

**Safety-Driven Model-Based
System Engineering Methodology Part I:
Methodology Description***

December 16, 2007

Prepared by:

Margaret Stringfellow Herring, MIT
Brandon D. Owens, MIT
Dr. Nancy Leveson, MIT

Dr. Michel Ingham, JPL
Dr. Kathryn Anne Weiss, JPL

* This research is supported by JPL University Subcontract 1297013

Table of Contents

Acknowledgements.....	4
1. Introduction.....	5
2. Background.....	5
2.1 Intent Specifications.....	5
2.2 STAMP.....	9
2.3 STPA.....	10
2.4 SpecTRM and SpecTRM-RL.....	16
2.5 State Analysis.....	18
2.6 Summary.....	22
3. System Engineering Methodology.....	22
3.1 Step 1: Identify Mission Goals, Requirements and Constraints.....	22
3.2 Step 2: Define System Accidents or Unacceptable Losses.....	23
3.3 Step 3: Define High-level Hazards.....	24
3.4 Step 4: Define High-level Safety-related Constraints.....	24
3.5 Step 5: Identify Environment and Customer Constraints.....	25
3.6 Step 6: Perform High-level Functional Decomposition.....	26
3.7 Step 7: Design High-level System Control Structure.....	28
3.8 Step 8: Perform Preliminary Hazard Analysis.....	30
3.9 Step 9: Define System Element Specifications.....	32
3.9.1 Step 9.1: Define Goals, Assumptions, Requirements, and Constraints.....	35
3.9.2 Step 9.2: Develop Models of the System Under Control.....	36
3.9.3 Step 9.3: Define and Design Control System Operational Behavior.....	39
3.9.4 Step 9.4: Develop Formal Models of the Control System.....	42
3.9.5 Step 9.5: Continue to Perform STPA.....	43
3.9.6 Step 9.6: Iteratively Refine the System Design.....	44
3.10 Step 10: Perform Validation Tests.....	45
3.11 Step 11: Generate Designs and Software Code.....	45
4. Methodology Conclusion and Summary.....	45
4.1 Conclusion.....	45
4.2 Methodology Summary.....	46
5. References.....	48
6. Appendix I.....	49

List of Figures

Figure 1. Intent Specification Hierarchy.....	6
Figure 2. Martian Lander Intent Specification Partial Example.....	8
Figure 3. Generic System Control Structure.....	10
Figure 4. Generic STPA Low-level Process Control Loop.....	11
Figure 5. Inadequate Control Action Types.....	12
Figure 6. Inadequate Control Actions for the Martian Lander Example.....	12
Figure 7. Martian Lander Descent Process Control Loop.....	13
Figure 8. Control Flaw Taxonomy.....	13
Figure 9. Sources of Inadequate Control for the Martian Lander.....	14
Figure 11. Example <i>And/Or</i> Table from Camera State Transition Logic.....	17
Figure 12. SpecTRM-RL Camera Mode Control Process Model.....	18
Figure 13. Conceptual View of the State-Based Control Architecture.....	20
Figure 14. Relationships Between Elements of State Analysis and Functional Analysis.....	21
Figure 15. Level 1 Mission Goals.....	23
Figure 16. High-Level Requirement OPE Example.....	23
Figure 17. System Accidents.....	24
Figure 18. High-level Hazards.....	24
Figure 20. Environmental Description, Constraint, and Assumption.....	25
Figure 21. Customer-Derived Design Constraints.....	26
Figure 22. Customer-Derived Programmatic Constraints.....	26
Figure 23. A Portion of the Design Structure Matrix used for the OPE Functional Decomposition.....	27
Figure 24. Outer Planets Explorer Control Structure.....	30
Figure 25. Partial STPA Hazard Analysis.....	31
Figure 26. Partial Hazard Log Example.....	32
Figure 27. Level 1 OPE Example.....	33
Figure 28. Level 1 Constraints OPE Example.....	34
Figure 29. Design Decisions OPE Example.....	35
Figure 30. OPE Example of High Gain Antenna Control System Design.....	35
Figure 31. State Effects Diagram Example.....	37
Figure 32. Continuous State Effects Model for Joint i Angle j.....	38
Figure 33. Discrete State Effects Model for Joint i Motor j OpMode & Health.....	38
Figure 34. OPE Example of System Operational Behavior.....	40
Figure 35. State Effects Diagram HGA Boom angle joints.....	41
Figure 36. Example <i>And/Or</i> Table.....	42

Acknowledgements

The authors would like to thank Brad Burt, Karla Clark, Bharat Chudasama, Jeffrey Estefan, Cecilia Guiar, Peter Kahn, Steven Larsen, Cin-Young Lee, Robert Lock, Kenneth Meyer, David Nichols, Robert Rasmussen, Henry Stone, Robert Vargo, and Stephen Wall at the Jet Propulsion Laboratory for their support of this study.

This research was performed at the Massachusetts Institute of Technology (supported through JPL University Subcontract 1297013), and at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

1. Introduction

Conservative design has led to tremendous success in space exploration. As the complexity of spacecraft increases, however, and the use of new technology, particularly computers and software increases, serious problems and even mission losses have resulted. To handle this added complexity, new methodologies that design safety into spacecraft systems from the beginning of the design process and that use model-based techniques to find logical design errors early are needed. Note that safety is used here in the broad sense of reducing the risk of mission failure.¹

Typically, hazard analyses are performed later in system development, after many critical decisions have been made and cannot be undone without significant and perhaps unacceptable cost. In addition, fault tolerance and redundancy methods are based on assumptions that are true for hardware but are not necessarily effective for software.

The complexity of software is usually so great that it is difficult to find all the errors, particularly before system integration when fixing errors has tremendous penalties on schedule and budget. Model-based engineering techniques have the potential for performing much of system validation early in the development process before any software implementation and coding when errors can be fixed with fewer costs.

This report presents a Safety-Driven, Model-Based System Engineering Methodology that addresses these problems by enabling system engineers to design systems from a safety point-of-view, i.e., with hazard analysis folded into the nominal design process rather than conducted as a separate activity. This methodology integrates MIT's Systems-Theoretic Accident Model and Processes (STAMP), STAMP-Based Hazard Analysis (STPA), intent specifications (a structured, constraint-based system engineering specification framework), and JPL's State Analysis (a model-based systems engineering approach).

Readers already familiar with these various techniques can skip to Section 3.

2. Background

2.1 Intent Specifications

An intent specification is a specification and development framework supporting system design and other system engineering activities, intended to provide more readable and reviewable specifications. Intent specifications are based on research in human problem solving and on basic principles of system theory and system engineering [1].

¹ The NASA General Safety Program Requirements (Document NPR 8715.3B) defines Safety Critical as “describing any condition, event, operation, process, equipment, or system that could cause or lead to severe injury, major damage, or mission failure if performed or built improperly, or allowed to remain uncorrected.”

An intent specification differs from a standard system engineering specification primarily in its structure, which is designed to (1) facilitate the tracing of system-level requirements and design constraints down into detailed design and implementation and the documentation of design rationale, (2) assist in the assurance of various system properties (such as safety) in the initial design and implementation, and (3) reduce the costs of implementing changes and re-analysis when the system is changed, as it inevitably will be. Intent specifications contain the same information as would be found in traditional requirements artifacts; no extra specification is involved (assuming that projects produce the usual specifications). Intent specifications simply use a different structuring and linking of the information so that specifications provide more assistance in the system design and evolution process.

	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 Design	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 Specifications	Analysis Plans and Results
Level 4: Design Representation		HCI Design	Software and Hardware Design Specifications	Test Plans and Results
Level 5: Physical Representation		GUI and Physical Controls Designs	Software Code, Hardware Assembly Instructions	Test Plans and Results
Level 6: Operations	Audit Procedures	Operator Manuals Maintenance Training Materials	Error Reports, Change Requests, etc.	Performance Monitoring and Audits

Figure 1. Intent Specification Hierarchy

There are seven levels in an intent specification, as shown in Figure 1. Levels do not represent refinement, as in other commonly used hierarchical specification frameworks. Instead, each level of an intent specification represents a completely different model of the same system and supports a different type of reasoning about it: each model or level presents a complete view of the system from a different perspective. The model at each level is described in terms of a different set of attributes or language. Refinement and decomposition occurs within each level of the specification. In addition to intra-level refinement, the levels are organized in a “Means-Ends” hierarchy. In such a hierarchy, the information at a level acts as the goals (the ends) with

respect to the model at the next lower level [1]. In other words, the next lower level is where the means to the ends of the current level are implemented

The top level (level 0) provides a project management view and 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, requirements, design constraints, hazards, environmental assumptions, and system limitations.

Level 2, System Design, is the system engineering level and provides the structure and content needed for engineers to reason about the system in terms of the physical principles and laws upon which the system design is based. It documents the basic system-level design decisions made to satisfy the requirements and constraints at level 1.

The third level, or Blackbox Behavior level, enhances reasoning about the logical design of the system as a whole and the interaction among the components as well as the functional state without distractions from implementation issues. This level acts as an unambiguous interface between system engineering and component engineering to assist in communication and review of component blackbox behavioral requirements and to reason about the combined behavior of individual components using informal review, formal analysis, and simulation. The models at this level are formal and can be both executed and subjected to formal analysis (for example, completeness and consistency analyses).

The next two levels (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. The effort in this task has predominantly focused on levels 0-3 of the intent specification. Levels 4 and 5 represent the standard component documentation used on most any engineering project.

Figure 2 shows an example of intent specification traceability between Levels 0, 1, and 2 through partial specification of a spacecraft capable of landing on a planet surface. Traceability is captured through hyperlinks denoted by arrows and the specification item tag (for example, ↓Hazard.1). Traceability links denote different relationships between specifications based on their direction. An up arrow (↑) denotes that the current specification item is involved in the implementation of the intent of a specification item at a higher level in the “means-ends” hierarchy denoted by the tag after the arrow. A down arrow (↓) points to a specification item at a lower level in the “means-ends” hierarchy that is involved in the implementation of the intent of the current specification item. Left and right arrows denote relationships between specification items at the same level in the “means-ends” hierarchy that affect the items’ relationships to items on other levels. The direction of the arrow for this type of relationship depends on the physical location of the specification item in the intent specification document. A left arrow (←) points to a specification item at the same level that appears earlier in the specification than the current specification item. Conversely, a right arrow (→) points to another specification item at the same level that appears later in the current specification document. Thus, in Figure 2, the hazard (H1 at Level 1) will show an upwards link pointing to an accident (ACC1 at Level 0). This relationship

shows ‘why’ the hazard is of concern: it can lead to the accident ACC1 shown in Level 0. The accident has a downward arrow pointing to H1 showing ‘how’ the accident could occur. Similarly, H1 points across the level to a safety constraint (SC1) derived from the hazard. The safety constraint has downward pointing links to Level 2 where that safety constraint is enforced with system design decisions. Lastly, the relationship between the design decisions is captured through traces across Level 2.

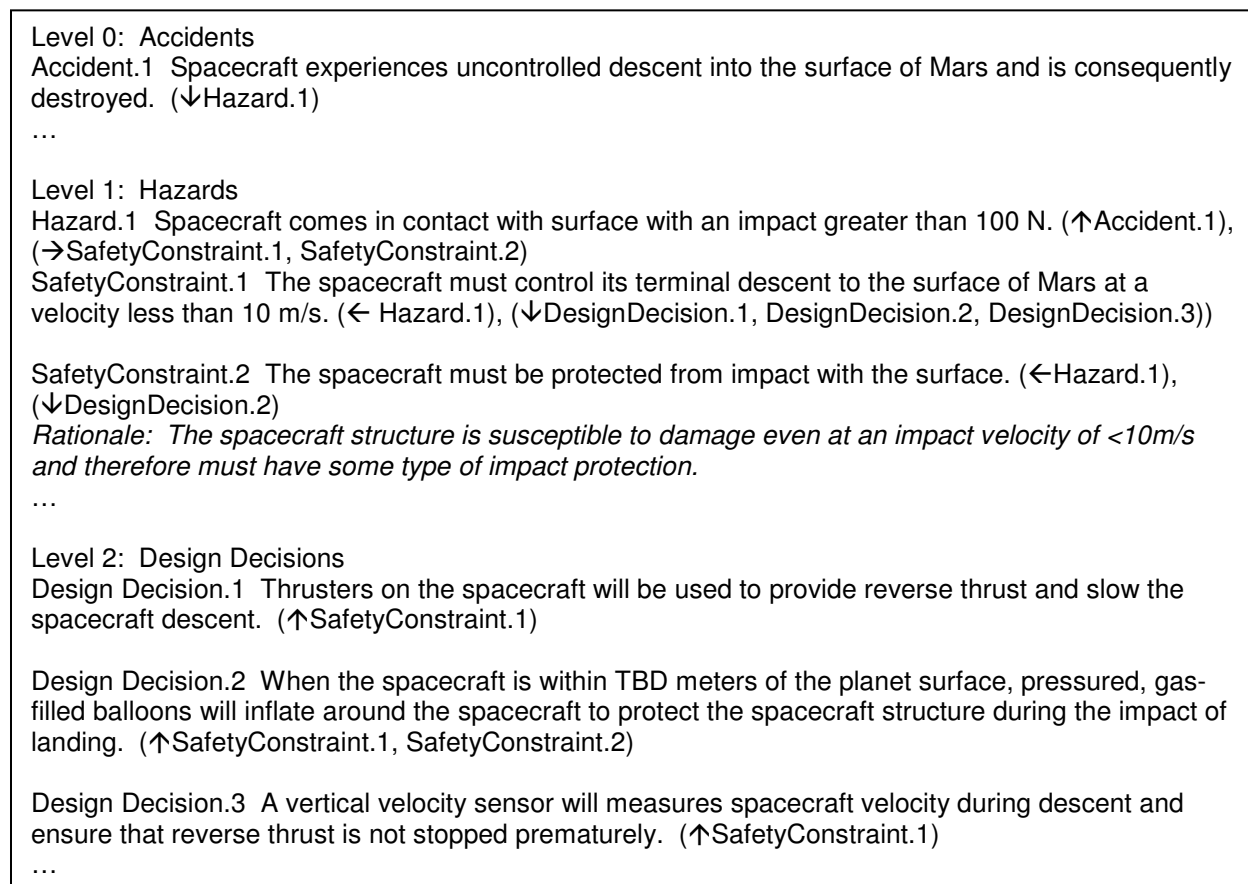


Figure 2. Martian Lander Intent Specification Partial Example

Intent information represents the design rationale upon which the specification is based. This design rationale is integrated directly into the specification. Each level also contains information about underlying assumptions upon which the requirements, 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. In the traditional system engineering specification approach, 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 been changed, it is very difficult and resource-intensive process to figure out which parts of the design used that assumption.

In summary, intent specifications allow a seamless transition from system to component (including software) specifications and the integration of formal and informal aspects of system and software development. The specification structure facilitates the tracing of system-level requirements and constraints into the design and the assurance of various system properties (such as safety) in the initial design and implementation; it also reduces the costs of implementing changes and re-analysis by providing traceability and rationale capture.

2.2 STAMP

STAMP, which stands for System-Theoretic Accident Modeling and Processes, is an accident causality model in which accidents are conceived as resulting not from component failures, but from inadequate control or inadequate enforcement of safety-related constraints on the design, development, and operation of the system [1], [2]. 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 system safety constraints.

In STAMP, 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 or controlled. The control processes that enforce the safety constraints must limit system behavior to the safe states implied by the safety constraints. Figure 3 shows a generic (example) control structure to enforce safety constraints. Each hierarchical level of the control structure represents a control process and control loop with actions and feedback. Two control structures are shown in Figure 3 – system development and system operations – both of which have different responsibilities with respect to enforcing safe system behavior.

STAMP treats a system not 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. 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 required for a spacecraft). 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 current state (the current values of the system variables), the ways the process can change state (the system dynamics), and the desired relationship among the system variables (the control laws). 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 [3].

