.A., Ong, E.C., and Leveson, N.G., "Reusable Specification Components for Model-Driven Development," *Proceedings of the International Conference on System Engineering* (INCOSE '03), July 2003.

[13]Dulac, N. and Nancy, L., "Incorporating Safety in Early System Architecture Trade Studies," *International System Safety Conference* (ISSC05), San Diego, CA, August 2005.

[14]Freudenburg, W.R., "Perceived Risk, Real Risk: Social Science and the Art of Probabilistic Risk Assessment," *Science, New Series*, 1988, Vol. 242, No. 4875, pp. 44–49.

[15]Paté-Cornell, E.M., "Organizational Aspects of Engineering System Safety," *Science*, Vol. 250, November 1990, pp. 1210–1217.

[16]Paté-Cornell, E.M., and Murphy, D.M., "Human and Management Factors in Probabilistic Risk Analysis: The SAM Approach and Observations from Recent Applications," *Reliability Engineering and System Safety*, Vol. 53, No. 2, Aug. 1996, pp. 115–126.

[17]Dulac, N., Leveson, N., Zipkin, D., Friedenthal, S., Cutcher-Gershenfeld, J., Carroll, J., and Barrett, B., "Using System Dynamics for Safety and Risk Management in Complex Engineering Systems" *Winter Simulation Conference (WSC05)*, Orlando, FL, Dec. 2005.

[18]Mark, Z., Nancy, L., and Kristina, L., "Investigating the Readability of State-Based Formal Requirements Specification Languages," *Int. Conference on Software Engineering*, Orlando, May 2002.

Richard Doyle
Associate Editor