SAFETY GUIDED SPACECRAFT DESIGN USING MODEL-BASED SPECIFICATIONS

Cody Fleming[(1)], Takuto Ishimatsu[(1)], Yuko Miyamoto[(2)], Haruka Nakao[(3)] , Masa Katahira[(2)], Nobuyuki Hoshino[(3)], John Thomas[(1)] , Nancy Leveson[(1)],

[(1)] Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139-4307, USA
Email: chf44@mit.edu, takuto@mit.edu, jthomas4@mit.edu, leveson@mit.edu
[(2)] Japan Aerospace Exploration Agency, 2-1-1 Sengen, Tsukuba-shi, Ibaraki 305-8505, Japan
Email: miyamoto.yuko@jaxa.jp, katahira.masafumi@jaxa.jp
[(3)]Japan Manned Space Systems Corporation, Urban Bldg., 1-1-26, Kawaguchi, Tsuchiura, Ibaraki 300-0033, Japan
Email: nakao.haruka@jamss.co.jp, hoshino.nobuyuki@jamss.co.jp

ABSTRACT

Most of the basic design decisions affecting safety are made in the concept development stage of system development. Once these decisions are made, the cost of changing them later in development is often enormous and perhaps even infeasible. At the same time, most hazard analysis methods require a fairly complete design to be most useful. By the time enough design has been completed for hazard analysis to be able to identify flaws in the design, the cost of rework and changing basic decisions is great.

The solution to these problems is to integrate safety tightly into the system development process from the very beginning of system conception. In this paper, we describe a process for tightly intertwining design and analysis starting in the early development stages. The process involves defining safety as a control problem (STAMP) and using model-driven development and executable requirements specifications.

1. INTRODUCTION

There are several challenges in ensuring the safe development, deployment, and operation of spacecraft. First, while most design decisions affecting safety (and other system properties) are made early in the design cycle, most hazard analysis techniques require a much more mature design in order to be most effective. Sufficient design maturity is not typically achieved until at least PDR, or even CDR, and the best that can be done to achieve safety at this late stage is the addition of extra redundancy or other costly and often not very satisfactory fixes. This approach can result in significant design changes and has ramifications for cost, schedule, and performance. Second, hazard analysis is often done as a separate activity by an independent group and communication with the spacecraft designers can be limited.

The solution to these problems is to integrate safety tightly into the system development process from the very beginning of system conception. While spacecraft designers clearly think about safety as they make design decisions, they have few tools to help them make optimal safety-related decisions. In this paper, we will describe a process for tightly intertwining design and analysis (and system testing) starting in the early development stages. The process involves defining safety as a control problem (STAMP) and using model-driven development and executable requirements specifications.

In this framework, design and analysis are both supported by the use of executable models that start from system requirements and are iterated and refined as basic design decisions are made. The models have a formal foundation so they can be analyzed as well as executed and the system simulated using automated tools. Because the specifications and models are executable, much important system testing can be performed before the actual components are created, therefore reducing the very costly rework occurring when fundamental design flaws are discovered during system testing near the end of development. The models, which are based on state machines, use basic control concepts familiar to all spacecraft engineers and carefully designed experiments have found them to be easily readable and reviewable by engineers [1,2].

In this paper, we demonstrate the process on a generalized version (without proprietary information) of a real JAXA satellite. The new hazard analysis technique, called STPA, has been presented at a previous IAASS conference [3]. STPA can be integrated into a sophisticated spacecraft development process using model-based development and automated tools to assist with important design decisions and finding design weaknesses or flaws during the entire development process. The resulting documentation

incorporates traceability from requirements to design and implementation and supports the specification of design rationale.

2. SPACECRAFT EXAMPLE

The analysis presented herein is representative of the process used on a real, sophisticated spacecraft development program. This example system consists of a spacecraft bus and integrated payload, where management of operational modes is performed on the ground by human operators, as well as automatically with spacecraft software. Our analysis focuses specifically on hazards that arise due to the interaction between the payload and the rest of the spacecraft, as opposed to other more general spacecraft hazards such as launch loads, thermal gradients, or other space-based environments. The approach presented in this paper can be extended to a more general set of hazards.

