National Aeronautics and
Space Administration

SLS-RPT-087-02

DRAFT FOR PDR

RELEASE DATE: TBD

Draft Maturity 70%
May 22, 2013

SPACE LAUNCH SYSTEM PROGRAM (SLSP) INTEGRATED MISSION AND FAULT MANAGEMENT (M&FM) DESIGN ANALYSIS AND PERFORMANCE ASSESSMENT VOLUME 2: SLS GOAL TREE/SUCCESS TREE (GT/ST)

VERSION AND HISTORY PAGE

Status	Version No.	Change No.	Description	Release Date
Draft	-	-	Initial Draft of SLS DRD 1406VM-018 for SLS PDR, MSFC 4511 Concurrence on File	05/22/2013 Draft

NOTE: Updates to this document, as released by numbered changes (Change XXX), are identified by a black bar on the right margin.

TABLE OF CONTENTS

PARAGRAPH **PAGE**

APPENDIX

TABLE

FIGURE

1.0 INTRODUCTION

A Goal Tree/Success Tree (GTST) is a hierarchical, branched representation of a system's goals and objectives, specified in success space. By success space, we mean that all lower-level goals must be achieved for their respective higher-level goals to be achieved. The Mission and Fault Management (M&FM) group, an organization within the Vehicle Management (VM) team of the Space Launch System (SLS) Program, uses the GTST to represent and assess the means by which potential Fault Management (FM) designs can be used to protect SLS goals and objectives. To do this, it combines classical functional decomposition methods with Goal Tree methods to identify and use system state variables and their relationships to represent the physics and the physical and logical variables to be controlled to achieve system goals. Within the GTST structure, potential and actual FM goals are identified and assessed with respect to SLS system goals. These assessments include evaluations of abort trigger detection coverage, abort condition and trigger gaps and overlaps, abort scenario identification, and assignment of abort conditions and triggers to abort scenarios. Other potential FM goals that could employ the GTST assessment include Caution and Warning and Redundancy Management.

1.1 Purpose

The purpose of this document is to explain the structure of the GTST Model, the methodology employed to use the GTST Model, and results of using the GTST Model. This includes a summary of the analysis results using the Model, and the key Model design and implementation issues and decisions addressed in its development and usage. This document does not intend to provide an exhaustive description of the GTST Model, but rather to provide understanding of the Model, and to enable its current and potential future use. This document also provides instructions for how to download and view the Model directly.

1.2 Scope

This document refers specifically to the SLS M&FM GTST Model delivered for SLS Program Preliminary Design Review (PDR). The Model currently represents the SLS Block 1 vehicle configuration SLS-10003, from launch tower clearance through the Core Stage Only Ascent, up to but not including separation of the Core Stage from the Interim Cryogenic Propulsion Stage (ICPS). The GTST Model contains some references to the Multi-Purpose Crew Vehicle (MPCV) in terms of SLS goals and potential FM goals that refer to supporting MPCV goals, such as crew safety and delivering the MPCV to the proper orbit. However, no details are provided for MPCV goals, beyond SLS goals needed to support them. The following describe the technical contents of this document.

- Section 3 introduces the GTST's purpose, relationships to other SLS tasks, and key implementation issues.
- Section 4 presents a top-level description of the GTST Model, and access instructions.

- Section 5 provides more detail of lower levels of the GTST Model, with descriptions of major modeling features and issues
- Section 6 describes how the GTST Model was used to contribute to M&FM analyses.
- Section 7 describes potential updates to the GTST Model and to this Report for the SLS Program Critical Design Review (CDR).
- Appendices describe acronyms, glossary, open work, historical origins and external references for the GTST.

1.3 Change Authority/Responsibility

The NASA Office of Primary Responsibility (OPR) for this document is the Integrated Systems Health Management and Automation Branch/EV43 within the Spacecraft and Vehicle Systems Department, Engineering Directorate of the Marshall Space Flight Center (MSFC)..

Changes to this document shall be controlled by the SLS VM discipline, and processed in accordance with SLS-PLAN-008, SLSP Configuration Management Plan.

2.0 DOCUMENTS

2.1 Applicable Documents

The following documents include specifications, models, standards, guidelines, handbooks, and other special publications. The documents listed in this paragraph are applicable to the extent specified herein.

SLS-MDL-068 Revision A	Space Launch Systems Program (SLSP) Vehicle Functional Analysis Model (VFAM)
SLS-PLAN-008 Revision B	Space Launch Systems Program (SLSP) Configuration Management Plan (CMP)
SLS-PLAN-085 Version 1	Space Launch Systems Program (SLSP) Fault Management (FM) Plan
SLS-RPT-072 Revision A	Space Launch Systems Program (SLSP) Vehicle Functional Analysis Report (VFAR)
SLS-RPT-087-01 Version 1	Space Launch Systems Program (SLSP) Mission and Fault Management (M&FM) Design Analysis and Performance Assessment (DAPA), Volume 1: Monitored Conditions Report (MCR)
SLS-RPT-087-03 Version 1	Space Launch Systems Program (SLSP) Mission and Fault Management (M&FM) Design Analysis and Performance Assessment (DAPA), Volume 3: Abort Triggers Package
SLS-STD-038 Baseline	Space Launch Systems Program (SLSP) Modeling and Simulation Standard

2.2 Reference Documents

The following documents contain supplemental information to guide the user in the application of this document.

Dardenne, A., S. Ficklas, and A. van Lamsweerde (1991). ‘Goal-Directed Concept Acquisition in Requirements Elicitation,” *Proceedings of IWSSD-6 – 6th International Workshop on Software Specification and Design*, Como, 14–21.

Friedenthal, Sanford, Alan Moore and Rick Steiner. A Practical Guide to SysML: The Systems Modeling Language, 2nd ed. Morgan Kaufman, 2011.

Holt, J. and S. Perry. *SysML for Systems Engineering*. IET, 2007.

INCOSE Systems Engineering Handbook. Version 3.2.1.

Ingham, M., Rasmussen, R., Bennett, M., and Moncada, A., "Engineering Complex Embedded Systems with State Analysis and the Mission Data System", *AIAA Journal of Aerospace Computing, Information and Communication*, Vol. 2, No. 12, Dec. 2005, pp. 507–536.

Johnson, Stephen B., et al., eds. *System Health Management: with Aerospace Applications*. Chichester, UK: John Wiley & Sons, 2011.

Kim, I.S., M. Modarres (1986). *Application of Goal Tree-Success Tree Model as the Knowledge-Base of Operator Advisory Systems*. Chemical Process Systems Engineering Laboratory, Department of Chemical and Nuclear Engineering, University of Maryland. Submitted to *Nuclear Engineering and Design Journal*, October 1986.

Modarres, Mohammad, and Se Woo Cheon (1999). "Function-Centered Modeling of Engineering Systems using Goal Tree-Success Tree Technique and Functional Primitives," *Reliability and Safety Engineering* 64, 181–200.

Mylapoulos, J., L. Chung, and B. Nixon (1992). 'Representing and Using Nonfunctional Requirements: A Process-Oriented Approach', *IEEE Transactions on Software Engineering*, Vol 18, No. 6, June, 483–497.

NASA SLS GTST SysML Conventions Power Point Slides (GTST Conventions 8-27-12.pptx)

Nilsson, N.J. (1971). *Problem Solving Methods in Artificial Intelligence*. McGraw-Hill.

Van Lamsweerde, Axel. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Chichester, UK: John Wiley & Sons, 2009.

Weilkiens, Tim. *Systems Engineering with SysML/UML: Modeling, Analysis, Design*. MK/OMG Press, 2008.

3.0 GTST PURPOSE, TASK & TOOL RELATIONSHIPS, AND IMPLEMENTATION ISSUES

3.1 GTST Purposes

The PDR GTST has several purposes:

- To aid top-down assessment of the detection coverage of potential abort triggers against SLS and integrated stack system goals;
- To aid in the understanding of the physical and logical relationship of abort conditions to each other, including gaps and overlaps with respect to system goals and functions;
- To assess whether the Abort Conditions and Abort Triggers (AC/AT) List is missing abort conditions or abort triggers;
- To support the mapping of abort triggers to failure scenario-Loss of Mission Environment (FS-LOMEs) combinations in the Abort Condition Detection Hardware Study matrix;
- To support development of the AC/AT List by linking abort conditions and abort triggers to the state variables that capture the behaviors described by the abort conditions, and monitored by the abort triggers.

These uses are described in detail in the analysis Section 6. Sections 4 and 5 describe the model, which must be understood before the user can understand how it is used. The remainder of section 3 describes GTST's relationship to other tasks on SLS, issues that had to be resolved in GTST implementation on SLS and their resulting implementation as modeling conventions. Appendix TBS describes other GTST-like approaches outside of SLS.

3.2 GTST Relationships to Other SLS Tasks

The GTST is implemented currently by the M&FM organization within Vehicle Management of SLS. Its initial use for PDR is explicitly targeted to support M&FM internal analysis tasks. Specifically, it supports the following M&FM tasks and analyses:

- Abort Condition and Abort Triggers List
- Safety Net Abort Trigger Detection Coverage Analysis
- Abort Effectiveness Analysis (supporting Safety and Mission Assurance (S&MA) Probabilistic Risk Assessment (PRA) Loss of Crew (LOC) calculations)
- Abort Condition Detection Hardware (ACDH) Study

Support to these tasks is described in Section 6 of this report.

Potential support to future tasks, both internal and external to M&FM, is described in Section 7 of this report.

3.3 GTST Implementation Issues and Decisions

The GTST model was implemented in the Systems Modeling Language (SysML) language due to its customization characteristics. These characteristics were essential to the GTST modeling effort, because standard SysML stereotypes did not provide the representations needed to concisely express all relevant GTST information. The GTST team therefore created new stereotypes, and in so doing, discovered a few issues with SysML's capabilities. The GTST model required not only new stereotypes, but an entire framework of modeling conventions, model views, and stereotypes to provide an effective representation of the SLS state-based GTST.

3.3.1 GTST Goal Implementation

The GTST model is a logical breakdown of the system's goals. There are two major goal types: nominal and off-nominal. Nominal goals define the range of a goal's output state variables, while off-nominal goals define new objectives required if the nominal range is not met. In effect, goals are requirements, and thus the requirement artifact in the SysML language was used to denote a goal. Using the SysML stereotype capability, off-nominal and nominal goals are described as shown in Figures 3-1 and 3-2. Although GTST goals are "requirements" in the conceptual sense, and use the SysML requirements capability, these are not official SLS requirements and do not require verification and validation.