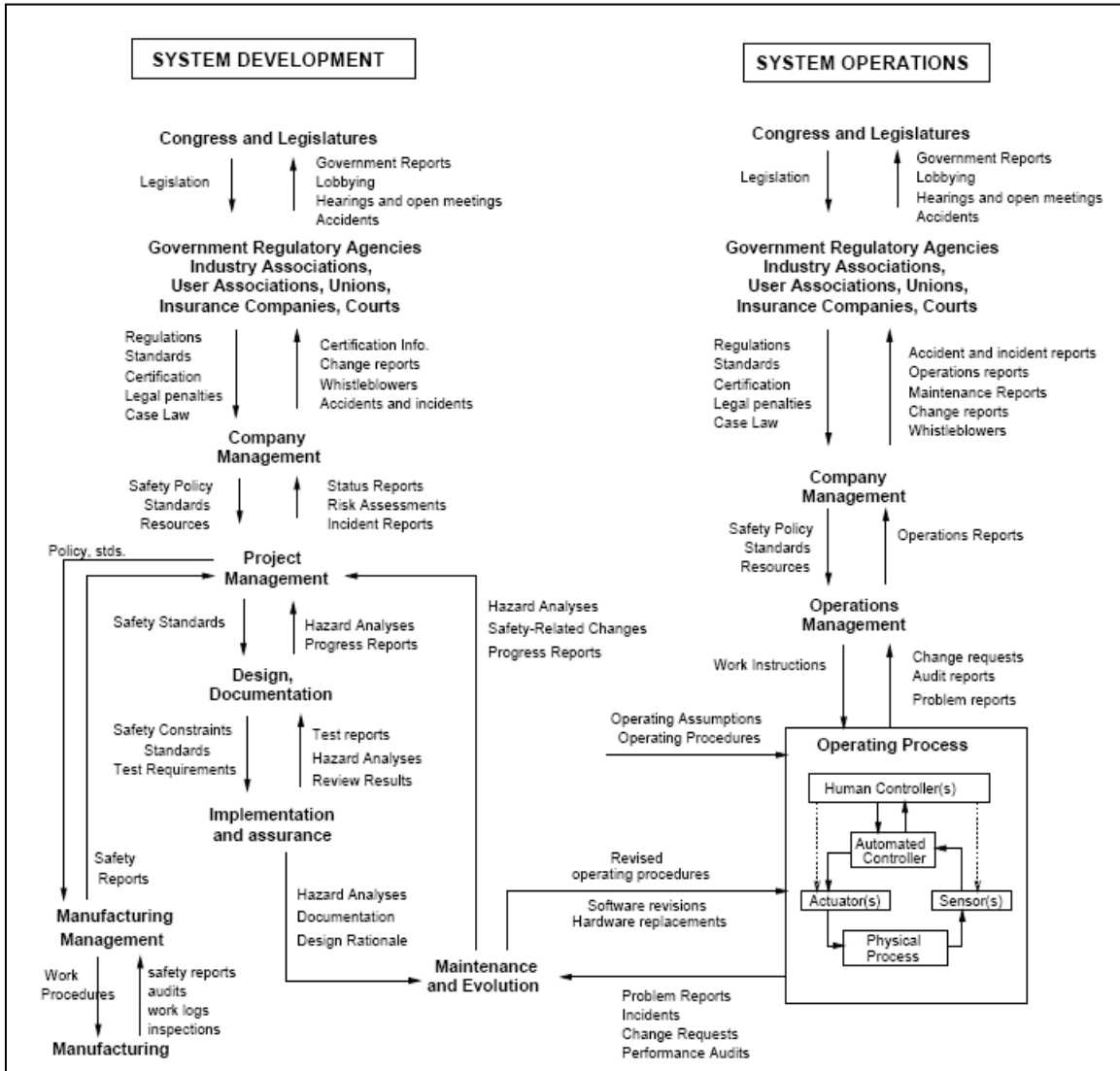


Figure 3. Generic System Control Structure

2.3 STPA

All hazard analysis techniques are based on a model of accident causation. STPA is a hazard analysis technique based on the STAMP model of accident causation. The objectives of STPA (STAMP-Based Hazard Analysis) are the same as that of a traditional hazard analysis (as described in [4]): (1) to identify the system hazards and the safety-related constraints necessary to ensure acceptable risk and (2) to accumulate information about how the safety constraints may be violated and use this information to eliminate, reduce, and control hazards in the system design and operation [5]. Although the first steps of 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.

Underlying the STPA process is the notion that hazards are eliminated or controlled through system design. Figure 4 presents a generic, low-level process control loop in STPA. As seen in the figure, the control input is a reference signal. The controller uses the control input in

conjunction with received measurements to generate commands. Continuing along the loop, the command is sent to the actuator, which implements the command through the arrow labeled U . The U vector refers to actions of the actuator that influence the controlled process. The control algorithm used by the controller is based on an internal process model of the controlled process. The controlled process, or plant, is subject to process inputs and disturbances. The process output may become input into another linked process control loop. The sensors measure the output resulting from the actuator's actions and disturbances, and generate measurements that are then fed into the estimator.

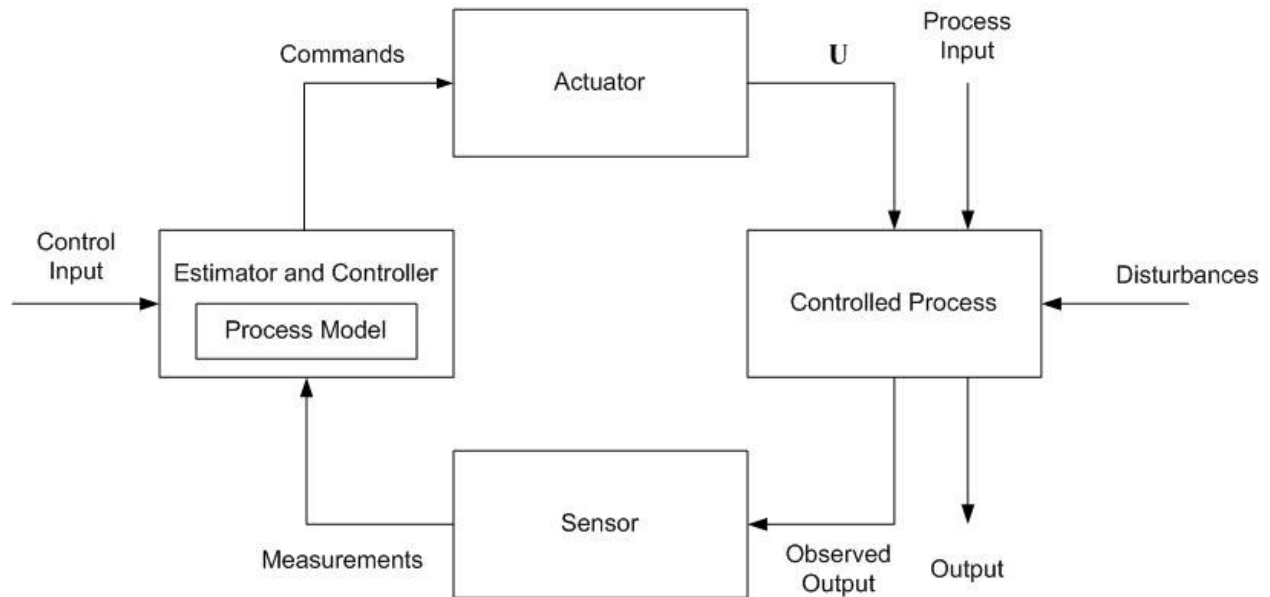


Figure 4. Generic STPA Low-level Process Control Loop

Depending on the particular system, the control input may be referred to as a goal, plan, sequence, directive or set point in spacecraft engineering parlance. The controller may send directives to a lower-level controller rather than an actuator in order to affect control on that process. Similarly, the lower-level control loop, rather than a sensor, may pass measurements or health to the higher-level control loop.

STAMP is based on the concept of controlling hazards rather than eliminating component failures (which are only one cause of hazards). When a safety constraint is violated, the hazard can occur and accidents can happen. For example, as illustrated in the Martian Lander example captured in Figure 2, if the physical process being controlled is the landing of a spacecraft, the relevant hazard is “Spacecraft comes in contact with surface with an impact greater than 100 N.” The spacecraft could be inadequately controlled, and the hazard state could occur if the landing controller directs or commands the thrusters to land such that the spacecraft impact is 120 N, for example. A control flaw, such as an incorrectly calibrated velocity sensor, could contribute to inadequate control of the landing process. The concepts of inadequate control and control flaws are discussed below.

Each hazard and related safety constraint is analyzed using STPA. Starting with a hazard and a safety constraint in place to control the hazard, *inadequate control actions* that could violate the safety constraint are identified. An inadequate control action is an action that leads to the violation of a safety constraint. The four types of inadequate control are shown in Figure 5.

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

Figure 5. Inadequate Control Action Types

Examples of each of the four types of inadequate control are shown for the Martian Lander example in Figure 6. The first step of an STPA hazard analysis is to list each combination of identified hazard and related safety constraint with the inadequate control actions that could lead to the violation of the safety constraint. The result of the first STPA step is shown in Figure 6 and is built from the partial intent specification of Figure 2.

SafetyConstraint.1: The spacecraft must control its terminal descent to the surface of Mars at a velocity less than 10 m/s.
ICA.1 Spacecraft descent control is not engaged.
ICA.2 Spacecraft descent control does not limit descent velocity to less than 10 m/s.
ICA.3 Spacecraft descent control is activated too late.
ICA.4 Spacecraft descent control is de-activated too soon.
...

Figure 6. Inadequate Control Actions for the Martian Lander Example

Using knowledge of the current system design, the next step in the STPA process is the identification of *control flaws* and *inadequate control executions*. Control flaws are the mechanisms that could lead to inadequate control due to errors in the control algorithm, poor understanding of the process, or poor coordination between multiple controllers. Control flaws are identified through inspecting the process control loop to determine how the system can produce an inadequate control action. Figure 7 depicts the process control loop for the Controlled Process of “descent” for the Martian Lander example. For example, the Inadequate Control Action of “Spacecraft descent control is not engaged” may result if the Control Input “Initiate Descent” is not received by the Martian Lander Estimator and Controller. Figure 8 contains a taxonomy to assist in the process of inspecting the control loop and identifying control flaws such as these. Early in the STPA process when most control loops are high-level, examination of the control loops are especially useful, as the number of loops and complexity of their interactions is tractable.

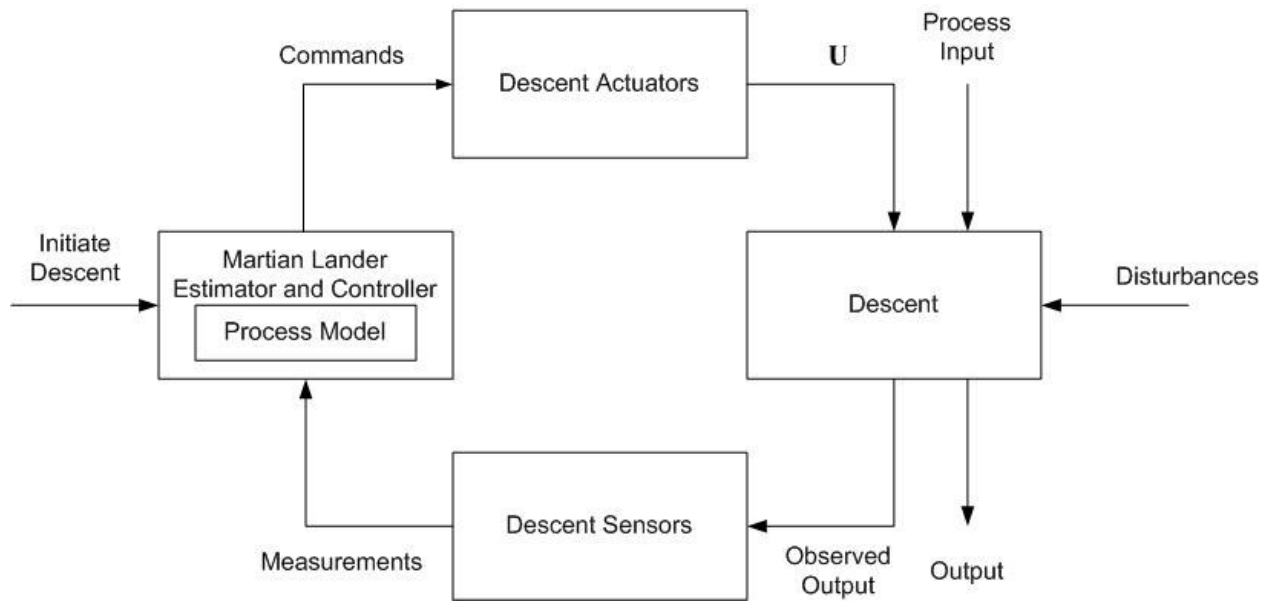


Figure 7. Martian Lander Descent Process Control Loop

1. Design of the control algorithm does not enforce constraints
 - Flaw(s) in creation process
 - Process changes without appropriate change in control algorithm (asynchronous evolution)
 - Incorrect modification or adaptation
2. Process models inconsistent, incomplete, or incorrect
 - Flaw(s) in creation process
 - Flaw(s) in updating process
 - Inadequate or missing feedback
 - Not provided in system design
 - Communication flaw
 - Time lag
 - Inadequate sensor operation
 - Time lags and measurement inaccuracies not accounted for
 - Expected process inputs are wrong or missing
 - Expected control inputs are wrong or missing
 - Disturbance model is wrong
 - Amplitude, frequency or period is out of range
 - Unidentified disturbance
3. Inadequate coordination among controllers and decision makers

Figure 8. Control Flaw Taxonomy

In addition to control flaws, inadequate control executions can also lead to inadequate control through failure of the mechanisms that actuate control (such as a motor failure) or transmission of control (communication line failure) or time lags (such as the sluggish response of the motor;

perhaps an indicator that the motor is soon to fail). In other words, inadequate control execution can occur when the process model is correct, and the correct control action is selected, but the control action is not successfully applied due to inadequate actuator or sensor operation, time lag, or a communication flaw. For example, as seen in Figure 7, if the descent sensors fail, the proper measurements will not be received by the Martian Lander Estimator and Controller, which may lead to the spacecraft not limiting the descent velocity to below 10 m/s.

As previously stated, the process control loops, control flow taxonomy, and identification of inadequate control executions are all used to guide the hazard analysis process and identify sources of inadequate control. The Martian Lander example is continued in Figure 9, which contains the sources of inadequate control for SafetyConstraint.1.

SafetyConstraint.1: The spacecraft must control its descent to the surface of Mars at a velocity less than 10 m/s.
ICA.5 Spacecraft descent control is not engaged.
CF.5.1 The spacecraft controller does not receive an initiate descent control input.
CF.5.2 The spacecraft controller does not command the descent actuators to activate.
CF.5.3 The descent actuators do not receive a command to activate.
ICE.1.1 The descent actuators do not activate.
ICA.6 Spacecraft descent control does not limit descent velocity to less than 10 m/s.
CF.6.1
ICA.7 Spacecraft descent control is activated too late.
ICA.8 Spacecraft descent control is de-activated too soon.

Figure 9. Sources of Inadequate Control for the Martian Lander

Once the sources of inadequate control have been identified, the associated hazard is mitigated through elimination, control, or damage reduction. The mitigation of hazards is accomplished through one of the following strategies:

1. Create a new safety constraint, modify the related safety constraint, or refine the related safety constraint to better enforce control.
2. Create new design or modify existing design to eliminate, prevent or mitigate the effect of the control flaw or inadequate control execution.
3. Accept the design as is and record the rationale for doing so.

Figure 10 illustrates the refinement of a safety-related design constraint and a design decision made to enforce the refined constraint.

SafetyConstraint.1 The spacecraft must reach the surface at a speed no greater than 10 m/s. (←H1) (↓DD1, DD2, DD3)

Safety Constraint 1.1 The spacecraft's estimated velocity must be accurate to within 0.2 m/s.(↓DD.4)

Design Decision.4 The spacecraft's velocity is calculated using measurement device with accuracy of +/- 0.05 m/s.(↑SC1.1)

Safety Constraint.2 The spacecraft must be protected from falling on a rock. (←H1) (↓DD2)

Rationale: The spacecraft structures is susceptible to high pressure impact, such as that encountered in a moderate impact (<10m/s) with a sharp pointy rock

Safety Constraint.2.2 Impact balloons must be capable of withstanding contact with rocks at speeds of up to 10 m/s. (↓DD.5)

Rationale: The reverse thrusters are only capable of landing the spacecraft at a speed in the range of 2-10 m/s. There must be a passive system in place to lessen the impact of landing.

Design Decision.5 The balloons must be inflated to a TBD pressure and made of a tear-resistant material.(↑SC2.2)

Figure 10 Refinement of Safety-related Design Constraint and related Design Decisions

SC 1.1 is a refinement of SC1, and was created to prevent ICA.1 and ICA.4 by trying to eliminate CF 1.1 and CF 4.1. In this case, one new safety constraint (and a new design decision to enforce the new safety constraint) helps to eliminate two inadequate control actions. The OPE examples described in section 3.8 show how the creation and refinement of safety constraints are recorded in the hazard log of the intent specification.

STPA should be performed iteratively and opportunistically. Engineers can either drill down into a particular hazard they wish to control or apply STPA more broadly across several hazards. In the early stages of intent specification creation, few design decisions have been made and control flaws and inadequate control executions may not yet be identified. However, performing STPA early will allow the results of the hazard analysis to inform the design process.

In essence, STPA starts with a hazard and its related requirements or constraints. The STPA taxonomy is used to identify inadequate control actions and the control flaws and/or inadequate control executions that lead to inadequate control actions. From there, engineers create new constraints or refine the existing constraints and create new design or modify the existing design until all hazards are mitigated, eliminated, or controlled. Engineering judgment is used to determine when the design is “safe and complete enough.”

The results of STPA (hazard analysis) are documented in the hazard log in level 1 of the intent specification.

2.4 SpecTRM and SpecTRM-RL

In this task, we have used a commercial system engineering toolset, SpecTRM (Specification Tools and Requirements Methodology) that provides support for capturing intent specifications [3]. SpecTRM focuses on the early stages of system development, where the foundation is set for later implementation, operations, and maintenance activities. The SpecTRM toolset includes support for requirements development and management, hazard analysis, requirements tracing (within the document and to external documents), recording of design rationale, and modeling and analysis of blackbox logic requirements. SpecTRM’s formal modeling language is discussed in the next section.

SpecTRM includes a formal modeling language, SpecTRM-RL (SpecTRM Requirements Language) to model the system blackbox behavior described at level 3 of the intent specification. SpecTRM-RL has a formal foundation so it can be executed and subjected to formal analysis while still being readable with minimal training and expertise in discrete math. In addition, the models are analyzable, and tools have been developed to check for completeness, consistency, and robustness.

SpecTRM-RL was designed to satisfy two objectives: to be easily readable enough to serve as part of the official specification of the blackbox behavioral requirements and, at the same time, to have an underlying formal model that can be executed and subjected to mathematical analysis. This underlying formal model, based on a Mealy automaton, which we call the requirements state machine (RSM), is very low-level and not appropriate as a specification language for complex systems. Instead, SpecTRM-RL acts as the specification language (or visualization of the underlying model) that overlays the low-level model. As long as the mapping from SpecTRM-RL to the RSM is unambiguous and well-defined, formal analysis is possible on both the underlying RSM formal model as well as the higher-level SpecTRM-RL specification itself.