Depending on a program mission, a payload's primary objectives can have adversarial effects on the rest of the spacecraft, if not managed correctly. For example, if the spacecraft is required to do a major orbital maneuver, there may be competition amongst several sub-systems for power. Therefore, during momentum dumping, priority should be given to the Guidance, Navigation, and Control sub-system. If the payload is operating at full power consumption along with reaction wheels or GMGs, then the system risks running out of power.

Another example of potentially hazardous interaction between the payload and other mission aspects is the use of active microwave sensor devices on meteorological satellites, such as the Synthetic Aperture Radar on the ERS satellites [4] and many other applications. The radiation given off by the payload instruments can be hazardous to the spacecraft bus (or launch vehicle) if mode selection and operation are not handled properly. These two example hazards, which represent the interaction of otherwise *required* properties of the payload (power consumption, active radiation), provide a basis for the hazard analysis in this paper.

3. HAZARD ANALYSIS

STAMP (Systems Theoretic Accident Modeling and Processes) is a model of accident causation that treats safety as a control problem, rather than as a failure problem [5]. Its associated hazard analysis technique, STPA, is also grounded in systems theory. Figure 1 shows a control structure to enforce safety constraints. Each hierarchical level of the control structure represents a control process and control loop with actions and feedback. Note that these control structures can include higher levels of organizational controls, both in design and operation [5].

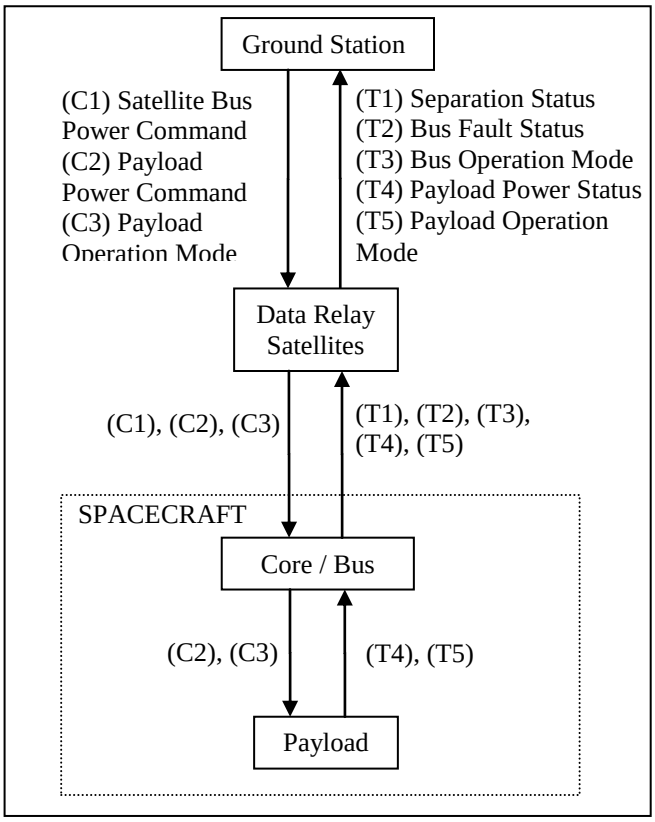

Figure 1: Hierarchical Control Structure for S/C Operation

While unsafe control includes inadequate handling of failures, it also includes system and software design errors and erroneous human decision making. In this systems theoretic framework, accidents are viewed as the result of inadequate enforcement of constraints on system behavior. The reason behind the inadequate enforcement may involve classic component failures, but it may also result from unsafe interactions among components operating as designed or from erroneous control actions by software or humans.

3.1. STAMP

Human and automated controllers use a process model[1] to determine what control actions are needed. The process model contains the controller's understanding of 1) the current state of the controlled process, 2) the desired state

[1] Often called a 'Mental model' for human controllers

of the controlled process, and 3) the ways the process can change state. Software and human errors often result from incorrect process models; therefore, accidents can occur when an incorrect or incomplete process model causes a controller to provide control actions that are hazardous. For example, the ground station crew thinks the spacecraft has separated from the launch vehicle and instructs momentum dumping, when in fact the spacecraft has not yet separated but provided incorrect telemetry. While process model flaws are not the only causes of accidents involving software and human errors, it is a major contributor.