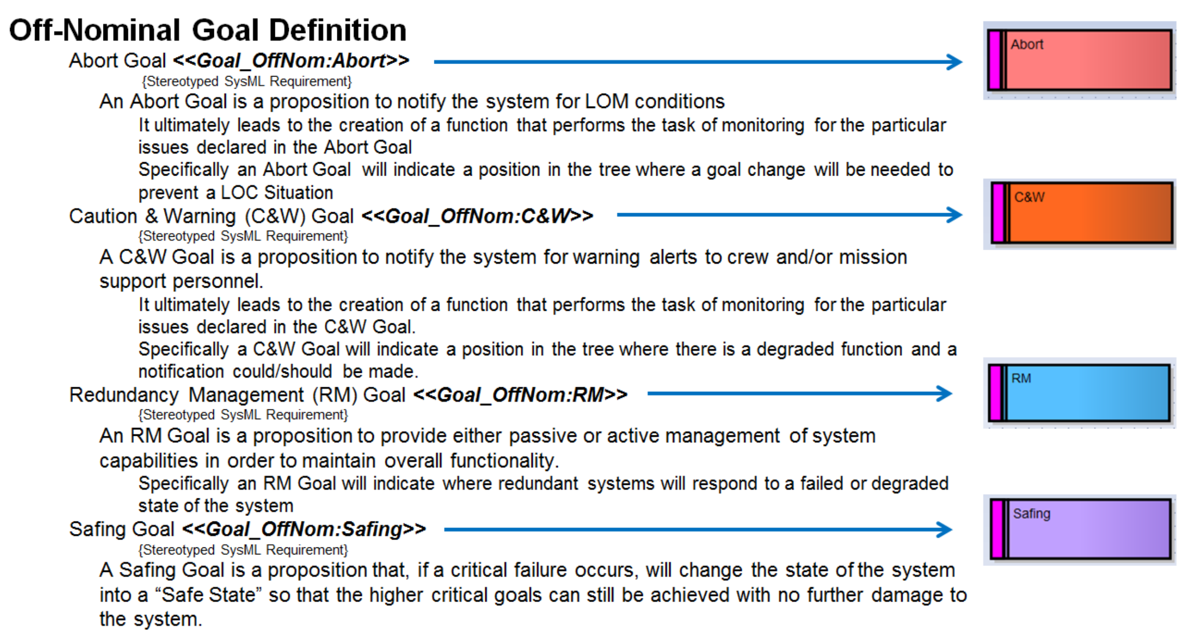

Figure 3-1. Off-Nominal Goal Definition

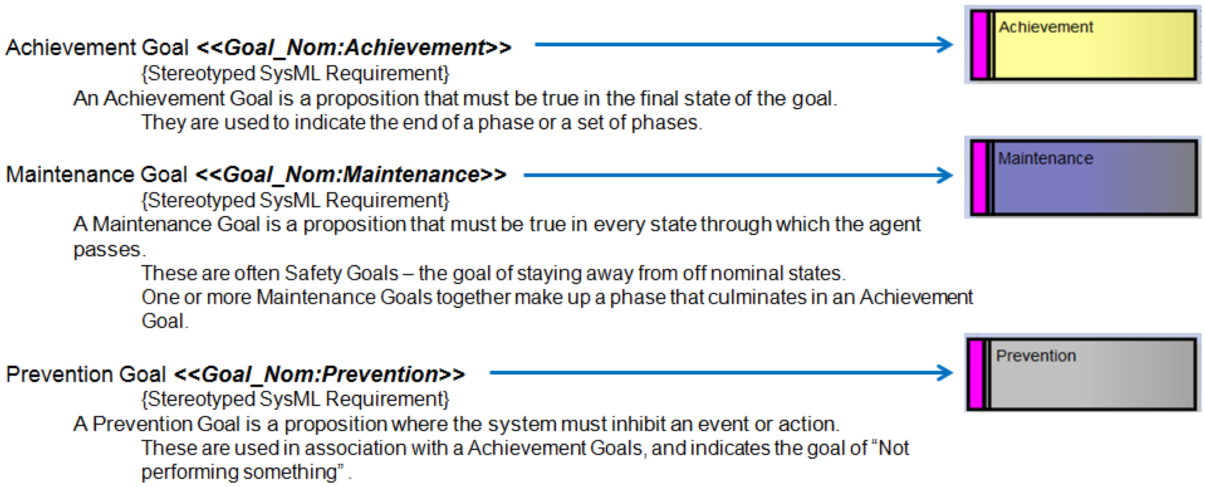

Figure 3-2. Nominal Goal Definition

3.3.2 GTST Function Implementation

Within the GTST modeling framework, there is a distinct difference between function space and goal space. Goal space is where goals are defined and decomposed, while function space is where functions defined and decomposed. In particular, in function space the GTST represents the change from one state variable, or set of state variables, into another through the function, as in the equation $y = f(x)$ where x and y are the input and output state variables, respectively, of the function f. The GTST uses SysML "Activity" artifacts as functions. Both nominal and off-nominal functions are used within the GTST. Nominal functions are non-stereotyped SysML activity artifacts and off-nominal functions are stereotyped SysML activity artifacts. (Note: Stereotypes are SysML mechanisms to customize objects and relationships.) Both nominal and off-nominal functions are defined in Figure 3-3.

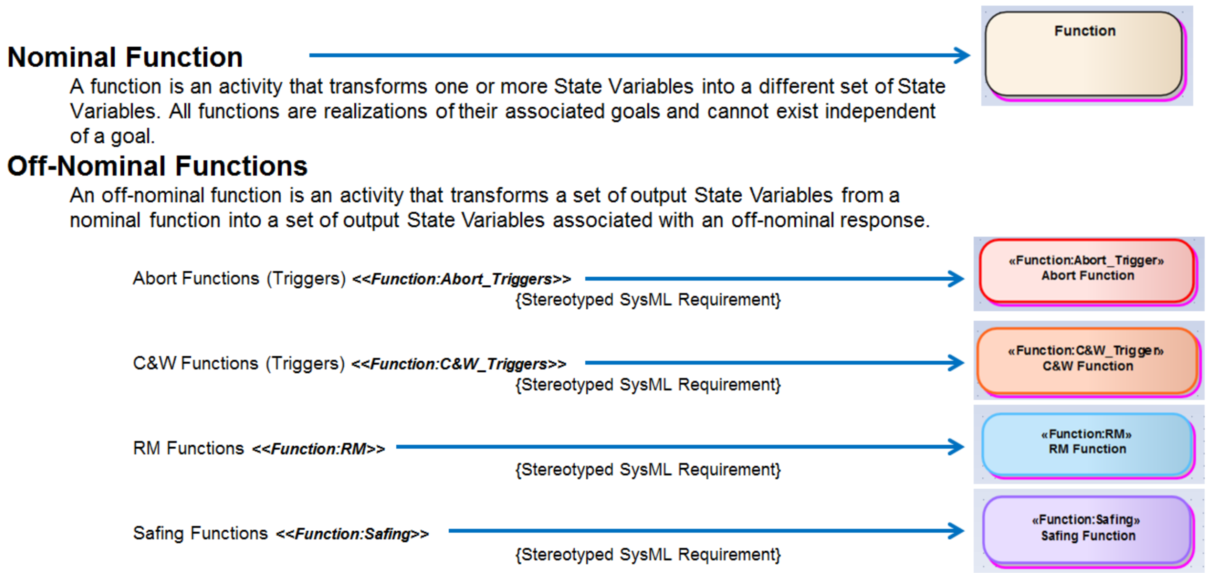

Figure 3-3. Nominal and Off-Nominal Functions

3.3.3 GTST State Variable/Vector Implementation

The GTST framework relies heavily on the definition of "State Variables" and how they connect physical and functional space. A State Variable is defined as a physical attribute of a system that must be maintained within an appropriate range for the success of the assigned goal. This heavy reliance meant that it was imperative to determine an approach in SysML to properly describe the state variable and how it relates to the rest of the model. This became an issue because of the difference in how the SysML language and the GTST framework approach the concept of the state variable. In the GTST framework, a state variable is considered a "thing" while in the SysML language that same concept would be an "attribute of a thing."

To make a state variable a "thing" instead of an "attribute of a thing," we established a new stereotype, a SysML Class artifact identified as a <<***Class:State_Variable***>>. This new class allows the GTST model to actively create and maintain differing levels of state variables and link them together consistently within the rest of the model. This also allows for the creation of Objects, SysML artifacts used to denote a specific instance of a state variable, stereotyped as <<***Object:State_Variable***>>. In addition to having to deal with single state variables it became apparent that the highest points in the tree had multiple state variables allocated to a single construct. To conserve diagram space and logically combine variables for decomposition reasons "State Vectors" were created. State Vectors are nothing more than a collection of state variables and both are represented in the GTST as Classes, as <<***Class:State_Vector***>> , and Objects, as <<***Object:State_Vector***>>, for the same reasons described previously. Class and Object State Variables and Vectors are further described in the Figure 3-4 below.

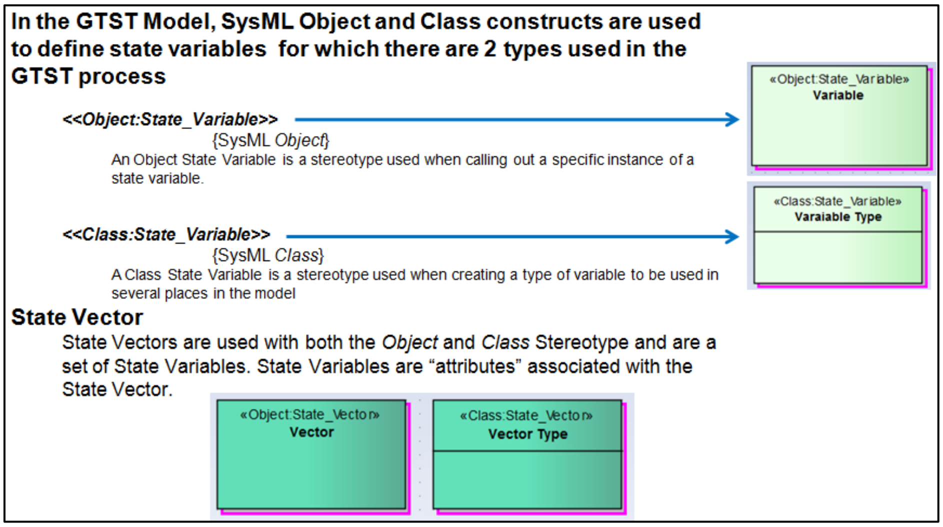

Figure 3-4. State Variable and Vector Definitions

3.3.4 GTST Status Implementation

Due to the graphical nature of SysML, it was determined early on that there needed to be a way to determine the development status of a particular artifact and whether that artifact had been deployed in the model as a placeholder, reviewed internally by the development team, or reviewed by external sources. Independent of the SysML language, the Enterprise Architect (EA) tool that was used to develop the GTST model and process provided a way to graphically display the status of a particular modeling artifact. The status definitions defined in the GTST to this point are shown below.

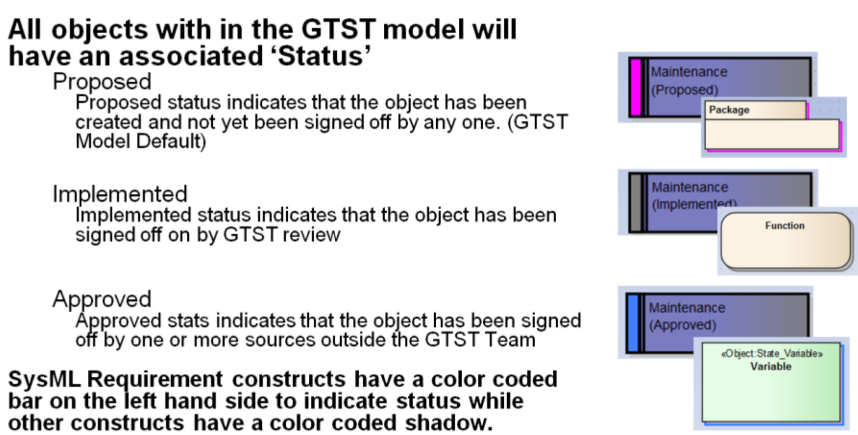

Figure 3-5. Status Definitions

4.0 GTST MODEL OVERVIEW AND ONLINE ACCESS INSTRUCTIONS

4.1 GTST Model Overview

The GTST development process is split into two sections: nominal and off-nominal. The nominal development process starts in the Goal Breakdown View with the creation of a goal, as explained in section 4.2.1. Once a goal is created in the Goal Breakdown View, its applicable state variables and vectors must be created. First, within the Model Library, the appropriate state variable and vector types are created as explained in section 4.2.4. Then, within the State Vector Breakdown View, the state variables and vector objects are created. The process for creating state variables and vector objects are explained in section 4.2.3. With the state variable and vectors created, they can now be assigned to the goal created within the Goal Breakdown View.

Next, within the Goal Breakdown View, the next level of goals to be created must be determined and placed within the model. If the state variables have been decomposed to their lowest level, and new state variables and/or vectors need to be created, then an elaboration is performed. An elaboration is a transformation of the state vector and indicates that one or more new state variables are needed to continue development of the tree along the particular path. These state variables are new only to this path; they could already exist elsewhere in the model. Corresponding new functions are added to transform the new state variables into the higher-level existing state variables. A goal elaboration within the GTST framework is depicted as a specialized SysML Decomposition line stereotyped as <<Elaboration:Goal>> and is depicted in the figure below.

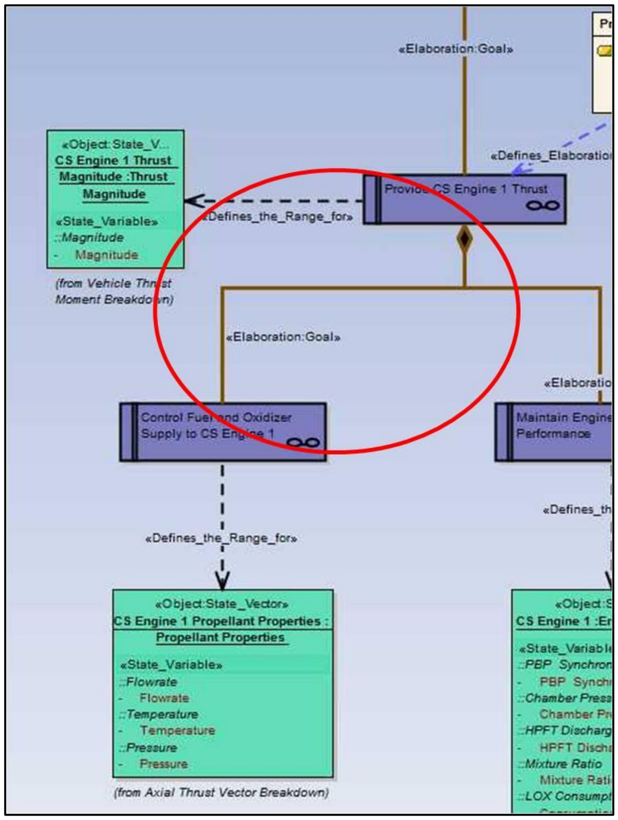

Figure 4-1. Goal Elaboration

Additionally, if an elaboration is needed after assigning the new state variable and/or vector to the elaborated goal, a function package is created, connected and deployed as described in section 4.2.2. After the function package and activity diagram are created, the elaborated goals and state variables/vector objects are deployed along with a newly created function and all of the objects are connected together. The diagram below shows this nominal GTST process.