The conditions under which an output is triggered (sent) can be specified by a predicate logic statement over the various states, variables, and modes in the specification. In our experience in specifying complex systems, however, we found that the triggering conditions required to accurately capture the requirements are often extremely complex. We also found propositional logic notation did not scale well to complex expressions in terms of readability and error-proneness. To overcome this problem, we developed a tabular representation of disjunctive normal form (DNF) that we call *and/or* tables. Figure 11 shows an example of operational mode control logic for a camera, expressed as an *and/or* table.

Figure 11 illustrates an example of a SpecTRM-RL *and/or* table specification; it defines the criteria for the transition of a spacecraft camera state into an *Idle* mode. 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 (in this case, *Idle*) if any of the columns evaluate to true. If one of the columns evaluates to *true*, then the entire table evaluates to *true*. 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. Underlined variables represent

hyperlinks. For example, clicking on [Camera-State](#) would show how the ‘Camera-State’ state variable is defined in the intent specification.

= Idle			
Previous Value of Camera-State in state Off	*	F	F
Turn-On-Camera-Command was Received	T	*	*
Previous Value of Power-Bus-State in state Powered	T	T	T
Time Since Camera-State Last Entered Ready >= 10 seconds	F	T	F
Previous Value of Camera-State in state Ready	F	T	T
Go-Idle-Camera-Command was Received	*	F	T

Figure 11. Example *And/Or* Table from Camera State Transition Logic

In the example described in Figure 11 the camera is only able to transition into *Idle* mode if: 1) the Camera was previously *Off*, the ‘Turn-On-Camera-Command’ was received, and the power bus is delivering power to the camera; or 2) the camera has been in *Ready* mode for at least 10 seconds and the power bus is delivering power to the camera; or 3) the camera has been in *Ready* mode for less than 10 seconds, the ‘Go-Idle-Camera-Command’ was received, and the power bus is delivering power to the camera.

The *and/or* tables used in the blackbox models describe the conditions for transitioning between states and the values of inputs and outputs. Visualizations for black box models can also be created in SpecTRM as shown Figure 12. A process model visualization shows all of the inputs, outputs, control modes, inferred state variables, and other controllers or devices necessary for control of the camera. Each state variable, input, and output has a model described in part by an *and/or* table in level 3 of the intent specification. Level 3 formally defines the control system behavior and includes the transitions between values of an inferred state variable and control modes. It also includes timing constraints, descriptions, state variable macros, and functions for the value calculation of continuous state variables.

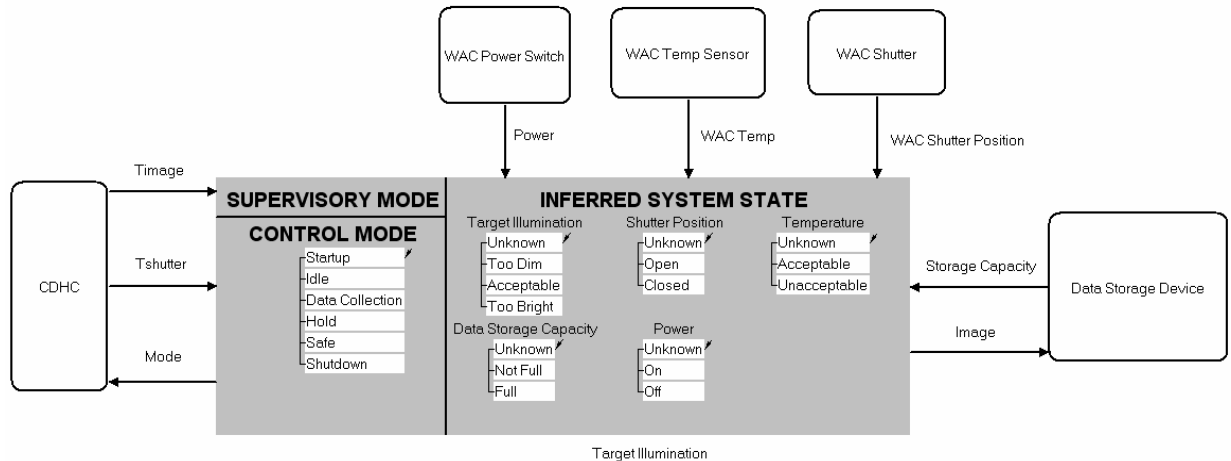


Figure 12. SpecTRM-RL Camera Mode Control Process Model

2.5 State Analysis

A novel model-based systems engineering methodology, called *State Analysis*, has been developed to complement the traditional functional decomposition approach and better address the challenges of our increasingly complex systems [6]. It provides a methodical and rigorous approach for:

1. Modeling behavior in terms of system state variables² and the relationships between them (*state-based behavioral modeling*);
2. Capturing mission objectives in detailed scenarios motivated by operator intent (*goal-directed operations engineering*); and
3. Describing the methods by which objectives will be achieved (*state-based software design*).

For the purposes of this task, we have focused primarily on the state-based behavioral modeling and state-based software design aspects of State Analysis³. The state-based behavioral modeling aspect provides an iterative process for *identifying state variables of the system under control* and for *incrementally constructing the model of the physical behavior of the system under control* associated with these state variables. The steps in this process, which is referred to as “State Discovery”, include the following:

1. Identify needs – define the high-level objectives for controlling the system.
2. Identify state variables that capture what needs to be controlled in order to meet the objectives, and define their representation.

² State is the momentary condition of an evolving system, and models describe how state evolves. The state of a system and our knowledge of that state are not the same thing. The real state may be arbitrarily complex, but our knowledge of it is generally captured in simpler abstractions that we find useful and sufficient to characterize the system state for our purposes. We call these abstractions “state variables.” The known state of a system is the value of its state variables at the time of interest. Together, state and models supply what is needed to operate a system, predict future state, control toward a desired state, and assess performance. More detail on the representation of state knowledge in State Analysis has been previously published. [6]

³ Future work is needed to address the goal-based operations aspect of State Analysis and how that can be incorporated with the intent specification approach.

3. Define state effects models for the identified state variables – these may uncover additional state variables that affect the identified state variables.
4. Identify measurements needed to estimate the state variables, and define their representation.
5. Define measurement models for the identified measurements – these may uncover additional state variables.
6. Identify commands needed to control the state variables, and define their representation.
7. Define command models for the identified commands – these may uncover additional state variables.
8. Repeat steps 2-7 on all newly discovered state variables, until all relevant variables and effects are accounted for.
9. Return to step 1, this time to identify additional objectives, and proceed with additional iterations of the process until the scope of the mission has been covered.

This modeling process can be used as part of a broader, iterative incremental system and software development process, in which cycles of the modeling process can be interwoven with concurrent cycles of software implementation.

The state-based software design aspect of State Analysis involves using these behavioral models to develop requirements and specify algorithm-level designs for the fundamental control system functions required to achieve the specified objectives. These control system functions map to the state-based control architecture, shown in Figure 13. Each state variable, estimator, controller, and hardware adapter represents a component of the control system. State Analysis defines an interconnection topology among the components in each control loop according to canonical patterns and standard interfaces; furthermore, the causal effects between state variables captured in the behavioral models can be used to specify appropriate interfaces between the corresponding control loops.

There are significant software assurance benefits to using a development methodology and architecture that share the same basic structure, namely an unprecedented level of coordination and control of the systems and software development processes. For example, requirements are cleanly partitioned and traceable directly to implementation, making it easy to track and manage each step in the development process. Verification and validation exploits the same explicit structure, as well as the objective specification of each system element and the overt declaration of success criteria at all levels of operation.

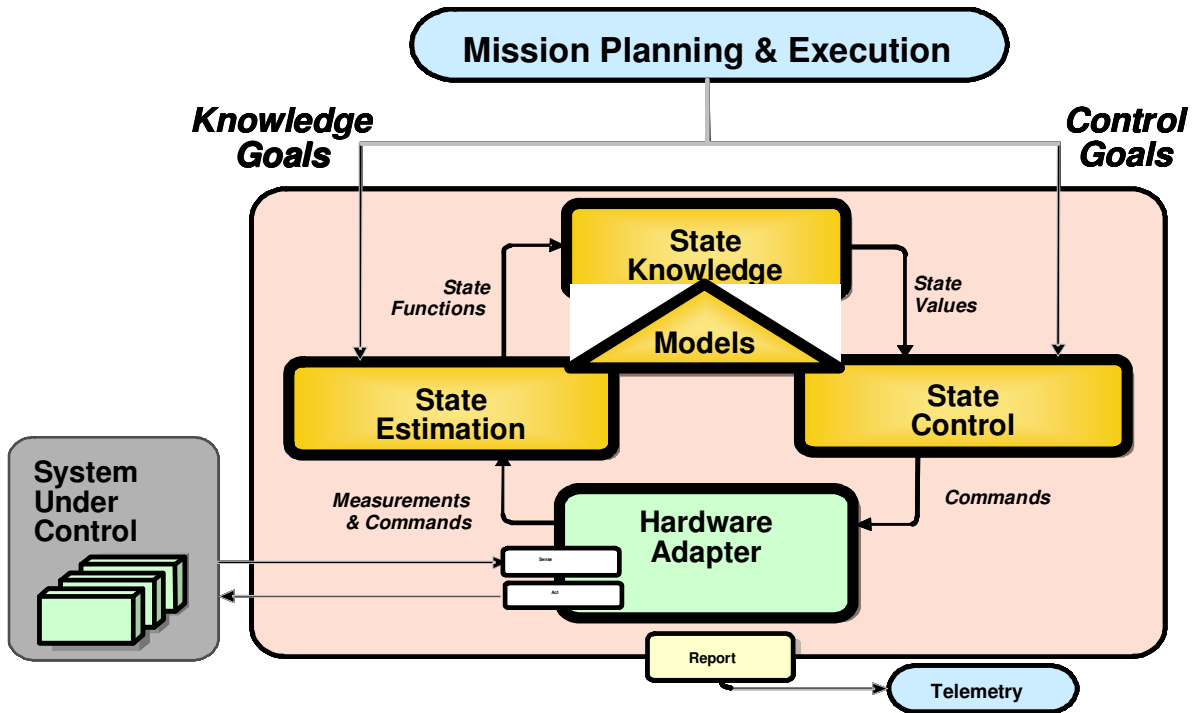


Figure 13. Conceptual View of the State-Based Control Architecture

State Analysis produces and compiles information that is traditionally documented in a variety of systems engineering artifacts, including Hardware Functional Requirements, Failure Modes & Effects Analyses, Command Dictionaries, Telemetry Dictionaries and Hardware-Software Interface Control Documents. Previous work has defined the relationships between State Analysis and a traditional Functional Analysis approach by identifying the appropriate set of relationships between elements of the State Analysis schema and elements of the Functional Analysis schema [7]. Figure 14(a) shows the relevant elements of the Functional Analysis schema, along with the relationships between these elements. As shown in Figure 14(b), this schema can be augmented with the following State Analysis elements: state variables, commands and measurements (goals are also shown in the diagram, but, as discussed above, the goal-based operations engineering aspect of state analysis was outside the scope of this task). As shown in the figure, relationships were defined between these Functional and State Analysis elements, (for example, functions act on state variables by producing commands that affect them, based on data provided to them through measurements).

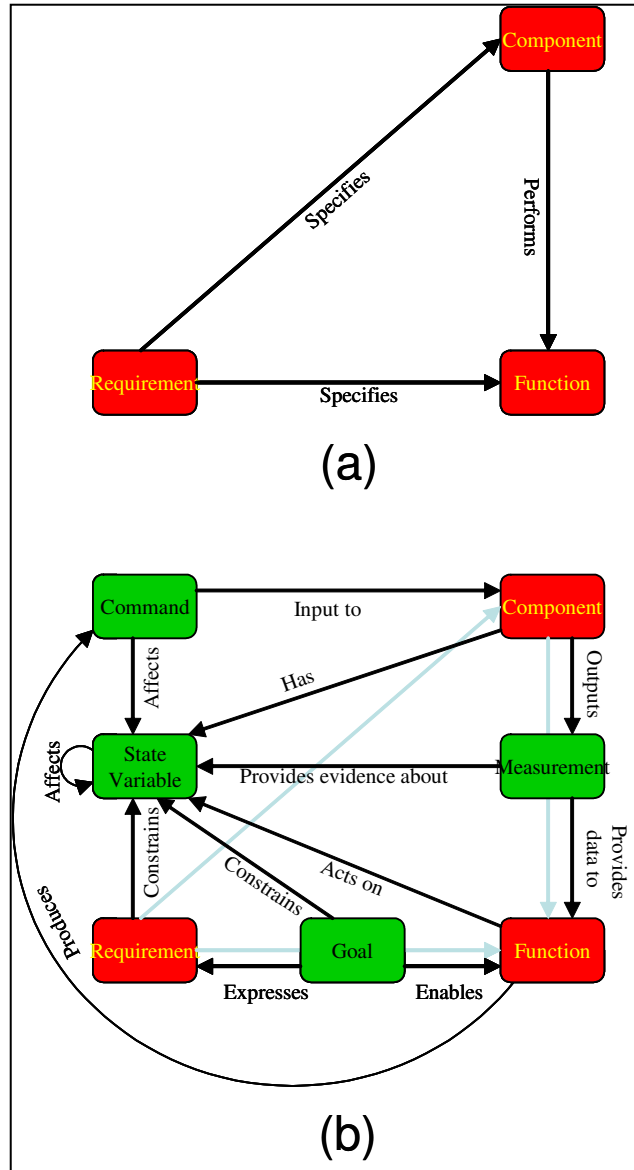


Figure 14. Relationships Between Elements of State Analysis and Functional Analysis

Part (a) of the figure illustrates functional analysis elements and their relationships.

Part (b) of the figure illustrates the elements and relationships of state analysis synthesized with functional analysis.

In summary, the State Analysis approach is novel and unique in three ways:

1. It is based on a state-based control architecture (see Figure 13) and leverages a core set of underlying architectural principles.
2. When coupled with a state-based control architecture, State Analysis provides a common vocabulary for systems and software engineers to communicate, and a common set of architectural elements such that the gap between the requirements provided by the systems engineer and the software developed by software engineer can be minimized. More specifically, it defines *direct mappings* from the system requirements (expressed in

the form of behavioral models) to software specifications, and from these software specifications to implemented software artifacts.

3. It considers the full breadth of system state variables (e.g., dynamics, environmental states, device status and health, parameters, resources, etc...) and allows for documentation of models using whatever representation is most appropriate (differential equations, state charts, tables, pseudo-code, textual descriptions, etc...).

2.6 Summary

Each of the approaches discussed above are intended to produce safety-driven design. STAMP and STPA provide an innovative new way to perform safety-driven design, while intent specifications and SpecTRM provide a unique way of capturing the system design specification in a way that allows for more transparency and fewer errors of omission or loss of information. State Analysis provides a way to explicitly model the system under control and uses those models to inform the control system design. The integration of these methods is particularly valuable because State Analysis benefits from using constraint-based requirements as a rigorous basis for informing the state discovery and modeling process. The MIT products benefit by having a modeling methodology that is natural for spacecraft engineers and produces models of the system that map more directly into the design of the control software.

STAMP, STPA, intent specifications and State Analysis approach the design of systems by focusing on different perspectives, but they have obvious complementary aspects. This task capitalized on the strengths of each to define a Safety-driven Model-based System Engineering methodology.

3. System Engineering Methodology

The integration of STAMP and STPA, intent specifications, and State Analysis has the potential for powerful synergy as a systems engineering methodology. As a safety-driven design methodology, it assists engineers in generating requirements aimed at hazard elimination and mitigation *concurrently* with generating functional requirements and spacecraft specification. Safety is designed into the spacecraft from the beginning of the design process rather than adding on fault protection after the fact.

The integrated methodology presented here in the context of an example of a hypothetical NASA mission to explore an icy moon of an outer planet in our solar system. Henceforth, this mission will be referred to as the Outer Planets Explorer Mission (OPE). The OPE example intent specification is included in Part 2 of this report.

3.1 Step 1: Identify Mission Goals, Requirements and Constraints

The starting point for the methodology is the definition of *mission goals* that we want or require the mission to achieve, *requirements* that the mission must meet to achieve its mission goals, and *constraints* on the mission that must not be violated in the efforts to achieve the mission goals. Mission goals, requirements and constraints can be drawn from or inspired by a number of

standard information sources such as: prior architecture studies, reused mission requirements, governmental mandates, corporate policies, laws, and standard system safety practices.

For this project, the science goals shown in the OPE examples were derived from Karla Clark’s paper describing a proposed Europa orbiter mission [8]. The goals listed in this paper are generalized below and shown in Figure 15. The example contains mission goals with links across level 1 pointing to high-level requirements. Figure 16 shows an example of a high-level requirement derived from the mission goals, corresponding to a high-level requirement called out in Clark’s paper. Mission goals, requirements, and constraints are the usual starting point for the state-based behavioral modeling aspect of State Analysis, as well (recall section 2.5). At this point in the methodology, any state variables of the system under control that are relevant to these mission goals and constraints should be documented in level 2 of the intent specification (e.g., states of key science data products).