STAMP is based on the observation that there are four types of hazardous control actions that need to be eliminated or controlled to prevent accidents:

1. A control action required for safety is not provided or is not followed
2. An unsafe control action is provided that leads to a hazard
3. A potentially safe control action is provided too late, too early, or out of sequence
4. A safe control action is stopped too soon or applied too long

3.2. STPA

STPA (System Theoretic Process Analysis) is a hazard analysis technique based upon STAMP. Identifying the potentially unsafe control actions for the specific system being considered is the first step in STPA. These unsafe control actions are used to create safety requirements and constraints on the behavior of both the system and its components. Additional analysis can then be performed to identify the detailed scenarios leading to the violation of the safety constraints. As in any hazard analysis, these scenarios are then used to control or mitigate the hazards in the system design.

Before beginning an STPA hazard analysis, potential accidents and related system-level hazards are identified along with the corresponding system safety constraints that must be controlled. This effort coincides with the generation of system level goals and subsequent architecture selection. Continuing the spacecraft example, consider a payload and bus with mutually competing requirements (e.g. resource allocation). The high level goals of such an earth observation satellite are shown above:

Table 1 shows such a table for the Ground Station control actions related to the Payload Power On/Off command. Note that the columns of the table represent the four types

System Level Goals

[G.1] Obtain frequent and accurate earth-observation measurement
[G.2] Achieve launch readiness by TBD date
[G.3] Obtain measurements from Low Earth Orbit LEO (~400 km) (this may actually be considered a mission constraint / requirement)

The accidents to be considered are: hardware damage due to RF radiation from active payload, and loss of power (due to payload interaction). The system-level hazards relevant to this definition of an accident include:

System Level Hazards

[H.1] Bus receives RF radiation from Payload
[H.2] Loss of Bus electric power during Payload operations

Based on the control structure in Figure 1 and the system-level goals and hazards listed above, there are necessary safety-related control actions for both the ground station and bus avionics.

Safety-Related Control Actions

Ground Station

[CA.1] Satellite Bus Power On/Off Command
[CA.2] Payload Power On/Off Command
[CA.3] Payload Operation Mode Select Command

Avionics (same as above)

Unsafe Control Actions

Next, for the two controllers with which we are concerned, we must determine how those controllers might exert unsafe control on the system—an action that has the potential to result in a hazardous system state or the lack of an action needed to prevent a hazardous system state.

To assist in identifying the hazardous control actions, we use a table to look at all these possibilities for each of the control actions. Only some of them will turn out to be hazardous but considering all unsafe control actions will identify those that are hazardous under certain conditions.

of hazardous control actions illustrated in Section 3.1, and each example scenario refers back to a system-level hazard.

Table 1: Potentially hazardous control actions for Ground Station

Control Action	Not Providing Causes Hazard	Providing Causes Hazard	Wrong Timing/ Order Causes Hazard	Stopped Too Soon or Applied Too Long
Payload Power On		Provided while bus is not in Mission mode [H.2]	Provided too soon before batteries have been replenished [H.2] Provided too soon during launch operations [H.1]	
Payload Power Off	Not provided while bus is in diagnostics or other non-observational modes [H.2]		Provided too late after change in payload mode may result in loss of power [H.2]	

A procedure has been developed to provide additional guidance for identifying the hazardous control actions during the first step of STPA [6]. The approach is based on the idea that many control actions are only hazardous in certain contexts. For example, a command to power on the payload is not hazardous by itself—it depends on the system state or state of the environment in which the command is given, for example if the solar arrays are not optional or the battery is not charged.. The procedure involves identifying potential control actions, identifying potentially hazardous states, and then analyzing which combinations together yield a hazardous control action.

There are two parts of the procedure that can be performed independently of the others. The first part deals with control actions that are provided under conditions that make the action hazardous. The second part deals with control actions that are not provided under conditions that make inaction hazardous.

Part A: Control actions *provided* in a state where the action is hazardous

In this procedure, a controller and the associated control actions are selected from the control structure. To continue the example, the ground station can provide two basic control actions: Payload Power On or Off. Next, the controller's process model is defined to determine the environmental and system states that affect the safety of the control actions.

Controllers use the states or values of the process model to determine what control actions to provide. In order to make safe decisions, the control algorithm must use process model variable values (i.e., system state or environmental values that are known to the controller). If the controller does not know the values of system state and environmental values that are related to hazards, then the controller cannot be designed to provide safe control actions. Figure 3 shows the required process model for the ground station operator to carry out its control safely. The required variables in the process model are identified by the definition of the system hazards.