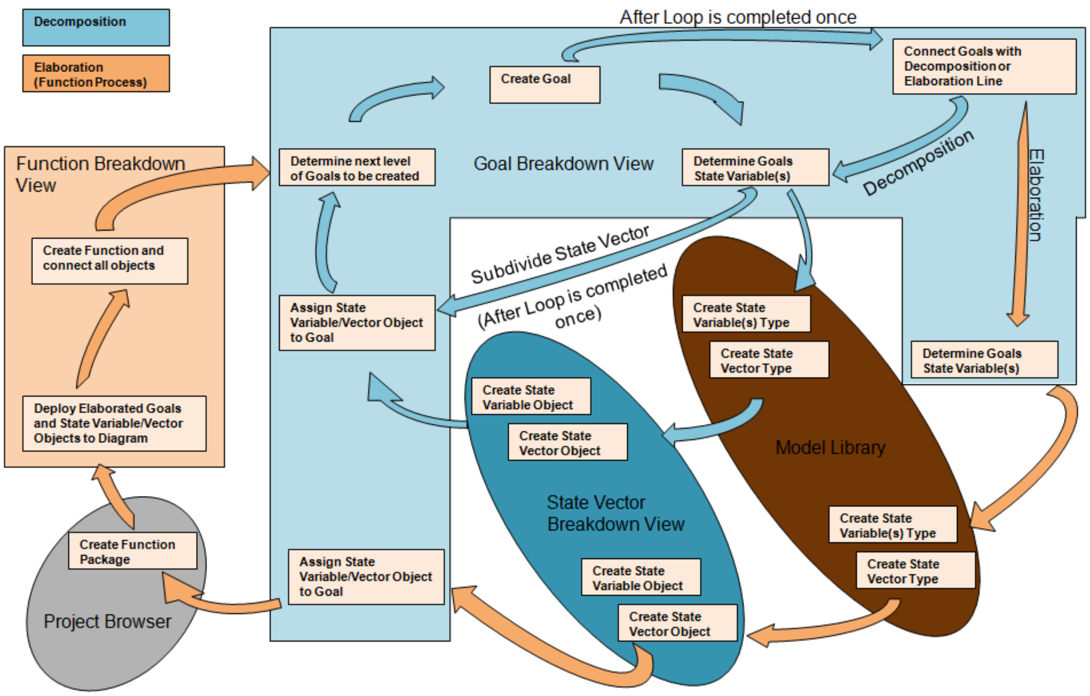

Figure 4-2. Nominal GTST Process

If the state vector is evaluated and determined that its state variables can be partitioned into subset state vectors without introducing any new state variables, then a goal decomposition is appropriate. A goal decomposition is a segmentation into a complete set of sub goals necessary to achieve the higher level goal. A goal decomposition within the GTST framework is depicted as a standard SysML decomposition line stereotyped as <<Decomposition:Goal>> and is depicted in the figure below.

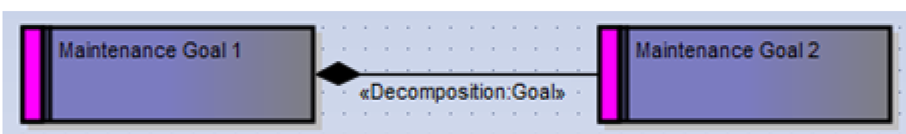

Figure 4-3. Goal Decomposition

For the Off-Nominal GTST Process additional steps are needed. If an off-nominal goal is determined to be needed an aggregation connection is used to link the off-nominal goal to its nominal goal within the Goal Breakdown view. A goal aggregation is used to indicate that there is a goal to monitor the attributes of a state vector within some predefined off-nominal range.

The goal aggregation is only used to aggregate the off-nominal goal within the lower level goals. This connection is a specialized SysML Aggregation line stereotyped as <<Aggregation:Goal>> and shown in the figure below.

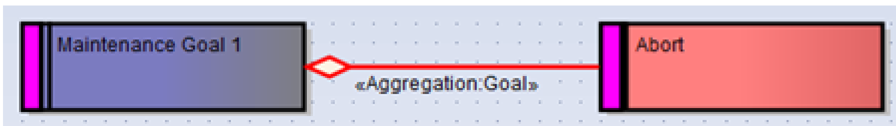

Figure 4-4. Goal Aggegation

Once the off-nominal goal is connected to the nominal goal, it is then deployed within the elaboration diagram in the function breakdown view. Additionally, an off-nominal function is created and connections are made between the off-nominal goal and off-nominal function, the elaborated goal's state variable/vector and the off-nominal function, and the off-nominal goal and the elaborated nominal goal. This procedure is described in section 4.2.2.2, and the off-nominal GTST Process is shown in the figure below.

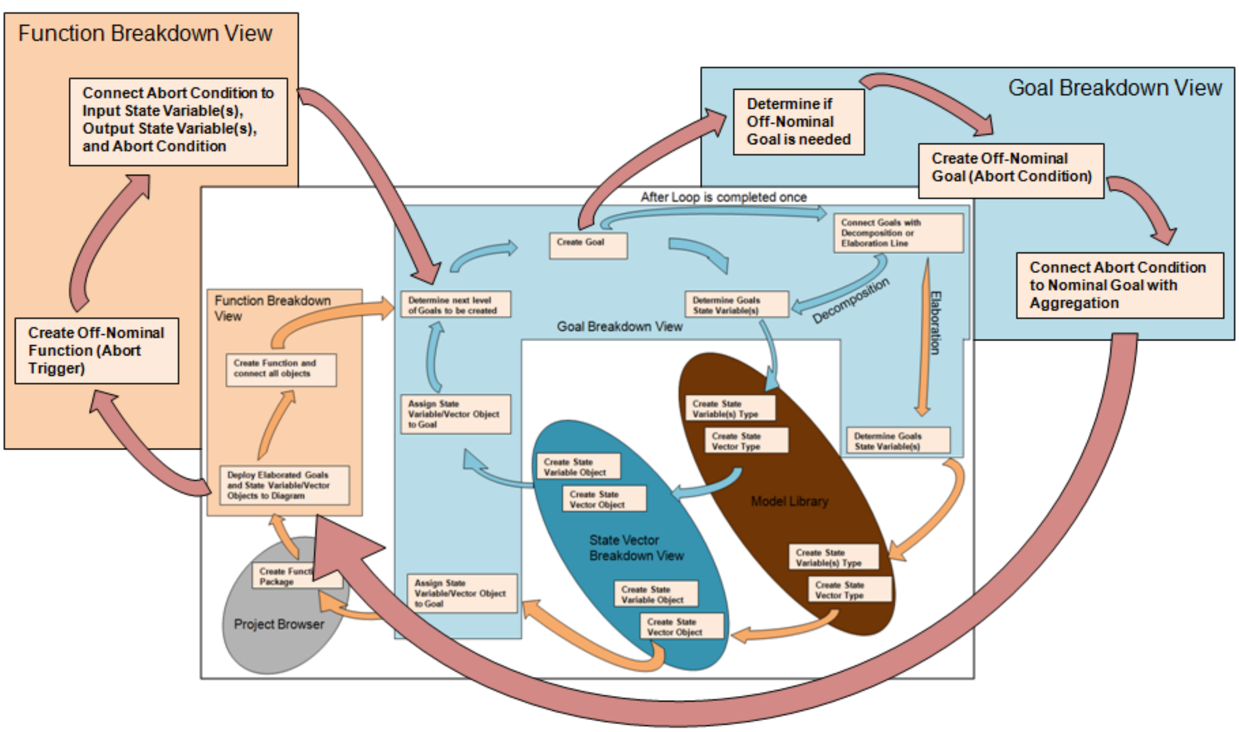

Figure 4-5. Off-Nominal GTST Process

4.2 Model Views

There are three main views associated with the GTST model. They are the Goal Breakdown View, the Functional Breakdown View, and the State Vector Breakdown View. Each of these views is explained in the sections below.

4.2.1 Goal Breakdown View

The Goal Breakdown View contains all of the goals for the GTST model and only contains SysML requirement diagrams. There are five packages associated with the Goal Breakdown View. They are the Top-Level Goal Breakdown, the Mission Success Goal Breakdown, the Crew Safety Goal Breakdown, the Abort, and the Avionic-Enabled Function packages. The Top-Level Goal Breakdown Package contains the top-level goals and diagrams that are detailed in section 5.1. The Mission Success Goal Breakdown Package contains each of the mission phases listed in section 5.1 as separate packages so that each of the modeling teams may work on their individual sections of the model. Each mission phase package is detailed in sections 5.2 - 5.5. The Crew Safety Goal Breakdown Package will hold the specific crew safety goals, their diagrams, and their breakdowns. The Abort Package will hold all of the abort conditions goals, their diagrams, and their appropriate breakdowns and internal/external connections. The Avionics Enabled Function Package will hold diagrams, connections between all of the avionics hardware goals and the functions that are enabled by them.

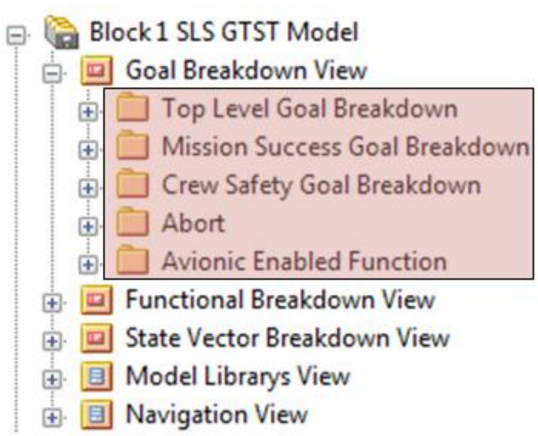

Figure 4-6. Goal Breakdown View

When a goal is made composite the resulting diagram will be a SysML Requirements Diagram. Each new diagram will have three levels of goal breakdown. Level one will be the goal that is being made composite. Level two will be the goals that are being elaborated and/or decomposed from the top-level goal. Level three will show the next-level goals that are being elaborated and/or decomposed. Packages and State Variables/Vectors will only be shown for level one and level two goals in each diagram.

4.2.2 Functional Breakdown View

The Functional Breakdown View will have a set of packages where each will be associated with one goal from the Goal Breakdown View and describe all of the elaboration connections of that goal. Each package will consist of a SysML Activity Diagram that contains the context of the elaboration detailing which state variable/vector is transformed by a particular function. All of the nominal functions will be created and maintained within the appropriate packages. All other modeling objects will be deployed from other parts of the model.

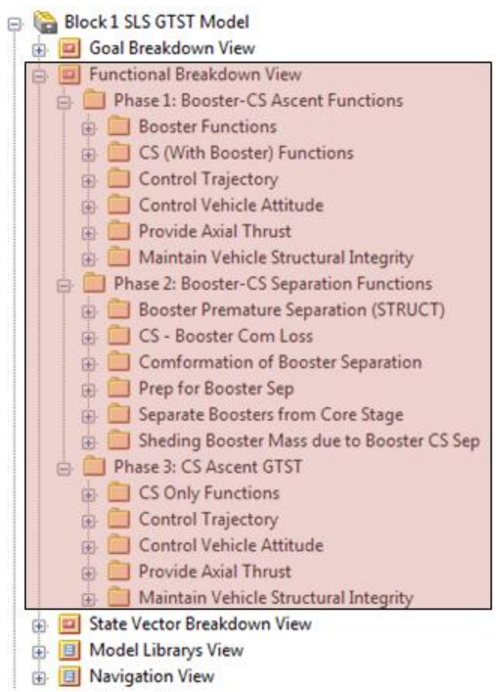

Figure 4-7. Functional Breakdown View

4.2.2.1 Functional Diagram Setup

Function Diagrams are set up using SysML Activity Diagrams and are used to show the links from goals to state variable/vector, state variable/vector to function, and function to goal. When a function package is created, deployed, and linked to a goal in a requirement diagram an activity diagram is used to show the context associated with the elaboration. These function packages are the same as the packages described in section 4.2.2.