- | |
|---|
| <p>G1. Characterize the presence of a subsurface ocean on an icy moon of an outer planet (Clark, 2007). (↑ACC4, ACC5), (→HLR3, HLR4), (↓SV-81)</p> <p>G2. Characterize the three-dimensional configuration of the icy crust of the icy moon of an outer planet, including possible zones of liquid (Clark, 2007). (↑ACC4, ACC5), (→HLR1, HLR2, HLR3)</p> <p>G3. Map organic and inorganic surface compositions of the icy moon of an outer planet, especially as related to astrobiology (Clark, 2007). (↑ACC4, ACC5), (→HLR2, HLR3)</p> <p>G4. Characterize surface features of the icy moon of an outer planet and identify candidate sites for future exploration (Clark, 2007). (↑ACC4, ACC5), (→HLR1, HLR2, HLR3)</p> <p>...</p> |
|---|

Figure 15. Level 1 Mission Goals

<p>HLR3. The mission shall image TBD% of the surface of the icy moon of the outer planet in spectra other than visual and infrared, to a resolution of TBD. (←G1, G2, G3, G4, G6), (→S/C-G1, S/C-G2, S/C-R1, S/C-R2), (↓2.1, SV-1, SV-101, SV-102)</p> <p><i>Rationale: The other bands of the spectrum provide insights into the chemical composition of the icy moon</i></p>
--

Figure 16. High-Level Requirement OPE Example

3.2 Step 2: Define System Accidents or Unacceptable Losses

To derive the safety requirements and design constraints, the engineer first defines system accidents or unacceptable losses. These losses may include loss of life, mission, or damage to the environment. Figure 17 contains accident definitions for the OPE example. System accidents are documented in level 0 of the intent specification.

<p>ACC1. Humans and/or human assets on earth are killed/damaged. (↓PC1, H5, SV-77, SV-78, SV-79)</p> <p>ACC2. Humans and/or human assets off of the earth are killed/damaged. (↓PC1, H6, SV-77, SV-78, SV-79)</p> <p>ACC3. Organisms on any of the moons of the outer planet (if they exist) are killed or mutated by biological agents of Earth Origin. (↓H4)</p> <p>ACC4. The scientific data corresponding to the mission goals are not collected. (↓G1, G2, G3, G4, G5, G6, G7, H1, SV-80)</p> <p>ACC5. The scientific data corresponding to the mission goals is rendered unusable (i.e., deleted and/or corrupted) before it can be fully investigated. (↓G1, G2, G3, G4, G5, G6, G7, H2, H3, SV-80)</p> <p>...</p>

Figure 17. System Accidents

3.3 Step 3: Define High-level Hazards

Next, the engineer defines the high-level hazards that could lead to system accidents, prevent mission goals and requirements from being met, or violate the mission constraints. Figure 18 lists examples of the high-level hazards that were derived using the OPE mission goals in combination with standard system safety engineering principles and analysis of the accidents identified above. System-level hazards are documented in level 1 of the intent specification. Then, from these high-level hazards, the high-level safety constraints are defined (see Step 4).

<p>H1. Inability of Mission to collect data. (↑ACC4), (↓SV-85)</p> <p>H2. Inability of Mission to return collected data. (↑ACC5), (↓SV-86)</p> <p>H3. Inability of Mission scientific investigators to use returned data. (↑ACC5), (↓SV-87, SV-88)</p> <p>...</p>

Figure 18. High-level Hazards

3.4 Step 4: Define High-level Safety-related Constraints

The safety constraints are requirements that eliminate or mitigate the hazard. If the hazard is of the form “Hazardous state occurs,” it involves a simple but important translation from the hazard into an engineering goal. For example, if the hazard is “Subway train leaves the platform with the doors open,” the corresponding safety constraint could be “Subway train must not be capable of leaving with the train door open.” An example from the OPE is show below in Figure 19. High-level safety constraints are documented in level 1 of the intent specification.

<p>H1. Inability of Mission to collect data. (↑ACC4), (↓SV-85) SC1. The mission must have the necessary functionality for data acquisition at the required times. (←H1), (→MOC-G1, MOC-G2, MOC-G3, MOC-G4), (↓2.1, 2.2, 2.4, SV-85)</p> <p>H2. Inability of Mission to return collected data. (↑ACC5), (↓SV-86) SC2. The mission must be able to return data at the required times. (←H2), (→MOC-G1, MOC-G2, MOC-G3, MOC-G4), (↓2.1, 2.3, 2.4, 2.5, SV-86)</p> <p>H3. Inability of Mission scientific investigators to use returned data. (↑ACC5), (↓SV-87, SV-88) SC3. Mission scientific investigators must be able to use the data from the mission at the required times. (←H3), (↓2.1, 2.3, 2.4, 2.5, SV-87, SV-88), (→MOC-G1, MOC-G2, MOC-G3, MOC-G4)</p> <p>...</p>

Figure 19 High-level Hazards and Safety Constraints

3.5 Step 5: Identify Environment and Customer Constraints

The next step in the methodology is to identify:

- environmental constraints and environmental assumptions,
- customer-derived system design constraints, and
- customer programmatic constraints (e.g., budgets, etc.).

Environmental description, constraints, and assumptions describe and constrain the environment of the system and are design independent. Given a goal of the mission to explore an icy moon of an outer planet, information regarding lighting conditions, radiation environment, atmospheric pressure, gravity, etc. would be documented in this section. In addition, this kind of information would be documented for other environments that the mission will encounter before and after the primary mission at the icy moon. An example of an environmental description, constraint, and assumption can be found in Figure 20.

<p>Magnetic Flux: The magnetic field surrounding the icy outer planet moon to be studied is poorly understood and different from the magnetic field in Low Earth Orbit.</p> <p>EC.1. The mission elements must withstand a solar flux of TBD Watts/m², the solar flux in the orbit of the outer planet moon. (↓SV-1, SV-83)</p> <p>EA.1. The translation and rotation of the moon of study with respect to the Sun and relevant outer planet will be relatively stable over the mission and thus predictable.</p>

Figure 20. Environmental Description, Constraint, and Assumption

Environmental constraints are documented in level 1 of the intent specification. At this point, state variables of the system under control that are relevant to the environmental constraints and assumptions are documented in level 2 of the Intent Spec (e.g., surface temperature of the icy moon; ephemeris state of the sun, the outer planet, and the icy moon). State effects models capturing the physics behind these environmental constraints and assumptions should also be documented.

Customer-derived system design constraints are constraints on the design of the system that are technical in nature. Typically, they involve how the system must interact with existing resources, engineering mandates, or initiatives the customer wishes to implement. Figure 21 shows an example of customer-derived system design constraints. Customer-derived system design constraints are documented in level 1 of the intent specification.

<p>DC1. The mission must be carried out with existing technologies and space exploration infrastructures as needed (i.e., technologies rated at Technology Readiness Level TBD as defined by NASA). (↓2.1) <i>Rationale: While technology development is expected to be an ongoing activity of NASA, it is assumed to be beyond the mandate of the mission.</i></p> <p>DC1.1. The mission must utilize the Deep Space Network (DSN) for any communications beyond earth orbit. (→S/C-R5, S/C-R6), (↓2.3) <i>Rationale: The DSN is a proven resource for ground communication with spacecraft operating beyond earth orbit. The capabilities that it provides were created at great expense and funding will not be provided to duplicate them.</i></p> <p>...</p>

Figure 21. Customer-Derived Design Constraints

Customer programmatic constraints are those programmatic decisions that will influence the design of the entire system. Conflicts between safety and customer programmatic constraints must be investigated, so it is critical to document programmatic constraints in the intent specification. Figure 22 shows an example of customer programmatic constraints. Customer programmatic constraints are documented in level 0 of the intent specification.

<p>PC1. Whenever the mission utilizes space exploration infrastructure that other space exploration missions make use of, it must do so without directly interfering with the successful completion of those missions. (↑ACC1, ACC2, ACC7), (→S/C-C3) <i>Rationale: It is possible for this mission to interfere with the completion of other missions through denying the other mission access to the space exploration infrastructure (e.g., over-use of limited DSN resources, damage to launch pad during this mission results another mission missing its launch window, etc.).</i></p>
--

Figure 22. Customer-Derived Programmatic Constraints

3.6 Step 6: Perform High-level Functional Decomposition

After the goals and external constraints are defined and documented, the next step in the methodology is to perform a high-level functional decomposition to define the system functions and assign functions to high-level system components.

The assignment of functions to components can be viewed as system-level design decisions and should involve an analysis to assist in making safety-related decisions. System-level design decisions have a huge impact on safety. For example, if, for business reasons, management decides to use radio-isotopic thermoelectric generators (RTGs), this decision will introduce a hazard of contaminating Earth by inadequately preventing the dispersion of radioactive materials into Earth's atmosphere. In order to incorporate safety from the beginning we recommend

following a risk-based architectural approach, such as described in the paper (included in an appendix to this report) “Incorporating Safety in Early System Architecture Trade Studies” [9].

While a functional analysis can be performed in various ways, our example uses Design-Structure Matrices (DSMs)⁴ to document and aid in the analysis [10]. As functions necessary to meet the system’s requirements and constraints were identified, physical and informational interactions between these functions (e.g., energy exchanges, information exchanges, and/or material exchanges) are also identified and recorded in the DSMs. Figure 23 below shows a portion of the DSM used in the initial functional decomposition of the spacecraft for the OPE mission.

Name		1	2	3	4	5	6	7	8	9	10	11	12	13	14	...
Spacecraft MOC Directive Execution	1	1	1	1						1						
Spacecraft MOC Directive Storage	2		2							1						
Spacecraft H&S Data Evaluation	3			3	1		1			1						
Spacecraft H&S Data Collection	4	1			4					1						
Spacecraft H&S Data Storage	5	1		1	1	5				1						
Spacecraft H&S Data Retrieval from Storage	6	1				1	6			1						
Spacecraft H&S Data Packetization	7	1					1	7		1						
Spacecraft Power Generation	8	1							8							
Spacecraft Power Distribution	9	1							1	9						
Spacecraft Translation	10	1							1	10	1					
Spacecraft Pointing	11	1							1	1	11	1	1			
Spacecraft Antenna Deployment	12	1							1				12			
Spacecraft Antenna Pointing	13	1							1	1	1	1	13			
Spacecraft Science Data Collection	14	1							1	1	1					14
...																

Key	
Command & Data Handling (C&DH)	
Electrical Power (EP)	
Attitude & Articulation Control (A&AC)	
Science Data Collection (SCI)	

Figure 23. A Portion of the Design Structure Matrix used for the OPE Functional Decomposition

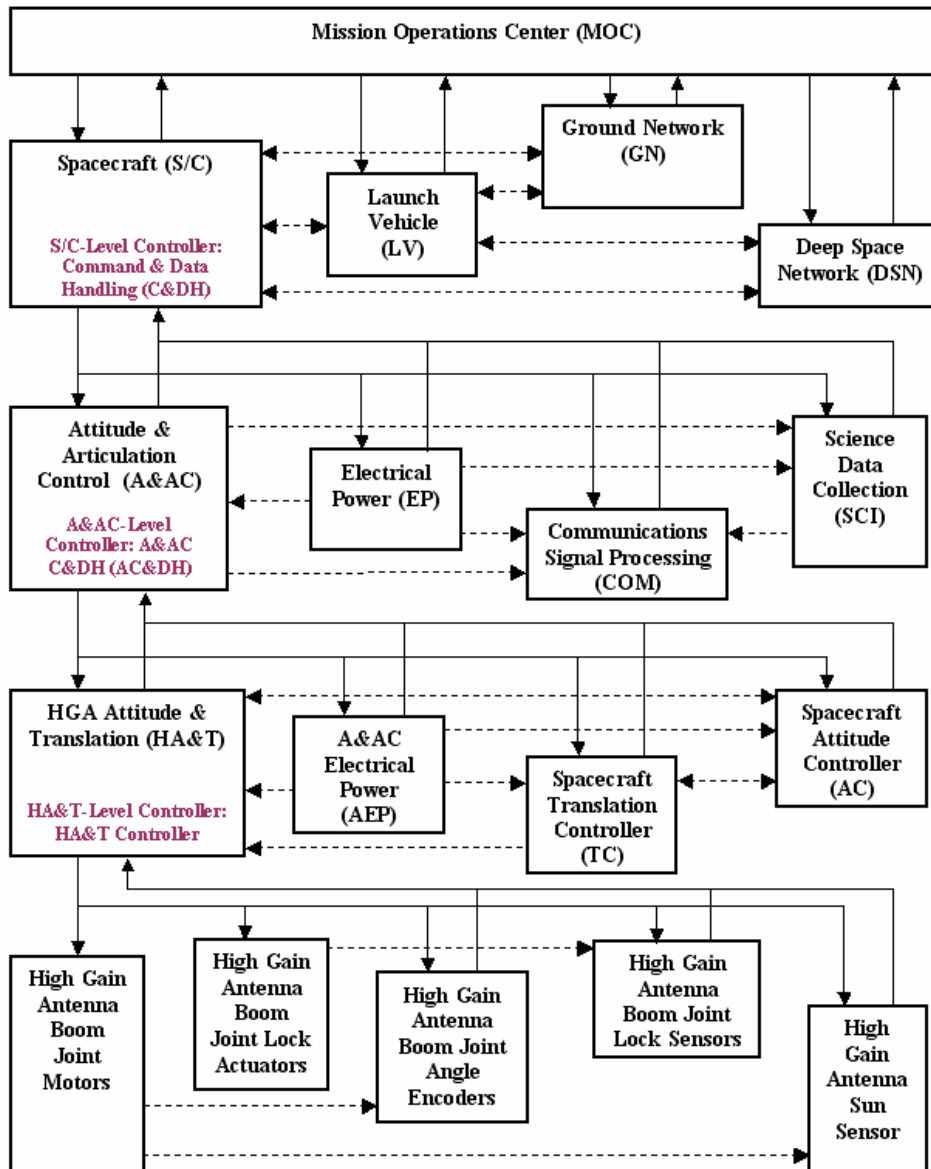
⁴ Design Structure Matrices are also referred to as N-Square Diagrams, Dependency Structure Matrices, Incidence Matrices, Dependency Maps, Interaction Matrices, Design Precedence Matrices, etc. in the literature.

The rows and columns of the matrix represent the functions to be performed. Each function is assigned a number that appears in the row next to the function name, the column corresponding to the function, and the diagonal where the function's row and column intersect. The "1's" in the rows of the matrix represent physical and informational interactions between the functions represented by the row and columns. A "1" in a row indicates that the function represented by that row receives a physical or informational input from the corresponding column function (e.g., the spacecraft translation function in Row 10 requires execution of a MOC Directive, electrical power, and pointing of the spacecraft so that the appropriate thrust vector can be obtained). Note that all physical interactions were considered equal in this analysis; it is possible to weight the interactions based on a number of criteria (e.g., scale, complexity, etc.), however, for the purpose of this analysis, such weighting was deemed beyond the scope of this study and left to future work.

Physical and informational coupling across functional components of complex systems has often been cited as a major factor in the accidents that occur in complex systems [11]. Therefore, once all functions and physical interactions identifiable at this point of the analysis are recorded in the DSM, the functions are clustered by manually reordering the matrix rows and columns in order to assign the functions to individual functional elements in a manner that minimizes coupling across the functional elements. The major functional elements defined in the DSM of Figure 23 are denoted by distinct colors that are explained in the key for the matrix. Once these functional elements are identified, requirements and constraints are derived for them. Section 3.9.6 discusses how lower-level functional elements are iteratively identified.

3.7 Step 7: Design High-level System Control Structure

As can be seen in Figure 24, it is not always possible to define functional elements of a system such that no physical and informational interactions across elements are necessary. To address this issue, we turn to the control structure. At each level of functional decomposition, each functional element is assigned responsibility for the control of the functional interactions within the element while one hierarchically superior element is assigned responsibility for control of the interactions across elements. Note that this is not a description of the software architecture, but a representation of the functions the system must perform and how the functions are related to each other. Mapping of this structure into, for example, a state-based control architecture as shown in Figure 13 would be appropriate, but was out of scope for this task. In the example, interactions between spacecraft functional elements are controlled by the spacecraft command and data handling functional element (C&DH) while interactions between functional elements of the Attitude and Articulation Control (A&AC) functional element are controlled by the A&AC command and data handling functional element (AC&DH). The system structure is provided in Figure 24 below.





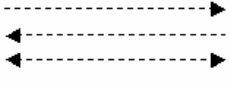
Control Structure Legend	
Diagram Item:	Description:
	Control in the form of Directive(s) or Command(s)
	Control Feedback in the form of State Information or Sensor Measurements
	Physical and Informational Interaction other than Control and Control Feedback Interactions
<div style="border: 1px solid black; padding: 5px; width: fit-content; margin: auto;"> <p style="text-align: center;">Functional Element Name</p> <p style="text-align: center; color: red;">Functional Element-Level Controller (if applicable)</p> </div>	Functional Element with the controller of its internal interactions (i.e., the functional element-level interactions)

Figure 24. Outer Planets Explorer Control Structure

Step 3.9.6 discusses how the control structure evolves iteratively to capture lower-level interactions and informs the lower-level design.

3.8 Step 8: Perform Preliminary Hazard Analysis

Using the information obtained in Steps 2 through 5, a preliminary hazard analysis using STPA is performed and the system hazard log is created.

STPA is performed as described in the background section 2.3. The results of STPA are recorded in the hazard analysis section of the intent specification, in level 1. It is worth reiterating from section 2.3 that in the initial phase, not much of the design information is known and that the STPA process will focus on safety constraint refinement and architecture selection. As discussed in step 9.5, STPA is an iterative process and in later iterations, STPA will involve more design refinement to enforce the safety constraints.

A partial result of an STPA hazard analysis from the OPE example is shown below in Figure 25. The STPA process is performed as usual and recorded as follows: Each inadequate control action is recorded once, and the relevant safety constraints and hazards are listed. For each inadequate control action the related control flaws and inadequate control executions are listed. The STPA process may lead the engineer to the discovery of C&DH-ICA1 by analyzing any of the hazard and safety constraint combination from the hazards and safety constraints below, but it is not relevant in a top-down presentation of the analysis to denote which safety constraint first leads to C&DH-ICA1.