Figure 2 shows the potential states of the system, which should then be implemented as process model components of the controller. The required variables in the process model are identified directly or through derivation by the definition of the system hazards. For example, hazard H-1 (RF radiation exposure) identifies the state of the payload operation, e.g. whether it is Launch Mode, as an important environmental variable in deciding which mode the payload should be in.

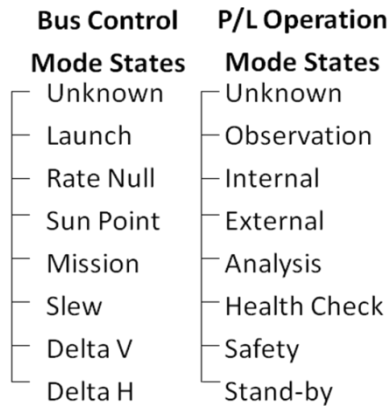

Figure 2: On-Orbit Spacecraft States

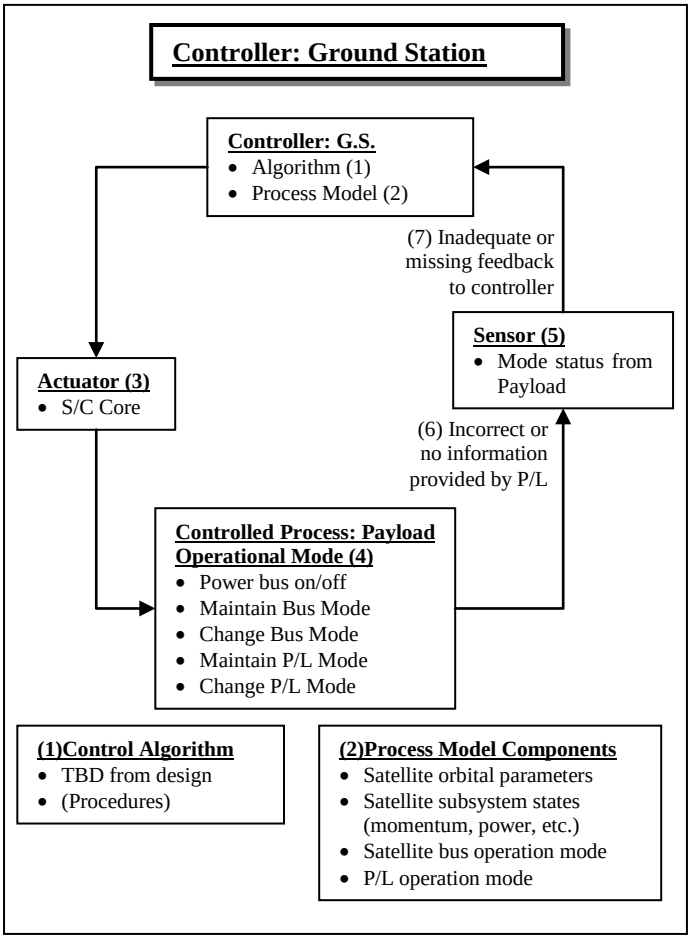

Figure 3: Control Diagram for Ground Station Operators

Once the process model variables have been identified, the potentially hazardous control actions can be identified and changed into safety requirements for the controller. These hazards are identified by examining each potential combination of relevant process model values in Figure 3 to determine whether issuing that control action in that state will be hazardous.

Part B: Control actions not provided in a state that makes inaction hazardous

This part of the procedure considers potential states in which the lack of a control action is hazardous. The same basic process is used: identify the corresponding process model variables and the potential values, create contexts for the action using combinations of values, and then consider whether an absence of the specified control action would be hazardous in the given context. Table 3 shows the hazardous control actions for the power off command not being provided.

Both Part A and Part B should be done for every control action, and this example obviously neglects "Payload Power on Not Provided" (the part b counterpart for Table 2), and "Payload Power off Provided" (the part a counterpart for Table 3). In this simple example the corresponding tables are not needed, as Table 1 already shows they are not hazardous. However, it is important to complete both parts, for every control action, in situations that are much more complex, and processes that have greater mode interaction.