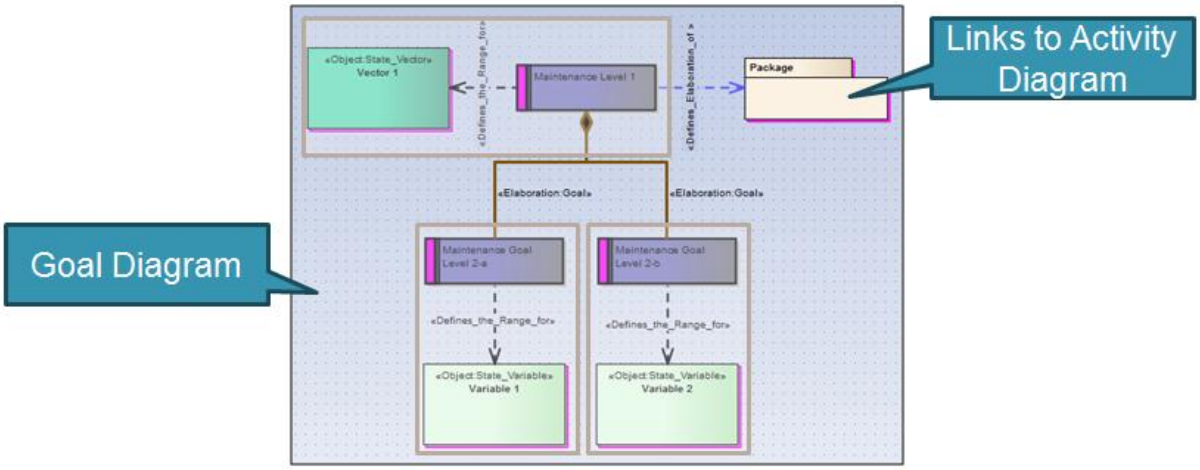

Figure 4-8. Function Package

Once the package is created and deployed in goal space, the new activity diagram can be utilized. Any lower level goal and its associated state vector/variable, which is linked with an elaboration line to the higher level goals, must be deployed on the Activity Diagram, along with a function to transform lower level state variables into higher level state variables as displayed in the figure below.

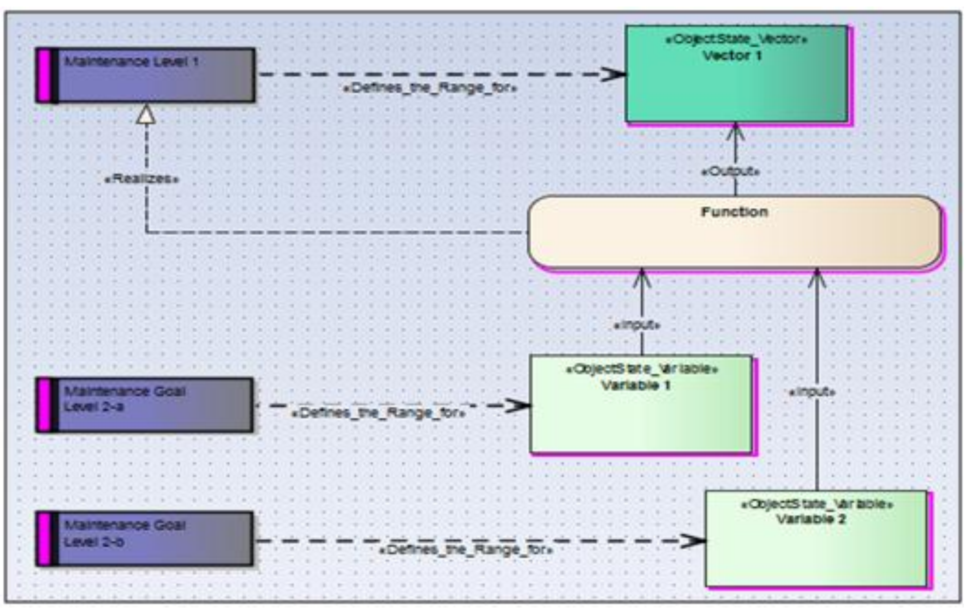

Figure 4-9. Function Activity Diagram

State variables/vectors are connected to functions using <<Input>> and <<Output>> stereotyped SysML Object Flow connections. This graphically indicates the variables/vectors that the function takes in and the variables/vectors that the function produces. Also a connection is made between the function and the top-level goal to show that there exists a function that associates the goal with its state variable/vector(s). This connection is a standard SysML Realization connection.

Additionally there are four special cases that occur within the GTST Function Diagrams that need to be discussed. They are: off-nominal function connections, state vector decompositions, abort trigger output state variables, and avionics connections. The conventions for abort trigger output state variables and avionics connections are forward work.

4.2.2.2 Off-Nominal Function Connections

Off-nominal function connections are used when there is a function package and an off-nominal goal connected to a nominal goal within the goal view of the model. Off-nominal goals are connected to nominal goals using a specialized SysML aggregation link stereotyped as <<Aggregation:Goal>>. On the function diagram the off-nominal goal is deployed with its associated off-nominal function. In the case of an abort goal this off-nominal function is the abort condition. The off-nominal function and goal are connected with a standard SysML realization link. Additionally the function's output state variables are connected to the off-nominal function with an <<Input>> SysML Object Flow connection to indicate that the off-nominal function will trigger from the measurement associated with that state variable/vector. The figure below shows both the goal view and the function view for this case.

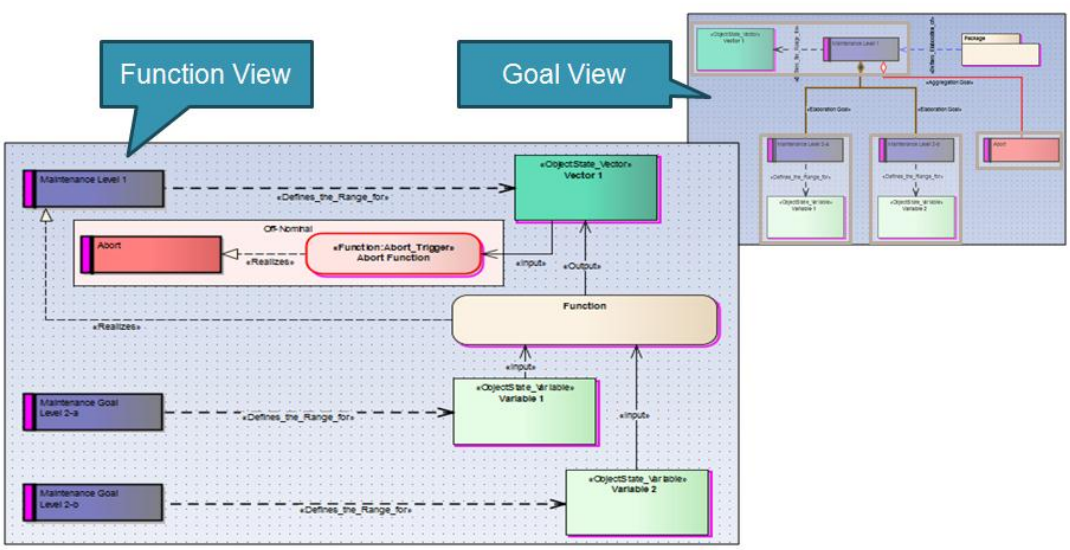

Figure 4-10. Off-Nominal Function Placement

4.2.2.3 State Vector Decomposition

State vector decomposition is sometimes needed to subdivide a state vector into its individual state variables or smaller, lower level state vectors. In the example shown in the figure below a vector is subdivided to show two different inputs into two different off-nominal functions. A specialized SysML stereotyped decomposition link, <<Decomposition:State_Vector>> is used to connect the higher level state vector to its lower level state variable components. The decomposition of the state vector is not created in the function diagram but rather within diagrams created in the State Vector Breakdown View described in section 4.2.3.

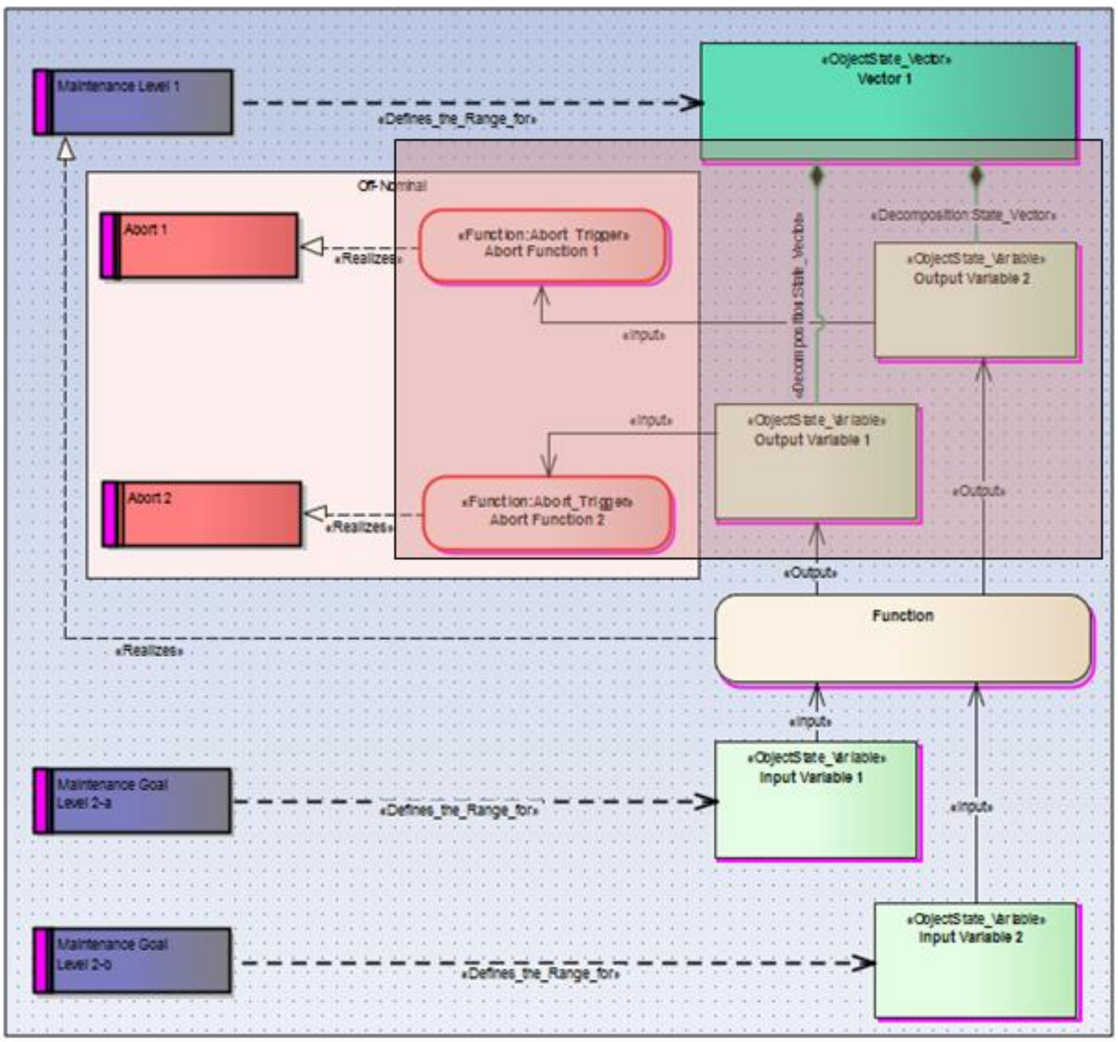

Figure 4-11. State Vector Decomposition

4.2.2.4 Abort Trigger Output State Variable

The abort trigger output state variable, which is slated as forward work in Version B GTST, is used to represent the notification of the off-nominal abort trigger function to an external source. This new state variable is connected to the off-nominal abort trigger function as shown in the diagram below. As this portion of the GTST framework is still under development its implementation and use is yet to be determined.