The associated design decision column shows all of the design decisions related to the control flaw and associated inadequate control action. It's important to record the associated design decision because the engineer may decide to modify the original design to enforce the constraints. In the far right column new requirements or constraints or refined requirements or constraints are listed.

Inadequate Control Actions, Control Flaws, and Inadequate Executions of Control Actions	Relevant High-level Hazard(s)	Associated Design Decision	Resulting Safety Constraint(s) or Requirements
C&DH-ICA1. The C&DH executes and/or delegates MOC Directives that are wrong. (←S/C-SC1, S/C-SC2, S/C-SC3, S/C-SC5, S/C-SC7)			
C&DH-CF1.1. A change in the state of the spacecraft or spacecraft environment that invalidates assumptions of directives occurs during the time delay between DSN transmittal of the directives and S/C reception of directives.	H1, H2, H4, H5, H6, H7	C&DH-2.1	C&DH-SC1
C&DH-ICA2. The C&DH does not receive an initial set of MOC Directives and/or updates. (←S/C-SC1, S/C-SC2, S/C-SC3, S/C-SC5, S/C-SC7)	H1, H2, H4, H5, H6, H7	C&DH-2.1	
C&DH-CF2.1. The C&DH does not initiate the proper functionality for reception of MOC Directives.			C&DH-SC2
C&DH-ICA3. The C&DH does not send Spacecraft H&S Data when necessary. (←S/C-SC2)			
C&DH-CF3.1. The communication path to the COM functional element is broken, obstructed, or non-existent.	H2	C&DH-2.1.2	C&DH-SC3
C&DH-CF3.2. The MOC Directives become obsolete (i.e., expire) and no baseline functionality is provided to transmit spacecraft H&S Data.			C&DH-SC4

Figure 25. Partial STPA Hazard Analysis

Creation of the hazard log follows STPA. For each high-level hazard, the functional element pertaining to the hazard is listed as well as the relevant operation or mission phase. The causal factors shown in the hazard log are pointers to the control flaws identified in the STPA analysis. The hazard log also captures other information such as the hazard severity and type of potential loss resulting from the hazard. An example of the hazard log is shown below in Figure 26. The

hazard log is documented in level 1 of the intent specification. The hazard log also includes pointers to state variables and models that are relevant to each hazard.

<p>H1. Inability of Mission to collect data. (↓SV-85)</p> <p>System Element: Spacecraft (C&DH, SCI, EP, and A&AC), Launch Vehicle, and Mission Operations Center</p> <p>Operation/Phase: Pre-Launch, Post-Launch/Pre-Icy Moon Orbit, Icy Moon Orbit, Disposal</p> <p>Causal Factors: Spacecraft loses functionality to collect data, (←C&DH-CF1.1, C&DH-CF2.1, C&DH-CF8.1, C&DH-CF9.1, C&DH-CF10.1, A&AC-CF1.1, A&AC-CF2.1, A&AC-CF2.2, A&AC-CF2.3, A&AC-CF3.1, A&AC-CF4.1, A&AC-CF4.2, A&AC-CF4.3, A&AC-CF10.1, A&AC-CF10.2, A&AC-CF10.3, A&AC-CF10.4, A&AC-CF10.5, A&AC-CF10.6, A&AC-CF10.7, A&AC-CF10.8, A&AC-CF11.1)</p> <p>Level and Effect: Potential loss of data collection opportunities (↑ACC4)</p> <p>Safety Constraints: The mission must have the necessary functionality for data acquisition at the required times. (→SC1), The spacecraft must have the necessary functionality for data acquisition at the required times. (→S/C-SC1), The C&DH must validate the time and state dependent assumptions of directives generated by the MOC. (→C&DH-SC1), In the absence of valid MOC Directives, the C&DH must initiate the proper functionality for reception of MOC Directives. (→C&DH-SC2), ...</p>
--

Figure 26. Partial Hazard Log Example

3.9 Step 9: Define System Element Specifications

Steps 9.1 through 9.6 are used to

1. Define goals, assumptions, requirements, design constraints and safety constraints for each subsystem or functional element at level 1. (Refer to Figure 27 and Figure 28 for an example).
2. Make design decisions at level 2 to implement the requirements and constraints. (Refer to Figure 29 for an example).
3. Create a formal model of the control system design at level 3. (Refer to Figure 30 for an example).

Steps 9.1 through 9.6 are performed iteratively until the design is set.

Level 1.1.3: Spacecraft Attitude and Articulation Control (A&AC) Goals, Requirements, and Constraints

Spacecraft Attitude and Articulation Control (A&AC) Goals

A&AC-G1. To provide the spacecraft velocity changes necessary for orbit insertion about the icy moon of the outer planet, maintenance of/changes to that orbit, and spacecraft disposal. (←[S/C-R1](#)), (↓[2.2](#), [S/C-2.3](#), [C&DH-2.1.6](#), [SV-1](#), [SV-2](#)), (→[A&AC-R1](#), [A&AC-R2](#), [A&AC-R3](#), [A&AC-R4](#), [A&AC-R5](#)),

Rationale: Because our functional analysis has shown tight coupling between spacecraft attitude and spacecraft velocity changes, we have allocated these functions to one functional element, the A&AC. However, in order to avoid confusion, it is worth noting that traditionally, spacecraft velocity changes have been allocated to a separate “Propulsion” functional element or subsystem in JPL spacecraft.

A&AC-G2. To point the spacecraft and spacecraft elements in accordance with science data collection and communications needs for the mission. (↓[S/C-2.3](#), [C&DH-2.1.6](#), [SV-1](#), [SV-2](#)), (→[A&AC-R6](#), [A&AC-R7](#), [A&AC-R8](#), [A&AC-R9](#), [A&AC-R10](#))

Rationale: The pointing of the spacecraft and spacecraft elements are both allocated to the A&AC functional element because the rotation and translation of spacecraft elements with respect to the main spacecraft structure will affect spacecraft attitude.

Spacecraft Attitude and Articulation Control (A&AC) Assumptions

A&AC-A1. Telescoping boom segments for the HGA are not practical for this mission. (↓[A&AC-2.2.1](#))

Rationale: While this assumption is largely a simplifying assumption, the complexity associated with telescoping boom segments for an object as massive as the HGA would warrant a trade study beyond the scope of this study if they were to be considered.

A&AC-A2. Four or more HGA boom joints are not practical for this mission. (↓[A&AC-2.2.1](#))

Rationale: While this assumption is largely a simplifying assumption, the complexity associated with four or more boom joints would warrant a trade study beyond the scope of this study if they were to be considered.

Spacecraft Attitude and Articulation Control (A&AC) Requirements

A&AC-R1. After release from the launch vehicle, the A&AC shall provide spacecraft velocity changes for spacecraft transit from the release point to the orbit of the outer planet. (←[A&AC-G1](#)), (↓[A&AC-2.6](#), [SV-1](#), [SV-2](#))

A&AC-R2. Upon arrival to the orbit of the outer planet, the A&AC shall provide spacecraft velocity changes necessary for spacecraft capture in the orbit of the outer planet. (←[A&AC-G1](#)), (↓[A&AC-2.6](#), [SV-1](#))

A&AC-R3. After capture in the orbit of the outer planet, the A&AC shall provide the spacecraft velocity changes necessary for orbit insertion about the icy moon of the outer planet. (←[A&AC-G1](#)), (↓[A&AC-2.6](#), [SV-1](#))

A&AC-R4. After orbit insertion about the icy moon, the A&AC shall provide the spacecraft velocity changes necessary for orbit maintenance for at least TBD days. (←[A&AC-G1](#)), (↓[A&AC-2.6](#), [SV-1](#))

...

Figure 27. Level 1 OPE Example

Spacecraft Attitude and Articulation Control (A&AC) Design Constraints

A&AC-C1. All attitude and articulation control components must fit within TBD% of the space beneath the payload fairing of a Delta-IVH. (←[S/C-C2](#)), (→[AC&DH-C1](#)), (↓[A&AC-2.2.1](#))

Rationale: Space is a limited resource inside the payload fairing of a launch vehicle and thus, space for each component of the spacecraft must be carefully budgeted. The space allocation process involves a number of architectural tradeoffs beyond the scope of this study. Therefore, we will use a TBD% space allocation to the attitude and articulation control functional elements.

A&AC-C2. The A&AC functional element must be able to receive and execute directives from the C&DH functional element. (←[A&AC-G2](#)), (↓[C&DH-2.1.6](#)), (→[AC&DH-G2](#))

A&AC-C3. The A&AC functional element must provide HGA Pointing State Information to the C&DH functional element as an input. (↓[C&DH-2.1.6.1](#), [A&AC-2.1.2](#), [A&AC-2.2.1.5](#)), (→[AC&DH-G3](#))

...

Spacecraft Attitude and Articulation Control (A&AC) Safety-Related Design Constraints

A&AC-SC1. The HGA must not rotate or translate with respect to the main spacecraft structure while the spacecraft is inside the payload fairing of the launch vehicle. (←[H1](#), [H2](#), [H5](#), [H6](#), [H7](#)), (↓[A&AC-2.2.1.4](#), [A&AC-2.2.1.8](#), [A&AC-2.3](#), [A&AC-2.4](#)), (→[A&AC-ICA10](#), [A&AC-ICA11](#), [AC&DH-G1](#), [HA&T-G1](#), [HA&T-G2](#), [HA&T-G3](#))

A&AC-SC2. HGA translation and rotation with respect to the main spacecraft structure must be restrained during periods of spacecraft velocity changes so that the HGA does not deform and/or detach from the spacecraft due to inertial loads. (←[H1](#), [H2](#), [H4](#), [H5](#), [H6](#), [H7](#)), (↓[A&AC-2.2.1.4](#), [A&AC-2.3](#), [A&AC-2.4](#)), (→[AC&DH-G1](#), [HA&T-G1](#), [HA&T-G2](#), [HA&T-G3](#))

A&AC-SC3. While translating and/or rotating, the HGA and the radiation it emits must maintain minimum separation from other parts of the spacecraft. (←[H1](#), [H2](#), [H4](#), [H5](#), [H6](#), [H7](#)), (↓[A&AC-2.2.1.5](#), [A&AC-2.3](#), [A&AC-2.4](#), [AC&DH-2.1.2](#), [A&AC-2.2.1.9](#)), (→[A&AC-ICA10](#), [A&AC-ICA11](#), [AC&DH-G1](#), [HA&T-G1](#), [HA&T-G2](#))

A&AC-SC4. The communications paths between the A&AC functional element and the C&DH for transfer of the HGA Pointing State Information and HGA Translation State Information inputs must not become broken or obstructed. (←[H2](#)), (↓[A&AC-2.3](#)), (→[AC&DH-G1](#), [HA&T-G1](#), [HA&T-G2](#))

...

Figure 28. Level 1 Constraints OPE Example

...

A&AC-2.4. A HGA Attitude and Translation (HA&T) Control functional element is used for feedback control of the joint rotations necessary for HGA rotation and translation. (←[A&AC-2.1](#), [A&AC-2.2](#)), (↑[A&AC-SC1](#), [A&AC-SC2](#), [A&AC-SC3](#), [A&AC-SC5](#), [A&AC-SC6](#), [A&AC-SC7](#), [A&AC-SC8](#), [A&AC-SC10](#), [A&AC-SC12](#), [A&AC-SC14](#), [A&AC-SC15](#), [A&AC-SC16](#), [AC&DH-G1](#), [HA&T-G1](#), [HA&T-G2](#), [HA&T-G3](#)), (→[HA&T-2.1](#), [HA&T-2.2](#), [HA&T-2.3](#), [HA&T-2.4](#), [HA&T-2.5](#))

Rationale: HGA restraint, rotation, and translation are coupled heavily due to the multiple boom segment architecture.

A&AC-2.5. A Spacecraft Attitude Controller (AC) functional element is used for feedback control of spacecraft pointing. (↑[A&AC-R6](#), [A&AC-SC15](#), [AC&DH-G1](#)), (→[SV-1](#), [SV-2](#))

A&AC-2.6. A Spacecraft Translation Controller (TC) functional element is used for feedback control of spacecraft translation. (↑[A&AC-R1](#), [A&AC-R2](#), [A&AC-R3](#), [A&AC-R4](#), [A&AC-R5](#), [A&AC-SC16](#), [AC&DH-G1](#)), (→[SV-1](#), [SV-2](#))

A&AC-2.7. An A&AC Electrical Power (AEP) functional element receives and distributes electrical power from the EP to the A&AC functional elements. (↑[A&AC-C5](#), [A&AC-SC9](#), [A&AC-SC12](#), [AC&DH-G1](#))

...

Figure 29. Design Decisions OPE Example.

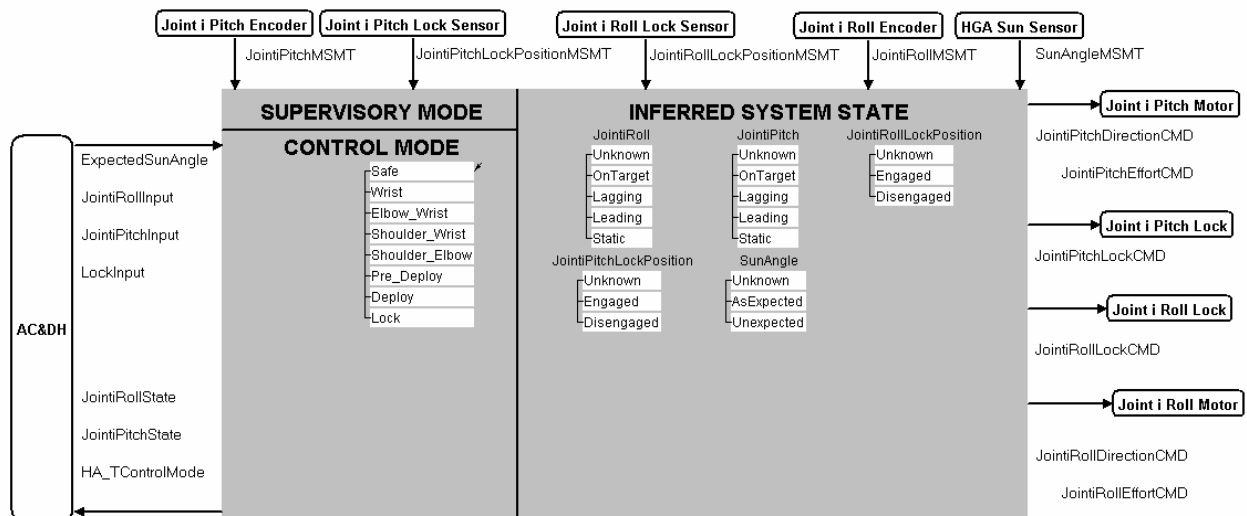


Figure 30. OPE Example of High Gain Antenna Control System Design

3.9.1 Step 9.1: Define Goals, Assumptions, Requirements, and Constraints

After the goals, assumptions, requirements, design constraints, and safety constraints have been defined for the mission, they can be defined for the functional elements identified in the high-level functional decomposition and control structure created in steps 6 and 7.

This step will be iterated in step 9.6, and subsequent iterations will involve the definition of goals, assumptions, requirements, and constraints on lower-level functional elements defined by further iterations of steps 9.2 to 9.5.

3.9.2 Step 9.2: Develop Models of the System Under Control

In this step, the system engineer performs state-based behavioral modeling of the system under control, per the State Analysis description in Section 2.5. Starting from the high-level hazards, constraints, and requirements, state effects diagrams are created, and models of the state variables, measurements, and commands are developed. These models are a repository for design information from subsystem engineers, including continuous physics, such as the pointing dynamics of the OPE's high-gain antenna, and discrete behavior, such as the operational mode and health of the joint motors for the HGA boom. The state-based behavioral model artifacts are documented on Level 2 of the Intent Specification. State Analysis is used to model the system under control; these models will be used in the design of the control system (estimators and controllers) described formally in Level 3 of the Intent Specification (per Step 10.3, below).

It is important that the models of the system under control are captured concurrently with the drawing of the state effects diagram. Drawing the state effects diagram alone shows all of the state variables and existence of physical effects between them, but without models that explicitly describe the nature of the physical effects, the state effects diagram is incomplete.

Also, while performing the state-based behavioral modeling of the system under control, assumptions and rationale must be documented. The extent to which the states and effects are modeled will be under the purview of the system engineer. For example, the number of “credible” fault modes for a device like the joint motor is subject to engineering judgment regarding the risk and severity of hazards related to these faults. Whereas an early iteration of the methodology may simply specify a “broken” fault for the motor, further iterations may identify control flaws and inadequate control executions related to more specific fault modes, such as “slipping”, “stalled”, and “stuck”, resulting in an elaboration of the joint motor health model to reflect these more specific fault modes.

Below, in Figure 31, is an example of the state effects diagram pertaining to the control of the HGA boom rotation joint angles. To control the angle of Joint i along degree of freedom j , commands are sent to the appropriate motor on Joint i . The angle measurement is used by the control system to send commands to the motors on joint i to control the angle. The lock position also has an effect on the joint motor (i.e., the joint motor is unable to operate nominally with the motor lock engaged).