Table 2: Contexts for Payload Power On *Provided*

Control Action	Bus Mode	Payload Mode	Hazardous Control Action?			Effect Mission Goals?
			Any time in this context	If provided too early	If provided too late	
Payload Power On *Provided*	Launch	(doesn't matter)	H.1, H.2	H.1, H.2	H.1, H.2	
	Rate Null	(doesn't matter)	H.2	H.2	H.2	
	Sun Point	(doesn't matter)	H.2	H.2	H.2	
	Mission	(doesn't matter)	No	No	No	G-1
	Slew	(doesn't matter)	No	No	No	
	Δ-V	(doesn't matter)	No	No	No	
	Δ-H	(doesn't matter)	H.2	H.2	H.2	

Table 3: Contexts for Payload Power Off *Not Provided*

Control Action	Bus Mode	Payload Mode	Hazardous Control Action?	Effect Mission Goals?
Payload Power Off *Not Provided*	Launch	Observation	H.1, H.2	
		Internal	H.2	
		External	H.1, H.2	
		Analysis	H.1, H.2	
		Health Check	H.2	
		Safety	No	
		Stand-by	H.2	
	Rate Null	*Same as Launch*	H.2	
	Sun Point	*Same as Launch*	H.2	
	Mission	(doesn't matter)	No	
	Slew	(doesn't matter)	No	
	Δ-V	(doesn't matter)	No	
	Δ-H	*Same as Launch*	H.2	

An important result of this approach is that tables allow for clear, concise translation of hazardous control actions to safety constraints on system behavior. Whenever a control action and set of process states may lead to a hazard, it should trigger the generation of an associated safety constraint.

Moreover, the structure of these tables and the logical relation between control actions, process states, and hazards lends itself to a formal, model-based safety verification approach.

4. MODEL-DRIVEN AND EXECUTABLE SPECIFICATIONS

As described in the Introduction, one of the difficulties in early project development is the lack of tools necessary to assist developers in making safety-related design decisions. Section 3 illustrates a method for performing hazard analyses in the early design phases[2], and now we introduce a model-based approach which helps ensure that safety-related specifications are developed and captured effectively.

In this paper, we have used a commercial toolset with a formal modeling language called SpecTRM-RL (Specification Tools and Requirements Methodology) to model the blackbox behavior required of the controllers. The specific toolset used here is less important than the desirable properties a tool should have in order to assist in safety-driven design. SpecTRM-RL is intended to satisfy two objectives: 1) to be easily readable by design engineers [1], and 2) contain an underlying formal model that can be used in mathematical analysis to check for completeness, consistency, and robustness. This formal, mathematical analysis helps designers to ascertain the specifications' efficacy in assuring safety, early in project development. SpecTRM-RL has been used to formally model the input/output behavior of software [7,8], but it can also be used to model the completeness and correctness of procedures used by human operators, as this example shows.

The underlying theory of SpecTRM-RL is described in detail elsewhere [1,2,7,8], and we include here only a brief introduction of the visual properties of the tool. Figure 4 shows three components of a specification for the spacecraft control logic: 1) a specification of the inputs and outputs to the controller 2) a specification of the control logic or algorithm of the system component that includes the available control modes, and 3) a specification of the process model of the controller that includes the inferred state of the system. In addition, the specification could include the supervisory modes of the controller, for example if the design includes control authority changes when the supervisor is in control mode.

Table 4 (below) illustrates an example of a SpecTRM-RL *and/or* table specification; it defines the criteria for the transition of Payload Mode Select input into Safety or Observation mode. A full specification would include all of the modes listed above in Figure 2. The far-left column of the *and/or* table lists the logical phrases of a predicate logic statement. Each of the other columns is a conjunction of those phrases and contains the logical values of the expressions. The rows of the table represent AND relationships while the columns represent OR relationships. The state variable takes the specified value

[2] These methods can and should be performed iteratively as the design matures and the specifications become more detailed.

(in this case, Safety or Observation, depending on the other spacecraft states) if any of the columns evaluate to true. If one of the columns evaluates to true, then the entire table evaluates to that state. A column evaluates to true if all the rows have the value specified for that row in the column. An asterisk denotes "don't care" while 'T' and 'F' denote true and false, respectively. Note that the underlying logic allows for consistency checks.

In this example, if the same set of states resulted in the simultaneous transition to both Observation and Safety, then the toolset will flag that the specifications are internally inconsistent. Each of the states and modes in the graphical representation of Figure 4 will be represented formally by an *And/Or* table, and the entire model can be checked for completeness (no missing transitions or modes), consistency (no contradictory specifications), and robustness (determinism, the specification of a response for every sequence of inputs).