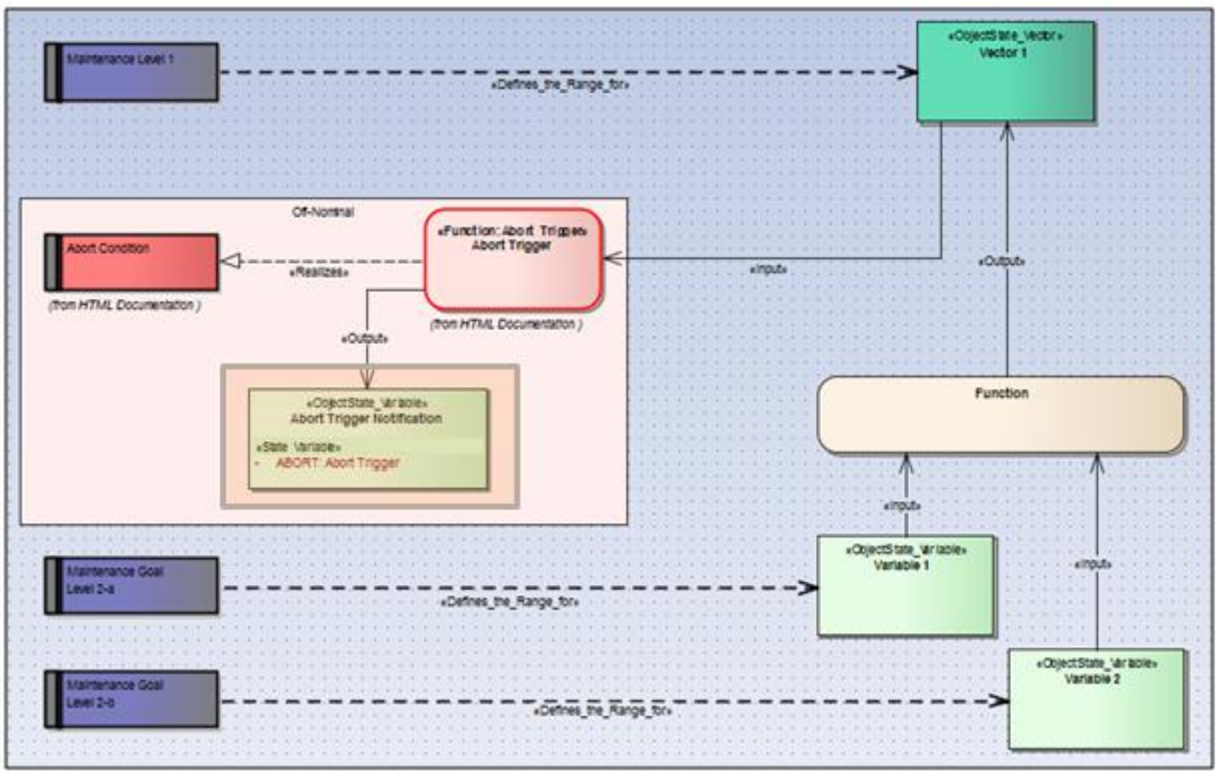

Figure 4-12. Abort Trigger Output State Variable

4.2.2.5 Avionics Connectivity

During the development of the GTST and based on prior Ares I assessments, it became apparent that there was a need to specify two types of functions, active and passive. Active functions are functions that require some kind of operation to be performed, such as actuating a valve or switching to another mode. Passive functions do not require any operation to be performed, such as transferring thrust from four engines through the thrust structure and providing total thrust of the vehicle. For the SLS vehicle, active functions are usually performed with avionics hardware and software; therefore a specialized SysML stereotype, <<Enables>>, was created to link an avionics hardware goal to an active function. This portion of the GTST framework is also still under development and its implementation and use are yet to be determined. The diagram below shows the current state of this particular part of the GTST framework.

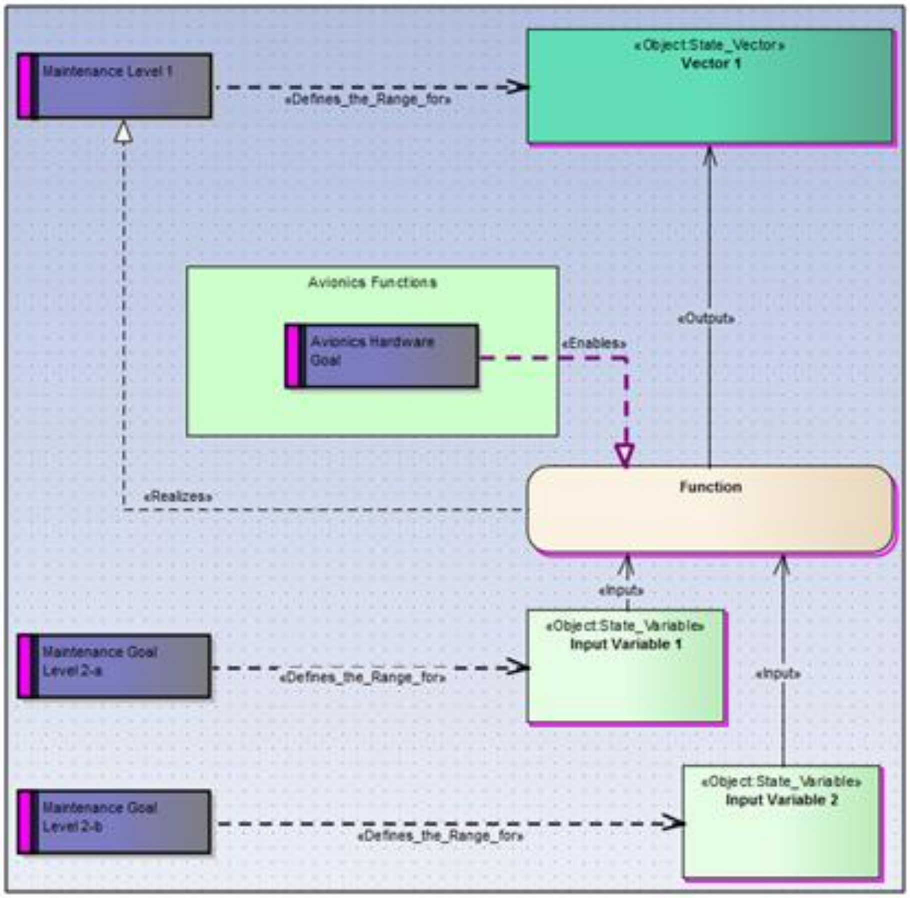

Figure 4-13. Avionics Goals and Active Functions

4.2.3 State Vector Breakdown View

The State Vector Breakdown View contains packages associated with the vehicle components such as; Main Propulsion System, TVC, Structure, and others. Each component package holds the state variable/vector objects, described in section 3.5.3 used in the goal and function diagrams. Each component package contains an activity diagram that depicts the decomposition and elaboration connections specific for each state variable/vector connection consistent with respect to the goal and function view. The state variables and vectors are maintained, individually, within the component packages. Any state vector or variable used in other views of the model are deployed from the state vector breakdown view.

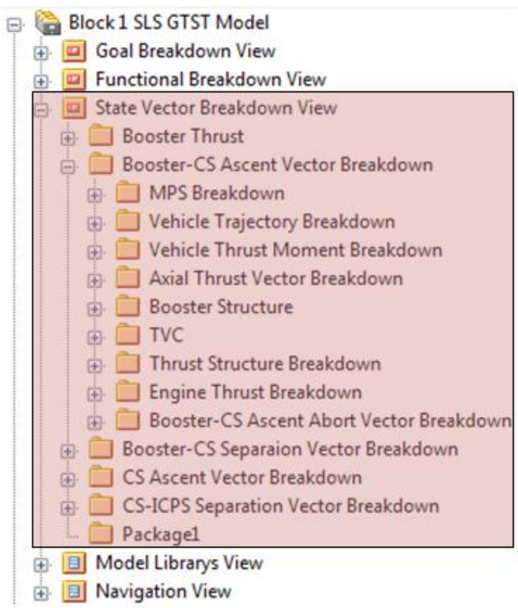

Figure 4-14. State Vector Breakdown View

Once a goal view has been created and all goals have been connected with elaboration and decomposition links, state variable and vector objects are created and linked to the goals. The figure below shows an example of how the state variables and vectors would be consistently maintained.

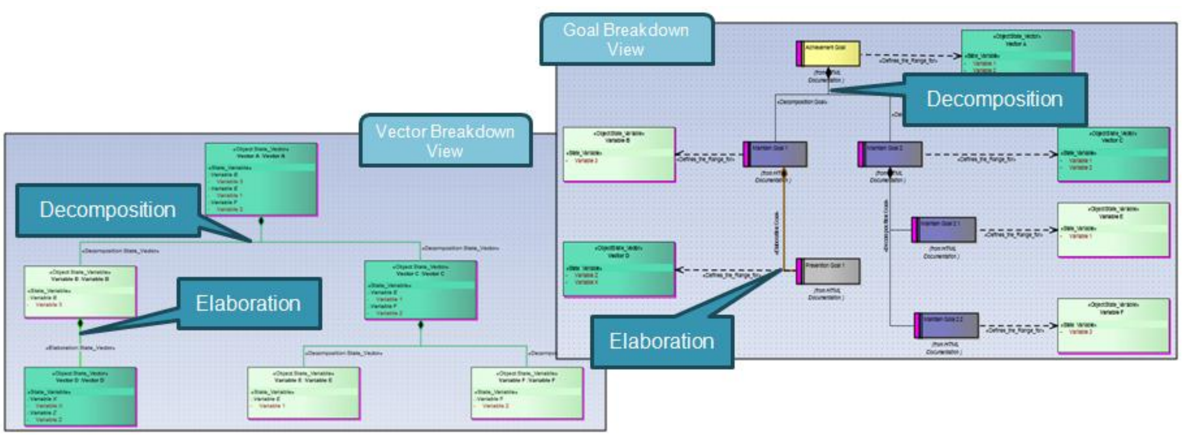

Figure 4-15. State Vector Breakdown View Detail

Two types of specialized stereotyped SysML Decomposition links are used: <<Decomposition:StateVector>> and <<Elaboration:StateVector>>. A state vector decomposition line shows the partitioning of one state vector into its sub state vectors. This partitioning can be performed down to the level where a state vector is eventually decomposed into its individual state variables. A state vector elaboration shows the connection between two state vectors that have been elaborated by a function. This is used when one or more new state variables are introduced that did not exist in the higher level state vector, in that particular path of the GTST.

4.2.4 Model Library View

The Model Library View has two high-level packages. The first, Modelers Library, holds important information about the people working in the model and is used to show who is working on a particular piece of the model. This is currently forward work. The second high-level package is the State Variable/Vector Type Library. This package is the place within the model that holds all of the State Variable/Vector Classes described in section 3.5.3. By creating types of state variables/vectors, the same kind of variables and vectors can be reused in multiple places in the model by utilizing the instant classifier option in EA.

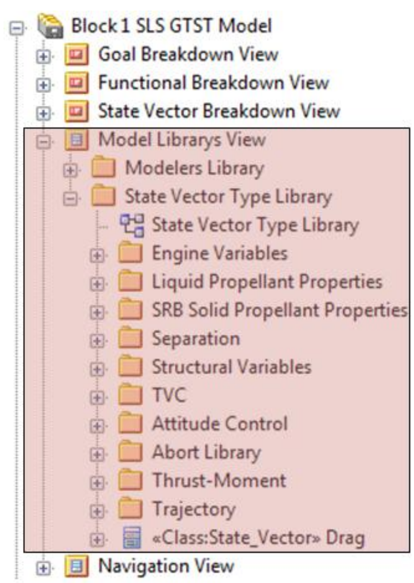

Figure 4-16. Model Library View

The Modelers Library package uses SysML actor constructs as an association to actual persons developing the model. The attributes of the actors (persons or organizations) are used for contact information. An Actor is linked to one or more packages that it is responsible for so that an integrated workflow picture is created. This part of the GTST framework is still in development and not fully implemented within Version B of the GTST and slated for forward work.

The State Vector Type Library, a package within the Model Library View, is subdivided into packages associated with groupings of state variable/vectors. SLS examples of this include but are not limited to Propellant, Structural, Trajectory, Thrust, and Control Properties. This information is design specific. Additionally each subpackage has its own block diagram that shows how the state variables/vectors are created. The <<Class:State_Variable>>, described in section 3.5.3, is created with a single attribute assigned to it representing the variable itself. Generally, the variable attribute name is the same as the variable class name. While this is permissible in EA SysML and seems to be useful in practice, it is not required. A Stereotyped SysML Generalization connection <<Inherit>> is used once all the variables are created to form a vector. The connection is created going **from** the vector, <<Class:State_Vector>> (See section 3.5.3), **to** the variables as shown in the figure below.