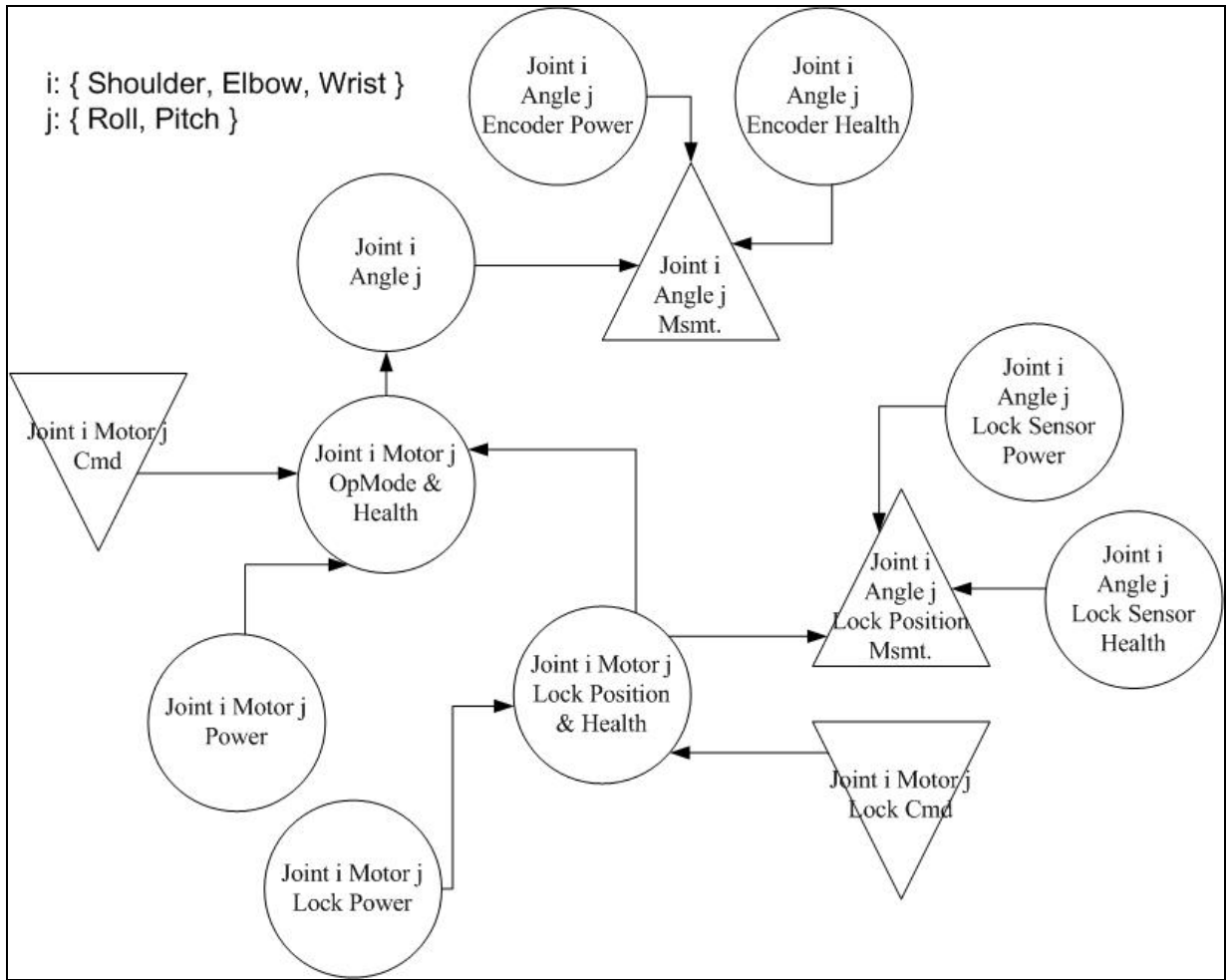


Figure 31. State Effects Diagram Example

Examples of state effects models for two of the state variables from this state effects diagram are presented in Figure 32 and Figure 33.

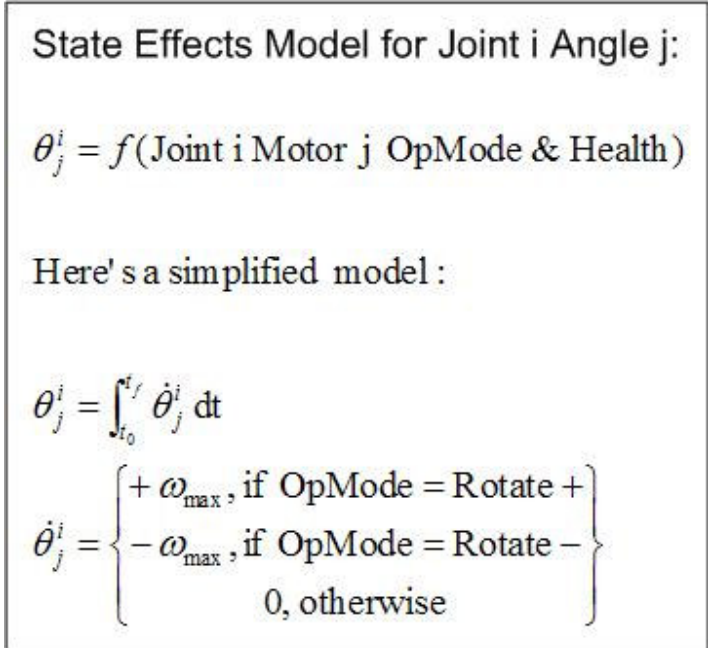


Figure 32. Continuous State Effects Model for Joint i Angle j

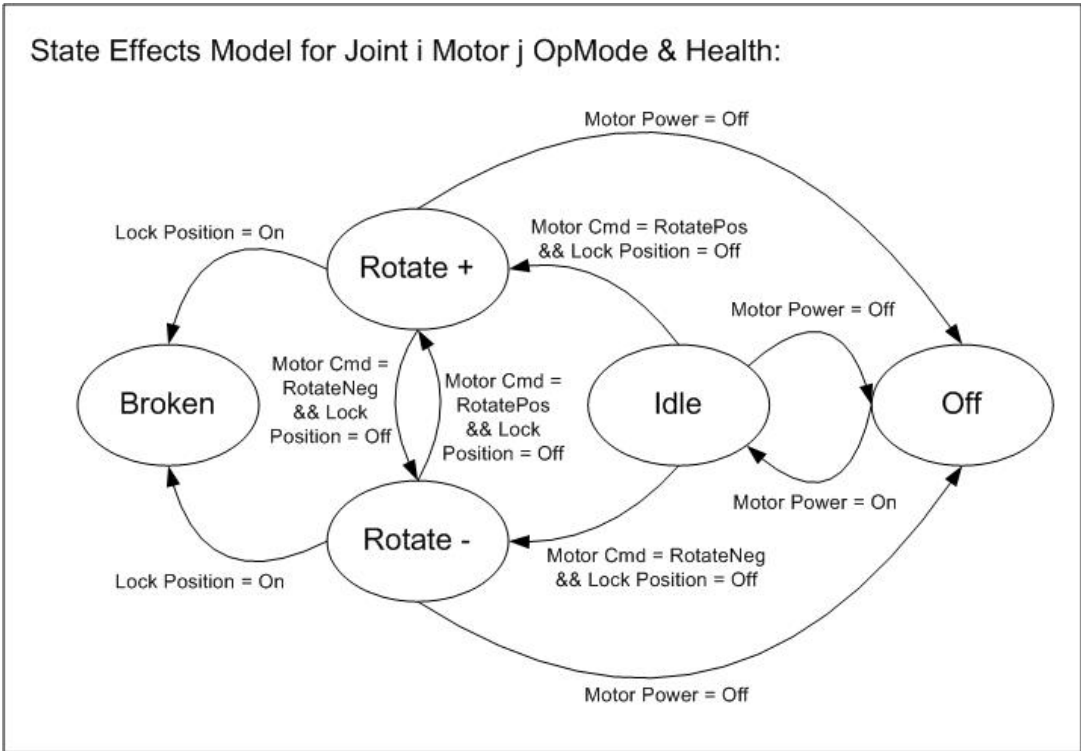


Figure 33. Discrete State Effects Model for Joint i Motor j OpMode & Health

3.9.3 Step 9.3: Define and Design Control System Operational Behavior

In this step, the system engineer documents design decisions pertaining to the control system. This information serves as the textual description of the control system and helps in the creation of the blackbox models. The state effects diagrams and models provide key insight into physical effects between the state variables of the system under control and are thus used to design the control system behavior appropriately.

For example, the design specification of how the HGA Boom directive is created and the AC&DH response is shown below in Figure 34. The design is a high-level description of the AC&DH controller behavior that will be shown formally in level 3.

AC&DH-2.1.1. The AC&DH computes and sends the ‘HGA Boom Rotation Joint Targets’ directive to the HA&T. (↑[AC&DH-R4](#), [AC&DH-R5](#), [AC&DH-C3](#)), (→[AC&DH-2.1.2](#))

AC&DH-2.1.1.1. The ‘HGA Boom Rotation Joint Targets’ directive is a vector of six values (in radians) corresponding to HGA Boom Shoulder Pitch, HGA Boom Shoulder Roll, HGA Boom Elbow Pitch, HGA Boom Elbow Roll, HGA Boom Wrist Pitch, HGA Boom Wrist Roll. (←[A&AC 2.2.1.1](#)) (→[SV-8](#), [SV-9](#), [SV-12](#), [SV-13](#), [SV-16](#), [SV-17](#))

AC&DH-2.1.1.2. The ‘HGA Boom Rotation Joint Targets’ directive is derived from C&DH directives related to spacecraft attitude and HGA pointing and the AC&DH’s knowledge of the following state information: Spacecraft Position, Spacecraft Attitude, and HA&T Control Mode.

Rationale: The AC&DH is assigned this function because it is the lowest-level functional element with access to state information relating to the spacecraft’s position relative to Earth, and the spacecraft’s attitude. This information is necessary to define pointing vectors for HGA signal radiation.

AC&DH-2.1.2. The AC&DH defines an imaginary 3-dimensional surface referred to as the "roll-pitch mask" that marks the allowable boundary of HGA motion and signal radiation relative to the main spacecraft structure and spacecraft appendages. Any C&DH directive that produces an ‘HGA Boom Rotation Joint Targets’ directive that would lead to a combination of shoulder, elbow, and wrist roll and pitch rotation that causes any part of the HGA, HGA boom, and/or the HGA’s radiated signals to breach the roll-pitch mask will not be authorized for execution by the AC&DH. (↑[A&AC-SC3](#), [A&AC-SC16](#), [AC&DH-C3](#)), (←[A&AC-2.1](#), [A&AC-2.2](#), [AC&DH-2.1.1](#)), (→[SV-8](#), [SV-9](#), [SV-12](#), [SV-13](#), [SV-16](#), [SV-17](#))

Rationale: The AC&DH is assigned this function because it is the lowest-level functional element with access to the ‘HGA Boom Joint Rotation Targets’ directive and state information relating to the attitude of spacecraft appendages relative to the main spacecraft structure. This information is necessary to determine HGA Boom Joint Rotation Targets that will violate the envelope of safe HGA translation and rotation.

AC&DH-2.1.2.1. The roll-pitch mask is derived from the mounting position of the HGA Boom Shoulder Joint on the main spacecraft structure, the model of the spacecraft’s structural configuration, and state information inputs from the estimators of other spacecraft appendage states.

AC&DH-2.1.2.2. Whenever an ‘HGA Boom Rotation Joint Targets’ directive is generated that would violate the roll-pitch mask, the AC&DH transitions into the “Safe” Control Mode

Rationale: If the AC&DH must reject the ‘HGA Boom Rotation Joint Targets’ directive that it derives, it must revert to some baseline functionality in order to restore nominal pointing operations. This baseline functionality could include reorientation of the spacecraft or the commanding of the HA&T to hold the current boom joint rotation angles. The issue of the determining what that baseline functionality should be requires a trade study during the next iteration of the STPA, which is beyond the scope of this study.

...

Figure 34. OPE Example of System Operational Behavior

The state effects models can be used in specifying the control system behavior. For example, the state effects diagrams in Figure 31 and Figure 35 include states relevant to design decision AC&DH-2.1.2.2. Given the current HGA Boom Joint rotation angles and the spacecraft attitude, if the MOC specifies a target set point for the HGA pointing that cannot be achieved without violating the roll-pitch mask constraint, a controller for the AC&DH functional element will transition to “Safe Mode” behavior. This control system design decision will be reflected in the blackbox models of control system behavior (see Step 9.4).

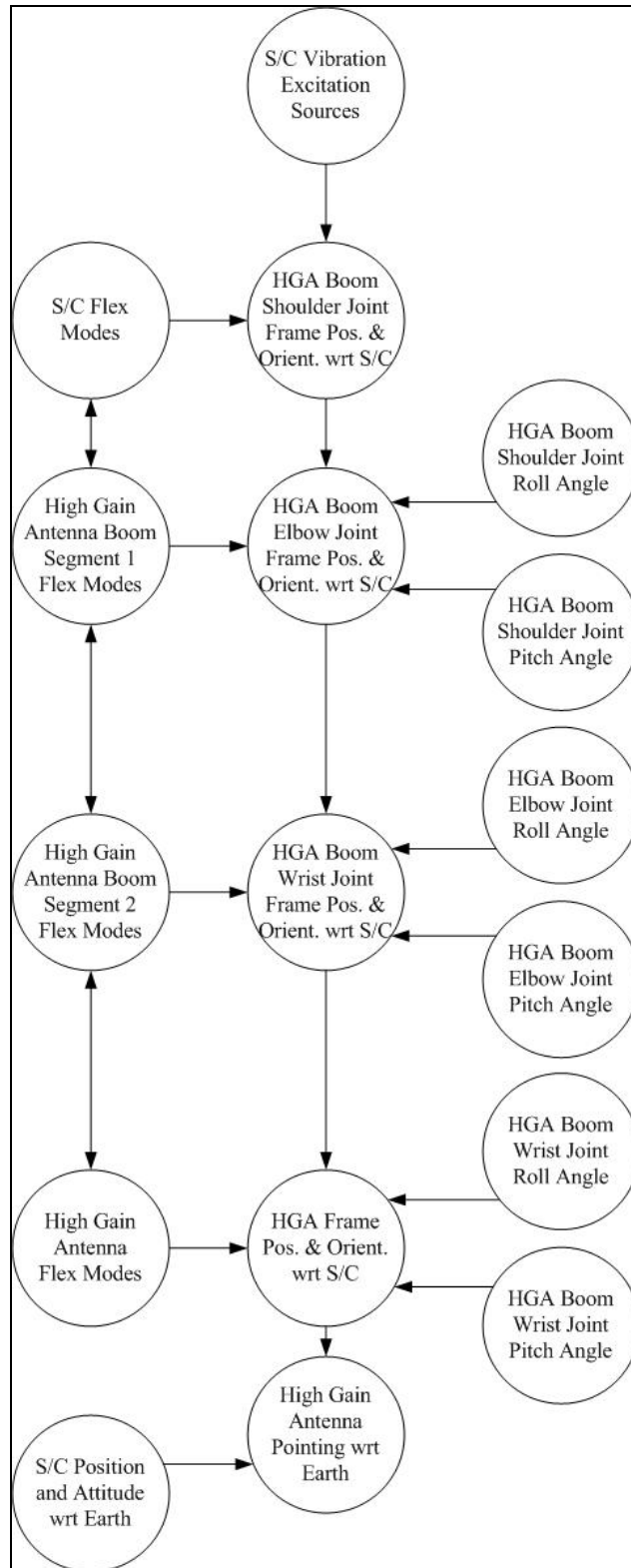


Figure 35. State Effects Diagram HGA Boom angle joints

3.9.4 Step 9.4: Develop Formal Models of the Control System.

In this step, the system engineer designs and specifies blackbox models of the control system using SpecTRM-RL. Estimation and control logic for discrete state variable models documented in level 2 can often be represented in SpecTRM-RL directly. Many state variables in the system under control model continuous phenomena, in which case engineers may either discretize the state variables or use SpecTRM *functions* and *macros* to represent state variable values.

Engineers use level 2 control system design and the control structure to populate the blackbox models. Figure 30 shows part of the formal control system, namely the control of the HGA Pointing state variable (within the HGA Attitude and Translation functional element). The behavioral models of the system under control (also at level 2) are also used to help specify the control system behavior. For example, the *and/or* table shown in Figure 36 shows part of the formal control system definition, namely the definition of the input, WristRollLockPositionMSMT. This input can be seen in the control system design shown in Figure 30 (note that the “Joint i” notation is used to compactly refer to the wrist, elbow, and shoulder joints).

DEFINITION

= New Data for WristRollLockPositionMSMT

WristRollLockPositionMSMT was Received	T
--	---

= Previous Value of WristRollLockPositionMSMT

WristRollLockPositionMSMT was Received	F
Time Since WristRollLockPositionMSMT was Last Received <= TBD seconds	T

= Obsolete

System Start	T	*	*
WristRollLockPositionMSMT was Never Received	T	T	*
Time Since WristRollLockPositionMSMT was Last Received > TBD seconds	*	*	T

Figure 36. Example *And/Or* Table

Engineers also use level 1 requirements and constraints to populate the blackbox model. For instance, there will often be safety constraints listed in level 1 for the maximum time allowed between safety-critical measurement readings. This information is used to specify the transition of a measurement from *valid* to *obsolete* in the SpecTRM-RL process model. Depending on the state variable, an obsolete measurement may, in turn, cause the estimate of a state variable to become UNKNOWN.

In summary, the formal control system design can be created according to the following guidelines:

1. The hierarchy of control modeled at level 3 should reflect the control structure described in level 2.
2. Control system behavior and the process model should reflect design described in level 2.
3. Control system design must enforce constraints and requirements documented in level 1.
4. Control system design should leverage state analysis artifacts as described below.