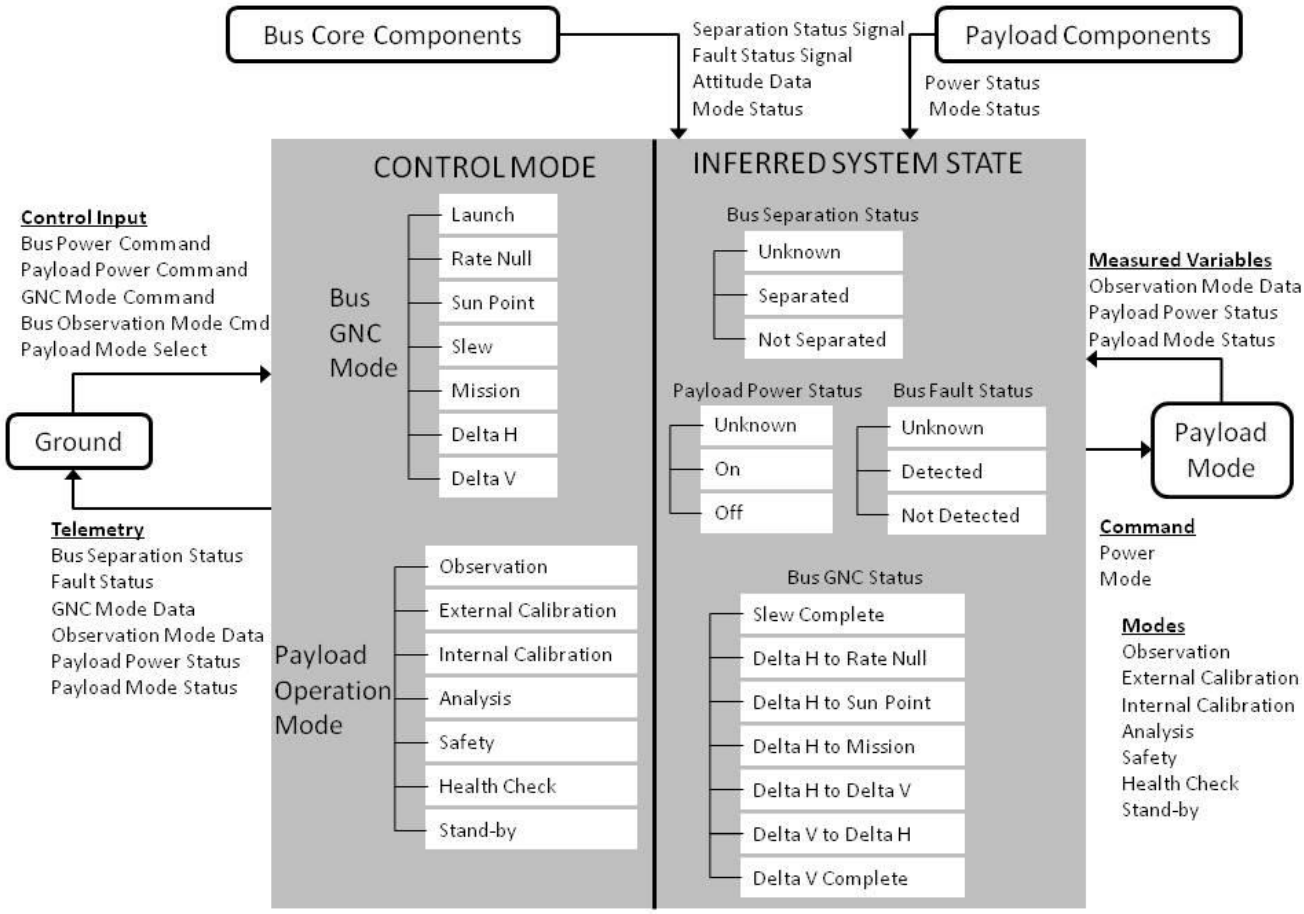

Figure 4: Graphical Specification of System (or Component) Model

In addition to analyses of completeness, consistency and robustness, SpecTRM safety analysis can be used for checking the existence of concrete hazardous conditions in the design. This is an extension of these systems- and control-theoretic concepts, where safe (and unsafe) control actions are a function of the context of process model variables. This extension allows for a systematic and logical formulation of safety constraints. The following subsection describes the Safety analysis steps.