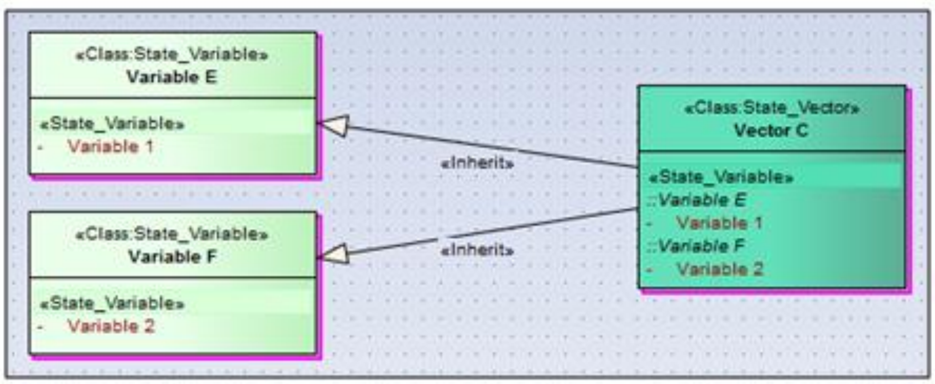

Figure 4-17. Inherit Connection Definition

The use of a class and an object artifact allows for the same variable to be used in association with many goals, but creates a unique definition of that variable's range to be used with a specific goal. For instance, an acceleration variable type can be classified as 'Vehicle' Acceleration, 'Engine' Acceleration, or 'Booster' Acceleration. This is done using the "Instance Classifier" option within the EA tool. However, EA does not provide the capability to automatically create continuity throughout the model and thus this must be done manually. The diagram below shows how that continuity would be maintained between the block diagrams of the Model Library and the activity diagrams of the State Vector Breakdown View.

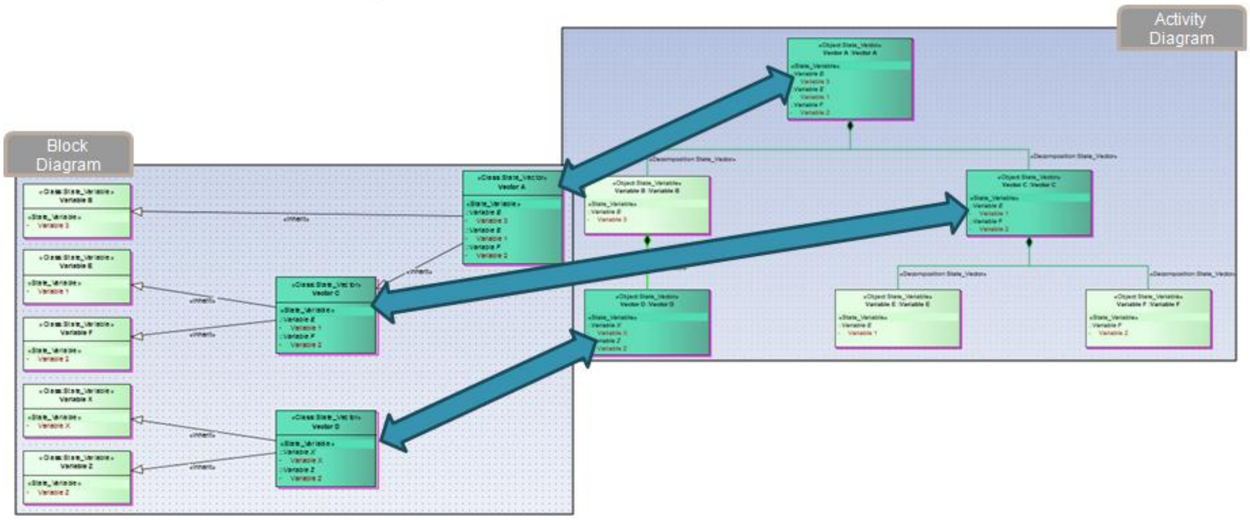

Figure 4-18. Class-Object Continuity Maintenance

4.2.5 Navigation Views

The Navigation View is used in two ways: to create a Hypertext Markup Language (HTML) document so that the model can be viewed outside of the EA tool, and to bookmark additional views that may be useful in the analysis performed on the GTST model. The diagrams of the model are configured in a way that allows for a link on every diagram that takes the user to the next highest level diagram. This model configuration is available in both the HTML and EA versions of the model. The links are available at the top left hand corner of the diagram; an example is provided in the figure below.

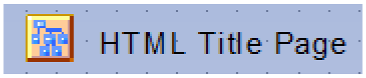

Figure 4-19. Link Example

Also within the Navigation View there is a package titled 'Top Level Goal Trees'. This package contains specific diagrams that have been put together for specific uses. The diagrams are: Abort, Avionics, Booster-CS Goal Tree, Staging Goal Tree, and CS Goal Tree. The Abort Diagram shows all of the abort goals and functions for quick access to all of the abort-related issues within the model. The Avionics Diagram shows all of the avionics goals and the active functions that they are connected to. The Booster-CS Goal Tree, the Staging Goal Tree, and the CS Goal Tree show the full goal tree, from top to bottom on one diagram, of that phase so that targeted analysis can be done.

4.3 Online Access Instructions

The SLS Block I GTST Version B can be accessed online at the MSFC document repository, Windchill. The hyperlink to the repository is found below:

GTST Version B (7-13-2012)

https://nasa-ice.nasa.gov/Windchill/app/#ptc1/tcomp/infoPage?ContainerOid=OR%3Awt.projmgmt.admin.Project2%3A26342144&oid=OR%3Awt.folder.SubFolder%3A164292509&u8=1

Within the folder linked above there are three files. The GTST_Version_B (7-13-2012) is the EA version of the model and requires EA to use and analyze the model. The GTST_Version_B_HTML (7-13-2012) is an HTML version of the model that anyone with Internet Explorer or other browser can use and analyze. Additionally there is a readme file for detailed instructions on opening the HTML version of the GTST.

4.4 Model Navigation Instructions and Tips

On each diagram within Version B of the GTST you will find three distinct breakdown levels. The top level specifies a single goal that is the “root” goal of the diagram. The name of the top-level goal is also the name of the diagram. The second level shows the immediate breakdown of that top-level goal. The third level simply provides information to help orient the user for navigation to the lower breakdown levels below level two of the current diagram. To navigate further down the tree, double click a goal on the second level to traverse to a lower level goal and diagram.

At the top of each page you will find a set of links used for navigation up the tree. Each link will take you to the next higher level diagram.

There are some situations where the tree 'forks.’ For example, there are situations in which a Loss of Control leads to Structural Breakup, but others in which Structural Breakup leads to Loss of Control. In these instances there will be multiple links instead of one, from the “top” of a given function or goal, leading to several different higher level goals. That is, the current GTST is not strictly constructed as a tree, in which any lower level goal leads to only one higher level goal. We did this to avoid replicating the same goal diagrams in several tree locations. While this reduces the number of diagrams, it creates some complexity in that one can have several paths up the tree from a given low level goal or function. It is logically equivalent to a tree in which there is only one path up the tree from a low-level goal, but with the same low-level goal repeated for each unique tree path.

Additionally, as described in section 4.2.5, there are diagrams that show the tree in its totality. These diagrams are consistent with the model and may be used to traverse the tree. These expanded tree views can be found within the navigation view of the model.

5.0 GTST MODELS STRUCTURE AND CONTENT

5.1 Model Top Levels

The GTST model shows the top-down decomposition of the Block I SLS Launch Vehicle. The model is constructed using the goal methodology described in section 3.5.1 of this document. Using the Vehicle Functional Analysis Model (VFAM) as a starting point, a set of top-level goals was defined to describe the vehicle at the highest level of abstraction. The list of top-level goals and their descriptions can be found in Table 5-1, and a graphical representation can be found in Figure 5-1. Note that Achievement and Maintenance goals are defined in Figure 3-2.

Table 5-1. Top Level SLS GTST Goals and Descriptions

Goal Name	Goal Type	Goal Description
Level 1		
Achieve Mission, Reliability, and Safety Goals	Achievement Goal	A goal used as a place holder to insure that all Level 2 Goals can be verified.
Level 2		
Provide Fleet Supportability Data to Ground	Maintenance Goal	A goal used as a place holder to ensure continuity for any potential lower level goals associated with Fleet Supportability and Ground.
Provide for Public Safety	Maintenance Goal	A goal used for detailing the Flight Termination system and any other lower level goals associated with Public Safety.
Maintain Crew Safety During Mission	Maintenance Goal	A goal used for detailing the decomposition of the lower level vehicle goals.
Level 3		
Deliver Crew and MPCV to Target Orbit Safely	Achievement Goal	A goal used to detail the decomposition of goals needed to provide for SLS, MPCV, and Crew needs to achieve mission success safely.
Perform Abort	Achievement Goal	A goal used to detail further decomposition of all abort related goals.
Level 4		
Deliver MPCV to Target Orbit	Achievement Goal	A goal used to detail the decomposition of goals associated with delivering the MPCV to orbit only.
Deliver Crew to Target Orbit Safely	Achievement Goal	A goal used to detail the decomposition of goals associated with keeping the crew safe during the mission.

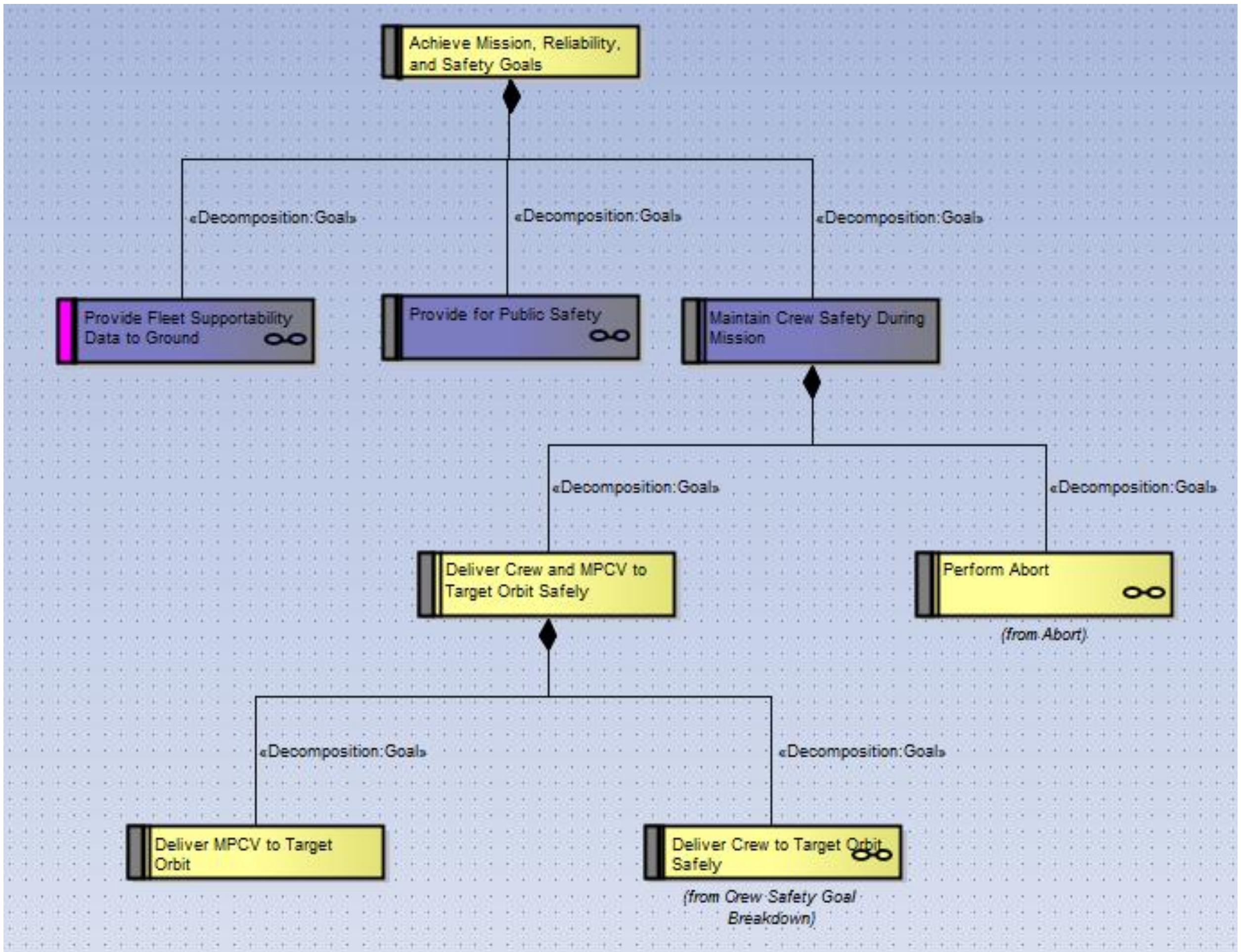

Figure 5-1. GTST Top Levels of Decomposition

Once the highest level goals were established, the detailed decomposition was developed. Most of the work done on the GTST took place on detailing the Table 5-1: Level 4 goal “Deliver MPCV to target Orbit”. This goal has been decomposed into each of the main five flight phases of the SLS Block I Vehicle; Booster/Core Stage Ascent, Booster/Core Stage Separation, Core Stage Only Ascent, Core Stage/ICPS Separation, and ICPS Ascent. Because each of these flight phases have a distinct set of state variables that must be within a particular range to be achieved, and each of the phases must take place within the order defined previously these goals are considered Achievement Goals. The list of goals and their phases of flight are in Table 5-2.