Referring back to the state effects diagram in Figure 31 we see how the process model design shown in Figure 30 can be informed from the state effects models. For the process to be controlled, in this case the HGA pointing process, the process model must include:

1. The state variable being controlled by this element of the control system (in this example, this state variable is the HGA joint angle, which is affected by the angle cmd).
2. All of the affecting state variables of the state variable being controlled by this element of the control system, which would be considered process inputs in the generic control loop shown in Figure 4.
3. All of the measurements in the state effects diagram affected by the state variable being controlled by this element of the control system, and other state variables directly affecting these measurements.

The state variables referred to in 1 and 2 are the relevant state variables needed for the control system to choose control modes. The measurements referred to in 3 are required for the control system to infer the value of the state variables in 1 and 2 and thus infer the state of the process to be controlled.

This process model is written in SpecTRM-RL and can be used for traceability, completeness analysis, and automatic code generation.

Distinction between SpecTRM-RL models and state effects models as used in this task:

State effects models are representations of the physics in the system under control. SpecTRM-RL models are discrete representations of the control system and the control's system's model of the system under control that together specify the system blackbox behavior. Succinctly, the state effects diagram and models describe the behavior of the system under control while SpecTRM-RL models describe the behavior of the control system.

Consequently, in the SpecTRM-RL model of the control system, system state variables will always have an "Unknown" state. If, for example, a measurement was not received the controller may not be able to infer the current value of a state variable and the controller's model of that state variable would transition to "Unknown." SpecTRM's handling of missing measurements is consistent with the state-based software design aspect of State Analysis, where the control system specifications are required to consider sources of uncertainty in the state estimates.

3.9.5 Step 9.5: Continue to Perform STPA

STPA is performed in parallel with the creation of design as discussed in section 3.8. As new design is created and new control flaws and inadequate control executions are found, the models of the system under control must be augmented and used for further hazard analysis. That is, consideration and mitigation of newly documented hazards should be reflected in the state effects models, to the extent that the hazards involve behavior of the system under control. After new design of the system under control has been specified, engineering judgment is used to determine whether other important state variables need to be added to the models of the system under control. Similarly, as new design is created and control flaws and inadequate control executions are found, the blackbox models of the control system must be augmented and used for further hazard analysis. That is, consideration and mitigation of newly documented hazards should be reflected in the SpecTRM-RL blackbox models, to the extent that the hazards involve behavior of the control system.

State Analysis modeling and the STPA process can be applied synergistically to document mitigations of inadequate control actions, model the system under control and refine the control system design. For instance, many inadequate control actions stem from poor controller behavior due to the controller's incorrect process model. The state effects diagram should aid in creating the process model and documenting the actual physics of the system. State analysis can also be used in the discovery of control flaws—state effects that could lead to instances of inadequate control.

New design is created during the STPA process to enforce newly defined constraints or better enforce existing constraints. In turn, the STPA process will lead the engineer to reevaluate state effects models and discover new state effects. These additional state effects must be recorded in an updated state effects diagram and modified state effects models. In addition, the STPA process may inspire changes to the control system and the control system's process model in which case the SpecTRM-RL blackbox models should be modified as well.

3.9.6 Step 9.6: Iteratively Refine the System Design

Two products completed at a high-level in steps 1 to 8 are iterated on to refine inform the lower-level design.

DSM Iteration (Functional Decomposition):

The identification and decomposition of lower-level functional elements must be performed in tandem with steps 9.1 to 9.6. As new or lower-level requirements and constraints are generated, new and lower-level functions within the elements will be identified in the DSM. Accordingly, the functional elements will each be decomposed with another DSM.

Control Structure:

The control structure is another critical element to inform the design process. As steps 9.1 through 9.5 are performed, the control structure will be fleshed out with more detail, which in turn should aid the STPA hazard analysis. The control structure evolves iteratively to capture lower-level interactions and informs the lower-level design.

Repetition of steps 9.1 through 9.6 may also change products from other steps in the methodology. Feedback to the higher levels of the methodology can occur as engineers create

new requirements and constraints as a result of STPA or if the hazard analysis inspires engineers to modify level 1 goals or level 2 design decisions.

The products from performance of step 9.1 act as inputs to steps 9.2 to 9.5 while at the same time, products from the performance of steps 9.2 to 9.6, will change the products of step 9.1 (e.g., generation of functional goals, requirements, and constraints).

Furthermore, other artifacts from steps 1 to 8 can change if new constraints or modified design (perhaps modified as a result of performing step 9.5) results in elimination of a high-level hazard and associated safety constraint. Iterations through the methodology, both through the “outer loop” of steps 1 to 9 and the “inner loop” of steps 9.1 to 9.6, are finished when the design is set and all hazards are eliminated, mitigated, or controlled.

3.10 Step 10: Perform Validation Tests

Steps 1 to 9 were the primary focus of this task, but validation is an important part of the methodology. SpecTRM has several tools for validation. SpecTRM-RL blackbox behavior models are executable and analyses can be performed on them (completeness, robustness, consistency) to evaluate and identify errors and omissions. If validation tests fail, they will provide an entry point back into modifying the design of the system.

3.11 Step 11: Generate Designs and Software Code

The final design of the system under control and the implementation of the control system in software code are the ultimate end-products of the methodology. Physical component designs can be generated from the model, and software code can be generated either manually or automatically.

4. Methodology Conclusion and Summary

4.1 Conclusion

The methodology described above defines a rigorous approach to complex system design. However, it is worth noting that the methodology is not a linear, “cookbook” process. This methodology is intended to support the natural opportunistic and iterative kind of design that good engineers typically follow, rather than trying to confine the design process into a rigid series of steps, which is often not followed and presents an encumbrance to the engineer.

This methodology presents a system engineering method that would assist engineers in the implementation of the new NASA software safety standard. The complete traceability between the products in the methodology is what is needed to implement the NASA software safety standard in a way that does not require extra or needless documentation from the engineers.

JPL engineers have typically thought of safety as human safety and pre-launch safety. We believe that by regarding safety as assuring mission success and following this methodology, a host of mission failures and accidents can be avoided. This methodology presents engineering that system engineers always perform, but provides a framework for better organizing that effort.

Implementing these steps creates a safer design without unnecessary documentation. The rigor of the methodology is achieved by a re-thinking the creation of typical system engineering specifications and using models directly to inform the control system design.

This methodology addresses the increasing complexity of spacecraft systems. The resulting specification is structured and provides complete traceability. Most critically, the STPA process folds the hazard analysis directly into the design process. Integration of STAMP, STPA, intent specifications, and State Analysis allows system engineers, software engineers and hardware engineers to share a common specification document that describes the motivation and rationale for design decisions and provides access to the models that were used in the creation of the design.

4.2 Methodology Summary

1. Identify Mission Goals, Requirements and Constraints.
Products: Level 1 intent specification of mission goals and constraints
2. Define System Accidents or Unacceptable Losses
Products: Level 0 intent specification documenting the accidents.
3. Define High-level Hazards.
Products: Level 1 intent specification documenting high-level hazards.
4. Define High-level Safety-related Constraints
Products: Level 1 intent specification documenting safety constraints.
5. Identify Environment and Customer Constraints
 - environmental constraints and environmental assumptions
 - customer-derived system design constraints
 - customer programmatic constraints (e.g., budgets, etc.)Products: Level 1 intent specification of environmental constraints and environmental assumptions, customer-derived system design constraints, customer programmatic constraints.
6. Perform High-level Functional Decomposition
Products: Level 1 intent specification documenting the functional decomposition.
7. Design High-level System Control Structure
Products: Level 1 intent specification documenting the high level control structure.
8. Perform Preliminary Hazard Analysis using STPA and Create Hazard Log
Products: Level 1 intent specification documenting the STPA hazard analysis
9. Define System Element Specifications
Products:
 - Level 1 intent specification documenting goals, requirements, design constraints and safety constraints for each subsystem or functional element.

- Level 2 intent specification documenting design decision made to implemented the requirements and constraints on level 1.
- Level 3 intent specification documenting the formal design of the control system

9.1 Define Goals, Assumptions, Requirements, and Constraints for Each Subsystem or Functional Element

Products: Level 1 intent specification documenting the goals, assumptions, requirements and constraints for each subsystem or functional element.

9.2 Develop Models of the System Under Control

Products: Level 2 intent specification documenting state analysis artifacts: State effects diagram, state variable models.

9.3 Define and Design Control System Operational Behavior to Implement Requirements and Constraints.

Products: Level 2 intent specification documenting high-level control system behavior.

9.4 Develop Formal Models of the Control System

Products: Level 3 intent specification documenting the formal blackbox behavior of the system.

9.5 Continue to Perform STPA.

Products:

- Level 1 intent specification documenting updated STPA hazard analysis, refined requirements and constraints.
- Level 2 intent specification documenting updated state analysis artifacts, and system design and high-level control system design.
- Level 3 intent specification documenting updated blackbox behavior.

9.6 Iteratively Refine the System Design

Products:

- Level 1 intent specification documenting updated STPA hazard analysis, refined requirements and constraints.
- Level 2 intent specification documenting updated state analysis artifacts, and system design and high-level control system design.
- Level 3 intent specification documenting updated blackbox behavior.

10. Perform Validation Tests

Products: Test results

11. Generate Designs and Software Code

Products: Design specifications and software code.

5. References

- [1]. Leveson, Nancy G. (2000), "Intent Specifications: An Approach to Building Human-Centered Specifications, *IEEE Transactions on Software Engineering*, Vol. 26, No. 1, pg. 15-35.
- [2]. Leveson, Nancy G. (2004), "A New Accident Model for Engineering Safer Systems," *Safety Science*, Vol. 42, No. 4, pp. 237-270.
- [3]. Weiss, Kathryn A.; Nicolas Dulac; Stephanie Chiesi; Mirna Daouk; David Zipkin; and Nancy G. Leveson (2006), "Engineering Spacecraft Mission Software using a Model-Based and Safety-Driven Design Methodology," *AIAA Journal of Aerospace Computing, Information, and Communication*, Vol. 3, No. 11, pp. 562-586.
- [4]. Leveson, Nancy. <http://sunnyday.mit.edu/book2.pdf>
- [5]. Leveson, Nancy (1995), *Safeware: System Safety and Computers*, Addison-Wesley Publishing Company. ISBN 0-201-11972-2.
- [6]. Ingham, M., Rasmussen, R., Bennett, M., and Moncada, A., "Generating Requirements for Complex Embedded Systems using State Analysis", *Acta Astronautica*, Vol. 58, No. 12, June 2006, pp. 648-661.
- [7]. Kordon, M., Wall, S., Stone, H., Blume, W., Skipper, J., Ingham, M., Neelon, J., Chase, J., Baalke, R., Hanks, D., Salcedo, J., Solish, B., Postma, M., and Machuzak, R., "Model-Based Engineering Design Pilots at JPL", *2007 IEEE Aerospace Conference Proceedings*, Big Sky, MT, March 2007.
- [8]. Clark, Karla B. (2007), "Europa Explorer—An Exceptional Mission Using Existing Technology," *2007 IEEE Aerospace Conference Proceedings*, Mar. 3-10, Big Sky, MT, paper #1417.
- [9]. Dulac, Nicolas and Leveson, Nancy G. (2004), "Incorporating Safety in Early System Architecture Trade Studies," *Proceedings of the 2004 International System Safety Conference*, San Diego, CA.
- [10]. Browning, Tyson R. (2001), "Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions," *IEEE Transactions on Engineering Management*, Vol. 48, No. 3, pp. 292-306.
- [11]. Perrow, Charles (1999), *Normal Accidents: Living with High-Risk Technologies*, 1999 Edition, Princeton University Press, Princeton, NJ.

6. Appendix I

AN APPROACH TO INCORPORATING SAFETY IN EARLY CONCEPT FORMATION AND SYSTEM ARCHITECTURE EVALUATIONS

Nicolas Dulac and Nancy G. Leveson

International System Safety Conference, San Diego, CA., 2004

ABSTRACT

While it is important to include risk in the early concept formation stage of a project, evaluating risk in these early stages is difficult due to the lack of design information available. In our MIT/Draper Laboratory contract from NASA to perform a Concept Evaluation and Refinement (CE&R) study for going to the Moon and Mars, safety was defined upfront as one of the most critical criteria for exploration success. Safety, therefore, needs to be an integral part of the early concept formation and trade studies so it can influence key architectural decisions that will be impossible to change later in the system lifecycle. The information available at the very beginning of a large project like this, however, is too high level to perform a full-fledged hazard or risk analysis. The need for some new way to perform a preliminary hazard analysis to inform the early system architecture studies became readily apparent. This paper describes an approach to performing a preliminary risk evaluation developed at MIT for a NASA study contract for the new space exploration initiative.

1. INTRODUCTION

The new Space Exploration Vision outlined by the President involves a return to the moon as a stepping-stone for the future human exploration of Mars. As part of this effort, MIT and the Draper Laboratory are members of a joint team that was awarded a contract to perform a Concept Evaluation and Refinement (CE&R) study for the development of a space exploration architecture that fulfills the needs of the many stakeholders involved in the exploration enterprise. Safety was defined upfront as one of the most critical criteria for a successful space exploration enterprise. It was also recognized that considering safety issues early in the system lifecycle would maximize safety while minimizing negative impacts on the system. In an ideal system safety process, hazard analysis begins at the very start of the system lifecycle. The analyses are continuously revised and refined and are used to design hazards out of the system proactively or, if that is not possible, to mitigate them in the design, instead of adding safety features to a completed design. In this project, we had the opportunity to consider safety at the very beginning of the lifecycle and to influence key architectural decisions that would be impossible to change later.

However, we soon realized that the information available at the very beginning of the project, although broad and plentiful, was so high-level that it would be impractical to perform a full-fledged hazard analysis at this stage. Preliminary Hazard Analyses could be performed and used in the early concept trade studies, but the scope of the system and breadth of information available created some difficult challenges. The methodology presented in this paper describes the approach we developed to perform a structured preliminary hazard analysis while coping with the complexity associated with the large scope of the system. The entire space exploration enterprise was considered in the analysis. In order to limit the scope of this article, however, the methodology is demonstrated using the transportation subset of the exploration architecture. In other words, the scope of the analysis presented in this paper includes bringing humans from earth orbit to mars surface and from moon/mars surface back to earth orbit. Earth launch and re-entry, as well as moon/mars surface operations, are not considered in this paper.

2. TASK DEFINITION AND KEY ASSUMPTIONS

The methodology developed involves an analysis of the hazard mitigation potential of several candidate architectures. A key assumption underlying the risk analysis is that for novel systems with little historical data, it is not possible to accurately predict the likelihood of any particular hazard occurring. The difficulty in predicting hazard likelihood is especially great at the very beginning of conceptual studies, where virtually no design

information is available. Inaccurate a priori evaluations of hazard likelihood inevitably lead to incorrect risk assessments that can compromise the safety of the system. Accidents in complex systems often happen because some hazards were initially overlooked due to an overoptimistic initial evaluation of likelihood. While hazard likelihood is extremely difficult to evaluate, hazard severity can be evaluated relatively easily by considering the worst-case scenario loss associated with a specific hazard.

The analysis described in this paper takes into account the randomness of some events such as micrometeoroid strikes, solar flares, and some mechanical failures, but it also recognizes that complex aerospace systems usually fail in non-random ways. For example, neither the Challenger nor the Columbia accidents involved unpredictable random failures. Rather, the accidents were caused by a failure to handle well-known and documented risks adequately [1]. Consequently, the number of hazardous activities performed and the time of exposure alone are not good indicators of risk. In other words, exposure to hazards is not considered as important as the design and operational strategies built into the system to mitigate hazards.

As with any hazard analysis process, the objective of the risk analysis presented in this article is to assist in designing a transportation architecture with an inherently high hazard-mitigation potential. Mitigating hazards at the architecture level improves safety from the very beginning of the system lifecycle, leaving fewer hazards to be mitigated at the system, sub-system, component and procedural levels. Safety is defined broadly in this paper: The analysis is not limited to preventing human losses; it also includes risks associated with mission and equipment losses.

3. ARCHITECTURAL DESIGN SPACE AND INFORMATION AVAILABLE

As mentioned previously, the task of the safety team was to create a methodology to evaluate the risk associated with different space transportation architectures. The information available at this stage of the lifecycle was in the form of a large number of possible Earth-to-Moon/Mars-and-back (E-M-E) transportation architectures created through a graphical programming language called Object Process Network (OPN). OPN [2] is a domain-neutral meta-language that can be used to represent, generate, and evaluate system models. The OPN architecture generator was used to create 1162 feasible transportation system designs. OPN essentially generates architectures by selecting transportation vehicles and functions based on a set of combination rules and constraints. A large number of these architectures were automatically filtered out based on launch mass constraints and other feasibility criteria. The risk evaluation methodology presented in this article could have been automated and used upfront as one of the architecture-filtering criteria. However, a more practical approach was taken, which involved filtering out highly inefficient architectures from a mass and feasibility perspective, and then performing a risk evaluation on the remaining subset of transportation architectures. A sample OPN-generated architecture visualization is provided in figure 1. A typical OPN transportation architecture includes information such as:

1. The number and type of vehicles and modules used
2. The role and activities for each vehicle/module, including:
 - a. Dockings and un-dockings
 - b. Assembly of vehicle/modules stacks
 - c. Discarding of vehicles/modules
 - d. Prepositioning of vehicles/modules in orbit and on the planet surface

In addition to the OPN-generated architectures, several Technology/Policy choices were available such as:

1. Propulsion Type: Hydrogen, Methane, Nuclear thermal, Nuclear electric...
2. Orbit Transfer Trajectory: Free-return, Hohmann, Bi-elliptical, Spiral...
3. Orbit Insertion Type: Propulsive, Aerocapture,...

Paper size constraints prevent the listing of every options and choices included in the analysis.