1. Identify conditions which lead to system hazardous scenarios identified by STPA, and define the conditions using the state variables in the SpecTRM model described in Figure 3. For example:

   Conditions lead to system hazard Hazard Scenario: Power Loss:

   i. Payload Power is On +,
   ii. Bus Mode is Launch +,
   iii. Payload Operation Mode is Safety.

2. Recursively identify more detailed (refined) conditions that lead to the original condition,
3. Repeat 1 and 2 until all the conditions are explained by all inputs into process model.
4. Identify set of possible conditions, and eliminate impossible sets of impossible conditions (e.g. logically or physically impossible conditions)

Table 4: *And/Or* Table for Mode Select Logic

Control Input

Payload Mode Select

Description: This Control Input defines the different allowable bus control modes, along with the necessary states and data to activate the various modes

References: ↓ Separation Signal, GNC Status State, Bus GNC Mode, …

Appears In: ↑ Operator control commands

= Safety

Separation Signal is OFF	T	*	*	*	*	*
Time Since Mode status was Last Received > 90 seconds	*	T	*	*	*	*
Bus GNC Mode in mode Launch	*	*	T	*	*	*
Bus GNC Mode in mode Rate Null	*	*	*	T	*	*
Bus GNC Mode in mode Sun Point	*	*	*	*	T	*
…						

= Observation

Separation Signal is OFF	T	T
Bus GNC Mode in mode Launch	F	F
Bus GNC Mode in mode Rate Null	F	F
Bus GNC Mode in mode Sun Point	F	F
Bus GNC Mode in mode Slew	F	F
Bus GNC Mode in mode Mission	T	*
Bus GNC Mode in mode Delta H	F	F
Bus GNC Mode in mode Delta V	*	T

5. CONCLUSION & FUTURE WORK

This paper briefly illustrates a method for assuring safety in spacecraft early in the design process, when only the basic architecture and operational modes are known. This method, called STPA, is based on systems theory, where safety is an emergent property that arises due to component interaction and coupling; not merely component failure. In this framework, safety is assured through enforcing constraints on system (and thus component) behavior, where components are required to make safe control actions.

We introduce an extension of these systems- and control-theoretic concepts, where safe (and unsafe) control actions are a function of the context of process model variables. This new extension allows for a systematic and logical formulation of safety constraints.

The safety constraints identified using STPA can then be modeled in a formal specification language, where state and mode transitions of the system are modeled mathematically. The inputs and outputs to each controller in the system are then analyzed for completeness, consistency, and robustness.

The spacecraft example used here, consisting of a generic bus, payload, and ground station crew, is simple but relevant to many spacecraft development programs. We wish to extend these tools to more complex space systems that consist of a greater number of components, systems with increased coupling among components, and more mode-rich systems.

6. ACKNOWLEDGMENTS

The research described was supported by a grant from JAMSS. John Thomas is supported by a fellowship from Sandia Labs.

7. ACRONYMS

STAMP – Systems Theoretic Accident Model and Process
STPA – Systems Theoretic Process and Analysis
SpecTRM-RL – Specification Tools and Requirements Methodology

8. REFERENCES

1. Leveson, N.G. (2000) Intent Specifications: An Approach to Building Human-Centered Specifications, IEEE Transactions on Software Engineering.
2. Zimmerman, M., Lundqvist, K., Leveson, N.G., (2002) Investigating the Readability of State-Based Formal Requirements Specification Languages, International Conference on Software Engineering, Orlando.
3. Ishimatsu, T., et al. (2010) "Modeling and Hazard Analysis using STPA" 4th IAASS Conference.
4. Drinkwater, M.R., Liu, X., (1997) ERS Satellite Microwave Radar Observations of Antarctic Sea-Ice Dynamics, 3rd ERS Symposium, Florence.
5. Leveson, N.G. (2011) *Engineering a Safer World: Systems Thinking Applied to Safety*. MIT Press.
6. Thomas, J., Leveson, N.G., (2011) "Performing Hazard Analysis on Complex, Software- and Human-Intensive Systems", *Proceedings of the 29th International Conference on Systems Safety*, Las Vegas, NV.
7. Stringfellow, M., et al. (2007) "Safety-Driven Model-Based System Engineering Part I", MIT Technical Report.
8. Leveson, N.G. (2002) "Model-Based Analysis of Socio-Technical Risk", Technical Report, Engineering Systems Division, MIT.