Table 5-2. SLS GTST Flight Phase Goals

Goal Name	Phase of Flight
Deliver Booster-CS to Booster Separation Point	Booster/Core Stage Ascent
Separate Boosters from Core Stage	Booster/Core Stage Separation
Deliver CS-ICPS to ICPS Separation Point	Core Stage Ascent
Separate Core Stage from ICPS	Core Stage/ICPS Separation
Deliver MPCV-ICPS to MPCV Separation Point	ICPS Ascent

Separate GTST trees could be constructed independently for each of these flight phase achievement goals. Three of these trees were constructed during this period and are described in the subsequent sections. Due to time constraints and lack of information about particular systems, the flight phases that are modeled in Version B of the GTST are Booster/Core Stage Ascent, Booster/Core Stage Separation, and Core Stage Ascent. The Core Stage/ICPS Separation and ICPS Ascent flight phases are scheduled for future work.

5.2 Booster-Core Ascent

The Booster-Core Ascent GTST represents that portion of SLS ascent that starts after the vehicle has cleared the tower and ends just before the Booster-Core separation gets initiated. Because of the size and complexity of the Booster-Core GTST, some simplifying "shortcuts" were taken. First, the Booster-Core GTST was split into two separate representations, each emphasizing the details of either the Booster's or the Core's contribution to the GTST and hiding the details of the other part. Another shortcut was to model only one of the two boosters in detail since the goals for the two boosters are in principal the same. The same is true for the Core Stage Engines (CSE) – the goals of only one CSE are modeled in detail. A third simplification involves showing relationships of certain goals to multiple goals on other branches rather than repeating those goals on each branch. As an example, Control Vehicle Attitude is not only a maintenance sub goal to the "Control Trajectory" goal but also to the "Maintain Structural Integrity" goal. In a strict tree structure, the "Control Vehicle Attitude" goal would be repeated for each of the applicable branches and in principle, different limits would apply for the different instances of the same goal, e.g. stricter rules could apply for preventing vehicle breakup than for staying on course.

"Deliver Booster-CS to Booster Separation Point" is the top-level achievement goal for the Booster-CS ascent phase of Flight. This achievement goal decomposes into two high-level maintenance goals, "Control Trajectory" and "Maintain Vehicle Structural Integrity." These maintenance goals define continuous and persistent functions performed throughout this phase of the mission. The "Control Trajectory" goal contains functions and goals to propel the vehicle along the planned flight path during this ascent period, while the "Maintain Vehicle Structural Integrity" goal contains those functions and goals that retain the vehicle composition and form.

Figure 5-2 displays the first two goal tiers beneath the “Deliver Booster-CS to Booster Separation Point” achievement goal.

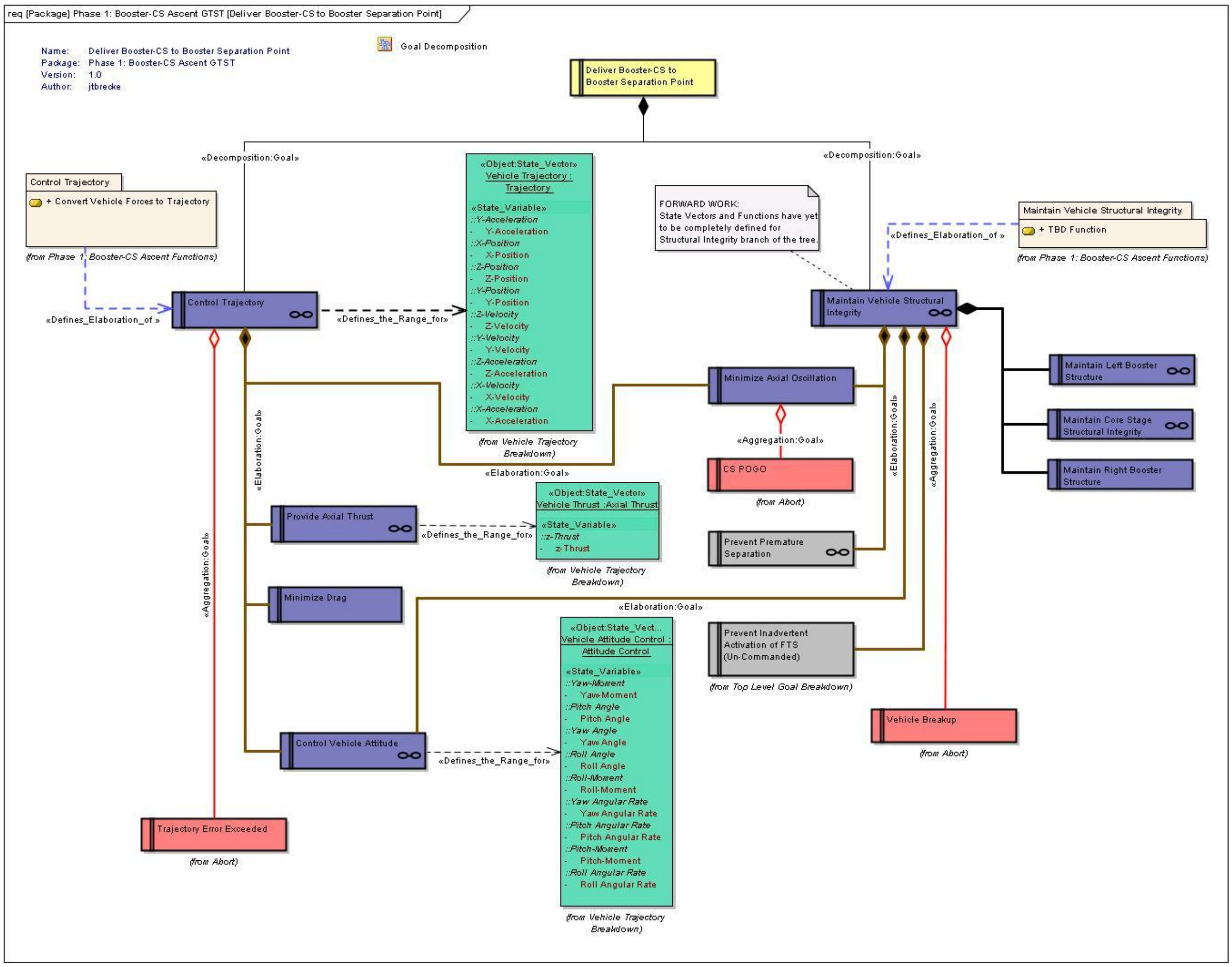

Figure 5-2. Top Tier of Booster-CS Ascent

The current GTST model identifies four maintenance goals that support the “Control Trajectory” goal: “Minimize Axial Oscillations”, “Minimize Drag”, “Provide Axial Thrust” and “Control Vehicle Attitude”. The details of the “Minimize Axial Oscillations” and “Minimize Drag” goals are yet to be determined, but both goals minimize negative conditions as the vehicle travels along the trajectory path. The state vector defined at the “Control Trajectory” goal, contains the position, velocity and acceleration at the vehicle center of mass, denoted by the capital letter variables: X, Y and Z. Within the lower level goals beneath the “Control Trajectory” goal, the state vectors include these parameters of position, velocity and acceleration relative to the local coordinate axis at each Booster motor or CSE center of mass. Therefore the function defined at the “Control Trajectory” goal, titled “Convert Vehicle Forces into Trajectory” combines and translates the forces and torques generated at the local engine level into the vehicle trajectory state vector. Figure 5-3 illustrates the GTST representation of this function.

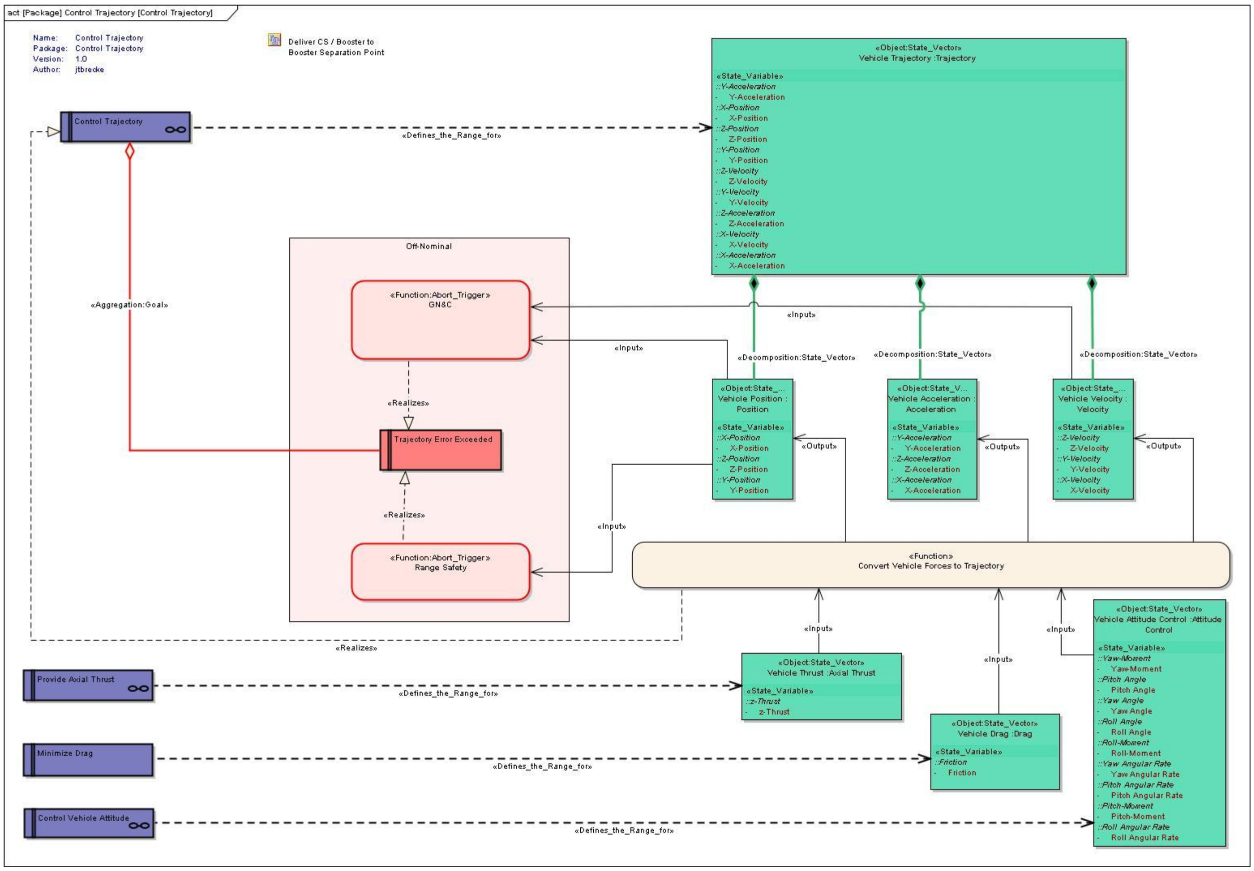

Figure 5-3. “Convert Vehicle Forces into Trajectory” Function

Note in Figure 5-3, the abort condition, “Trajectory Error Exceeded”, was identified and attached to the “Control Trajectory” goal. The condition is also represented within the defined function for the goal, shown in Figure 5-4, and located within the Off-Nominal area. The GN&C abort trigger function is defined to utilize the vehicle position and speed and compare to planned launch profile to determine if the error warrants an abort recommendation. At the same time, a second abort trigger is also defined from Range Safety, which looks at the current vehicle position and determines if it is beyond the safety envelope established for the flight.

The view of the GTST breakdown for the "Control Trajectory" goal, Figure 5-5, depicts the goals, functions and state variables defined under this portion of the GTST. It is important to note that there is a relationship between the control axial thrust and the control attitude goals. The TVC of each Booster motor and the CS engines affects the magnitude of the thrust vector along the vehicle's centerline axis and also controls the attitude of the vehicle. Therefore functions that support engine thrust generation and thrust vectoring control are common in both goals of "Provide Axial Thrust" and "Control Vehicle Attitude".