Architecture MB1

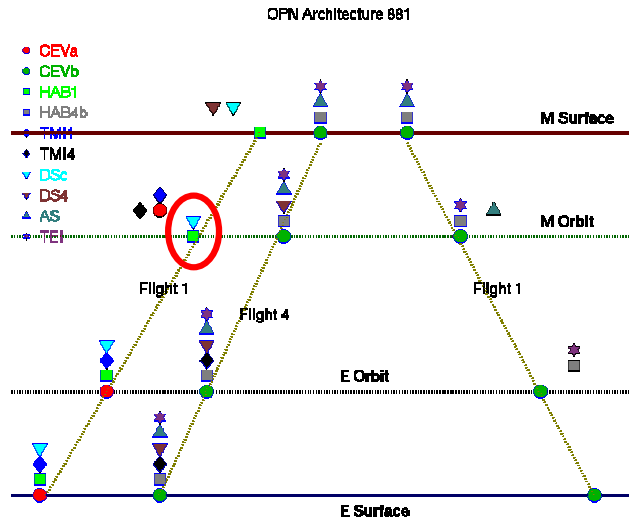


Figure 1: Sample Transportation Architecture Visualization

The entire architectural space can be theoretically obtained by taking the cross product of all OPN-generated transportation architectures and the possible combination of technology/policy parameters and options. The next section provides a step-by-step description of the methodology developed to evaluate risk for a subset of this architectural trade space.

4. HAZARD-BASED RISK ANALYSIS AND METHODOLOGY OVERVIEW

The hazard-based safety/risk analysis developed can be described as a three-step process:

1. Identify system-level hazards and associated severities
2. Identify mitigation strategies and associated impact
3. Calculate safety/risk metrics for a given transportation architecture

The first two steps are performed only once, at the beginning of the process. They may have to be repeated if the architectural design space changes and/or if additional hazards are identified. The third step is repeated in order to evaluate as many transportation architectures and variations as necessary. The following sections discuss each of the three steps in more detail:

4.1 Step 1: Identify System-Level Hazards and Severities

Just as in typical hazard analyses, the first step in the methodology is to identify the system-level hazards. A Hazard Identification and Tracking Database was created to streamline this process and to ensure proper tracking of the hazards. As with any preliminary hazard analysis activities, identifying system-level hazards involves 10% creativity and 90% experience. Consequently, domain experts were closely involved in identifying hazards for each mission phase.

Once the hazards were identified, the severity of each hazard was evaluated by considering the worst-case loss associated with the hazard. The losses were evaluated for each of three categories: Humans (H), Mission (M), and Equipment (E). A custom severity scale (see figure 2) was defined to account for the losses associated with each category.

Severity	Human - H
4	Loss of Life
3	Severe Injury or Illness
2	Minor Injury or Illness
1	Less than Minor Injury or Illness
Severity	Mission - M
4	Mission Abort or Total Mission Loss
3	Major Mission Objectives Incomplete
2	Minor Mission Objectives Incomplete
1	Less than Minor Objectives Incomplete
Severity	Equipment - E
4	System Loss
3	Major System Damage
2	Minor System Damage
1	Less than Minor System Damage

Figure 2: Hazard mitigation Scale and Priority

Some hazards identified, such as fire, explosion, or loss of life-support span multiple (if not all) mission phases. These hazards were grouped under the label “general” hazards to simplify the analysis. However, the mitigation strategies associated with these hazards depend on the mission phase to which they apply. The chart in Appendix 1A presents a summary of the identified hazards and severities, organized by mission phase.

4.2 Step 2: Identify Mitigation Strategies and Associated Impact

The second step of the methodology involves the identification and assessment of possible mitigation strategies for each hazard. The scope of the analysis was limited to the architectural option space predetermined by the transportation architecture group. However, the architectural space has a tendency to change very rapidly at the beginning of a project when different options are being explored at a rapid pace. Fortunately, the methodology is highly flexible and allows for rapid re-evaluations of architectures when changes occur. The key to this analysis is to determine the impact of each architectural option on each system-level hazard. In order to do so, a 4-level hazard mitigation impact scale was used (see figure 3). This scale is based on typical system safety hazard mitigation priority scales [3] and is used to determine the impact (if any) of a given architecture option on a given hazard. A database of mitigation impact was generated with the help of domain experts and was recorded in a spreadsheet. It contains the mitigation impact (1-4) of each architectural option, for each hazard, for each category (Human, Mission, Equipment).

Mitigation Impact	
Level	General Description
1	Eliminate Hazard
2	Prevent Hazard
3	Control Hazard
4	Reduce Damage

Figure 3: Mitigation Impact Scale

Appendix 1B provides a snapshot of the tool created to record hazard mitigation information and evaluate architectures. The system-level hazards and their associated severities (human, mission, and equipment) are listed in the top rows. The architectural and technology options are listed in the column on the left. The architectural space is divided into parameters with alternatives (e.g.: the parameter of launch vehicle type has alternatives HLLV, EELV, STS-derived, etc.) The effects of architectural parameters and technology options on each hazard are recorded in the database according to the 1-4 mitigation scale. For example, not performing Rendezvous in Transit and/or Highly Elliptical Orbital Rendezvous reduces the likelihood of being unable to dock (Mitigation Level: 3). Experience has shown that recording the rationale for each mitigation impact is critical as the database size makes it impossible to remember the inputs of every domain expert, which makes future changes very difficult. Once the

hazard mitigation database has been populated, it is possible to start evaluating the overall mitigation potential of various transportation architectures.

4.3 Step 3: Evaluate Architectures and Calculate Safety/Risk Metrics

A complete transportation architecture is defined as the union of an OPN-architecture with a set of technology/policy parameter options. In order to evaluate the risk associated with a specific architecture, an architecture vector is created that includes all of the parameter for that architecture. This vector is in the form of a large string of binary numbers. Looking at Appendix 1B, the architecture vector can be found in column C of the spreadsheet. The “1” values in the vector indicate that the corresponding option is selected. In this example, the vector shows that the selected architecture includes:

1. No Nuclear Surface Power
2. Low Level of Autonomy
3. No Highly Elliptical Orbital Rendezvous
4. No Rendezvous in Transit
5. No Artificial Gravity
6. And so on...

An architecture vector has to be created for each architecture evaluation. The architecture vector generation process can easily be automated if a large number of evaluations have to be performed. Once an architecture vector has been defined, the risk evaluation and metrics computation proceeds as follows:

1) For each hazard, and each hazard category (human, mission, equipment), that is, for each column of the spreadsheet, the tool searches for the option that provides the maximum hazard mitigation for each architectural parameter. These maximum hazard mitigations are added to obtain the total maximum hazard mitigation factor for each hazard and each category (H,M,E). These Maximum Mitigation Factors are architecture-independent. They only depend on the architectural option space and the hazards identified. The process of searching for maximum mitigation factors must be automated to provide the flexibility necessary to modify the architectural space or to make the tool evolvable. The maximum mitigation factors obtained in this step is almost never achievable in real-life because it would either be impractical or too expensive, or simply because some architecture options conflict with others in terms of hazard mitigation potential. Nevertheless, the maximum hazard mitigation factor is important because it provides an architecture-independent theoretical absolute upon which all the other architectures can be compared.

2) To evaluate a specific architecture, the analysis tool matches the selected options with their respective hazard mitigation impact and computes a sum of the mitigation factors obtained for the options selected in the architecture under evaluation. This process is repeated for each hazard and category (H, M, E). The result is a set of Hazard Mitigation Indices obtained for a particular architecture.

3) A Relative Hazard Mitigation Index is calculated for each hazard and each category using the following formula:

$$\text{Relative Hazard Mitigation Index (i)} = 1 - (\text{Hazard Mitigation Index (i)} / \text{Maximum Mitigation Factor (i)})$$

4) If a hazard is completely eliminated (Mitigation Level = 4) by a selected architectural option, the Relative Hazard Mitigation Index for this hazard is automatically set at zero.

5) A Relative Severity Index for each hazard and category is then calculated as follows: Relative Severity Index (i) = Relative Hazard Mitigation Index (i) * Original Hazard Severity (i) ^2. The “squared” severity is used to provide heavier weighting on the higher severity indices.

6) Three Safety/Risk Metrics (Human, Mission, Equipment) are obtained by aggregating the Relative Severity Indices for each category (H, M, E) across all hazards.

7) As needed, an Overall Safety/Risk Metric (OSRM) for an architecture can be obtained using a weighted average. In this project, the weighting factors selected were: Human: 9, Mission: 3, Equipment: 1.

Figure 4 provides a summary of the weighting factors used in the calculation of the OSRM for the MIT/Draper CE&R project.

		Human	Mission	Equipment
Hazard Severity	Weight	9	3	1
4.Catastrophic	16			
3.Critical	9			
2.Major	4			
1.Marginal	1			

Figure 4: Weighting Factors Summary

The Safety/Risk metrics are used to evaluate and rank potential transportation architectures. By systematically selecting and deselecting options in the architecture description, it is possible to perform a first-order assessment of the relative importance of each architectural option in determining the Overall Safety/Risk Metric.

5. SAMPLE RESULTS

Because of paper length constraints, it is impossible to provide a detailed description of analysis results. However, the main output of the risk/safety analysis performed is a preliminary ranking of the selected transportation architectures based on the hazard mitigation potential of each. According to the analysis, Mars Baseline Architecture 1 (MB1) (see Figure 1) is the architecture that has the most inherent hazard mitigation potential. Hundreds of parameters are considered in the Safety/Risk analysis, but some major contributors to the hazard mitigation potential of MB1 include the use of heavy module prepositioning on the surface of Mars and the need for rendezvous and docking maneuvers. Prepositioning modules allows for pre-testing and mitigates the hazards associated with loss of life support, equipment damage, and so on. On the other hand, prepositioning modules increases the reliance on precision landing to ensure that all landed modules are within range of each other. Consequently, using heavy prepositioning may require additional mitigation strategies to reduce the risk associated with landing in the wrong location. As another example, the transportation architecture MB1 requires no docking at Mars orbit or upon return, which mitigates the risk of collision or failed rendezvous and docking maneuvers.

An automated tool was created to perform multiple evaluations based on the needs of the team responsible for designing the transportation architecture. Hundreds of different architecture evaluations have been performed with little maintenance and data input efforts. The analysis started at the very beginning of the conceptual design phase. However, by the end of the first midterm review (4 months into the project), the information available was very different than at the very beginning. The methodology proved flexible and extensible enough to carry us from day 1 to the beginning of the detailed design phase, at which point, a more detailed hazard analysis methodology such as STPA [4,5] will be necessary. In addition, while automated tools can be used to rapidly perform a large number of analysis, it was understood that in some cases, system safety analysts and conceptual design experts may require more information than the metrics provided by the automated tool. An additional module was created that performs sensitivity analyses to identify the main contributors to risk mitigation in specific architectures. The result of this analysis can help system architects and safety people in fine-tuning their analysis or coming up with new solutions to further improve risk mitigation.

6. CONCLUSION

The methodology presented in this paper was developed in order to perform a structured preliminary hazard analysis to inform the early system architecture trade studies of the MIT/Draper CE&R project. The need for such a methodology emerged early in the project because of the highly complex, broad-scoped, multi-vehicle, time-dependent nature of the exploration enterprise. The methodology was developed with the specific purpose of evaluating transportation architectures generated by the MIT/Draper team. However, care was taken to make the methodology flexible, extensible, and adaptable to other complex systems. This paper focused on describing the methodology by using a very structured transportation architecture generation scheme as an example. However, the methodology has been applied with equal success to perform a structured preliminary hazard analysis of the surface operations mission architecture of the same project. The information available for the surface architecture

evaluation had very different format and content, but the safety analysis was similar. Our experience with the early system architecture trade studies for the MIT/Draper CE&R project convinced us that a highly structured preliminary hazard analysis process is required to incorporate safety early in a project of this size and complexity. The methodology described in this paper provides the structure necessary to perform preliminary hazard analysis early in the development of complex engineering projects.

7. ACKNOWLEDGEMENTS

This work was partially supported by a NASA CE&R contract to MIT and Draper Labs and by a grant (NAG2-1543) from the NASA Engineering for Complex Systems Program.

8. REFERENCES

- [1] Nancy Leveson, Nicolas Dulac, “Safety and Risk-Driven Design in Complex Systems-of-Systems. 1st Space Exploration Conference, Orlando, January 2005.
- [2] Benjamin Koo, “A Meta-Language for System Architecting”. Ph.D. Thesis, 2004. Engineering Systems Division (ESD), Massachusetts Institute of Technology.
- [3] Nancy Leveson, Safeware: System Safety and Computers, Addison-Wesley Publishers, 1995.
- [4] Nancy Leveson, “A New Approach to Hazard Analysis for Complex Systems.” International Conference of the System Safety Society. 2003.
- [5] Nicolas Dulac, Nancy Leveson, “An Approach to Design for Safety in Complex Systems”. INCOSE04, Toulouse, France, June 2004.

Appendix 1A: System-Level Hazards and Associated Severities

ID#	Phase	Hazard	Severity		
			H	M	Eq
G1	General	Flamable substance in presence of ignition source (Fire)	4	4	4
G2	General	Flamable substance in presence of ignition source in confined space (Explosion)	4	4	4
G3	General	Loss of life support (includes power, temperature, oxygen, air pressure, CO2, food, water, etc.)	4	4	4
G4	General	Crew injury or illness	4	4	1
G5	General	Solar or nuclear radiation exceeding safe levels	3	3	2
G6	General	Collision (Micrometeoroids, debris, with modules during rendezvous or separation maneuver, etc.)	4	4	4
G7	General	Loss of attitude control	4	4	4
G8	General	Engines do not ignite	4	4	2
PL1	Pre-Launch	Damage to Payload	2	3	3
PL2	Pre-Launch	Launch delay (due to weather, pre-launch test failures, etc.)	1	4	1
L1	Launch	Incorrect propulsion/trajectory/control during ascent	4	4	4
L2	Launch	Loss of structural integrity (due to aerodynamic loads, vibrations, etc)	4	4	4
L3	Launch	Incorrect stage separation	4	4	4
E1	EVA in Space	Lost in space	4	4	1
A1	Assembly	Incorrect propulsion/control during rendezvous	4	4	4
A2	Assembly	Inability to dock	1	4	3
A3	Assembly	Inability to achieve airlock during docking	1	4	3
A4	Assembly	Inability to undock	4	4	3
T1	In-Space Transfer	Incorrect propulsion/trajectory/control during course change burn	4	4	3
D1	Descent	Inability to undock	4	4	3
D2	Descent	Incorrect propulsion/trajectory/control during descent	4	4	4
D3	Descent	Loss of structural integrity (due to inadequate thermal control, aerodynamic loads, vibrations, etc)	4	4	4
A1	Ascent	Incorrect stage separation (including ascent module disconnecting from descent stage)	4	3	3
A2	Ascent	Incorrect propulsion/trajectory/control during ascent	4	3	3
A3	Ascent	Loss of structural integrity (due to aerodynamic loads, vibrations, etc)	4	3	3
S1	Surface Operations	Crew members stranded on M surface during EVA	4	3	3
S2	Surface Operations	Crew members lost on M surface during EVA	4	3	3
S3	Surface Operations	Equipment damage (including related to lunar dust)	2	3	3
NP1	Nuclear Power	Nuclear fuel released on earth surface	4	4	2
NP2	Nuclear Power	Insufficient power generation (reactor doesn't work)	4	3	3
NP3	Nuclear Power	Insufficient reactor cooling (leading to reactor meltdown)	4	3	3
RE1	Re-Entry	Inability to undock	4	3	3
RE2	Re-Entry	Incorrect propulsion/trajectory/control during descent	4	3	3
RE3	Re-Entry	Loss of structural integrity (due to inadequate thermal control, aerodynamic loads, vibrations, etc)	4	3	4
RE4	Re-Entry	Inclement weather	4	2	2

Appendix 1B: Snapshot of Hazard Mitigation Database/Tool

	A	B	C	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR
1		Hazard ID -->												
2		Hazard Name -->												
3		Design/Architecture Parameter	970	2	1	4	1	2	4	3	4	4	3	4
14		Nuclear Surface Power - Yes												
15		Nuclear Surface Power - No	1											
16		Level of Autonomy - High												
17		Level of Autonomy - Low	1											
18		Highly Elliptical Orbital Rendezvous - Yes												
19		Highly Elliptical Orbital Rendezvous - No	1					3	3	3				
20		Rendezvous in transit - Yes												
21		Rendezvous in transit - No	1					3	3	3				
22		Artificial gravity - Yes												
23		Artificial gravity - No	1											
24		High-closure ECLSS (H2O, O2) - Yes												
25		High-closure ECLSS (H2O, O2) - No	1											
26		Low boil-off propellant storage - Yes												
27		Low boil-off propellant storage - No	1	3										
28		In-space propellant transfer - Yes												
29		In-space propellant transfer - No	1	3										
30	Policy / Operational	HLLV - Yes				3	3		3	3				
31		HLLV - No	1			1	1							
32		Nuclear - Yes												
33		Nuclear - No	1											
62		Anything Prepositioned on M	1		1	1	1							
63		Nothing Prepositioned on M												
64		Prepositioning modules at Earth orbit - Yes												
65		Prepositioning modules at Earth Orbit - No	1											
66	Docking/Undocking	Need to Dock at Earth Orbit upon Return - NO	1					3	3	3				3
67		Need to Dock at M Orbit Inbound- NO	1					3	3	3				3
68		Need to Dock at M Orbit Outbound- NO						3	3	3				
69		Need to Undock at Earth Orbit upon Return- NO		2							3	3	3	
70		Need to Undock at M Orbit - NO		3							3	3	3	
71	Human/Cargo Couple	Transfer Together												