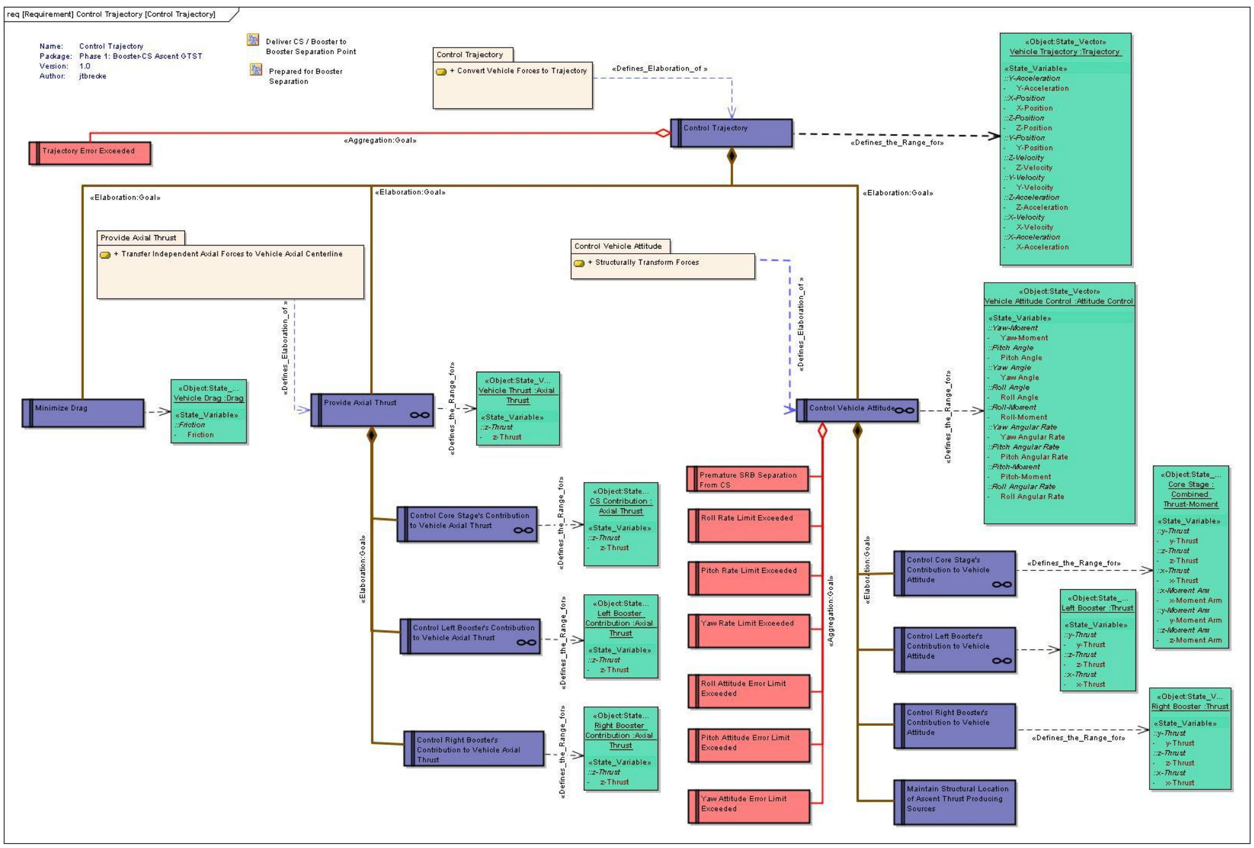

Figure 5-4. "Control Trajectory" Goal Decomposition

The "Provide Axial Thrust" goal is the combination of the axial thrust contributions from each of the vehicle's primary thrust sources, namely the two SRBs and the four CS engines. By a somewhat arbitrary choice, the thrust contribution from the CS engines was modeled with one additional level of goal, "Control Core Stage's Contribution to Vehicle Axial Thrust", whereas for the boosters, the thrust contribution goals for each individual booster was put directly under the "Provide Axial Thrust" goal. Figure 5-5 shows the breakdown of the "Provide Axial Thrust" goal into the contributions from the two boosters and each CS engine of axial thrust. The function that defines the combination of each individual CS engine contribution to vehicle axial thrust simply represents a summation of the individual CS engine thrust vector components along the vehicle's centerline axis. The contribution of axial thrust by each SRB is defined by the thrust and maintaining commanded thrust angles. The same concept is used for the CS engines. Details about these branches of the GTST are described in the respective sections Booster Details (5.2.1) and Core Details (5.2.2).

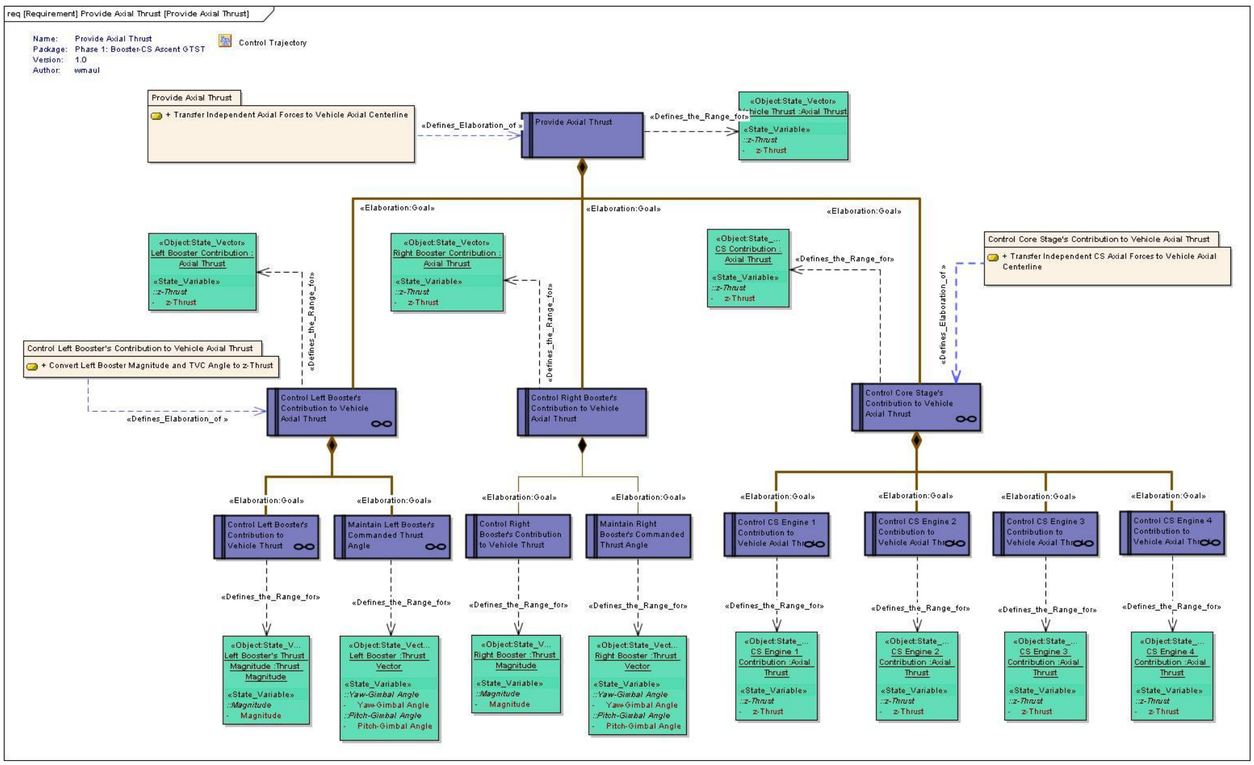

Figure 5-5. "Provide Axial Thrust" Goal Diagram under Booster-CS Ascent

The "Control Vehicle Attitude" goal from Figure 5-2 is decomposed into four (4) goals shown in Figure 5-6. One goal that maintains the structural locations of the major vehicle thrust producing sources, Core Stage engines, Left Booster and Right Booster, relative to the vehicle axis. The other goals control the CS engines contribution and each Booster motor contribution to vehicle attitude. Under the goal to control the CS contribution to vehicle attitude there are corresponding sub-goals for each engine to control engine thrust vector and maintain engine location relative to CS center axis. These thrust source positions define the moment arms that when the thrust vectors are applied will produce the torque to maintain the proper vehicle attitude along the flight path.

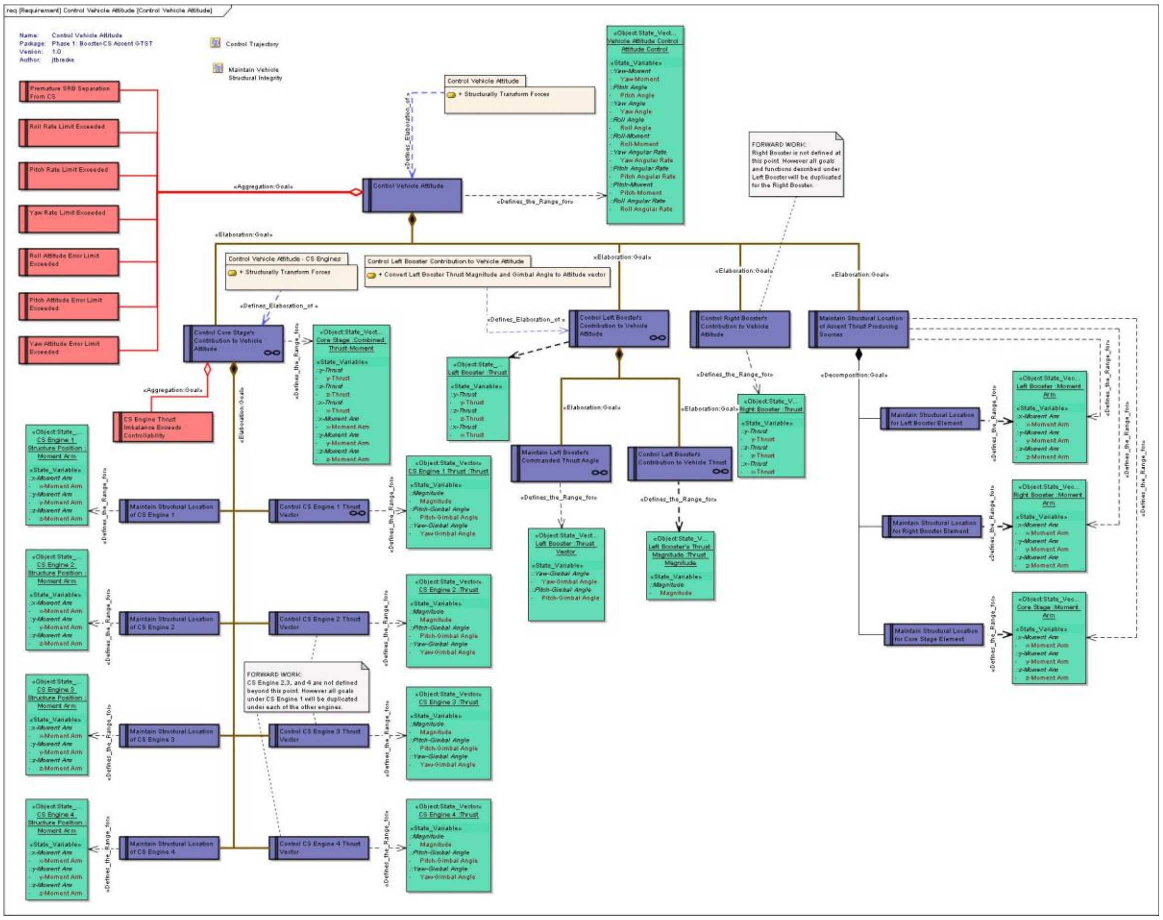

Figure 5-6. "Control Vehicle Attitude" Goal Diagram under Booster-CS Ascent

For the second tier maintenance goal, "Maintain Vehicle Structural Integrity", displayed in Figure 5-2, there are 5 supporting goals, 3 maintenance goals and 2 prevention goals. Two of the maintenance goals, "Minimize Axial Oscillation" and "Control Vehicle Attitude", also support the other second tiered maintenance goal in this diagram. As a simplification as discussed in the introduction, a single goal is displayed with relationships to multiple parent goals. The two prevention goals, "Prevent Premature Separation" and "Prevent Inadvertent Activation of FTS", represent the examples where the system not function either during the phase where it is not intended to operate, former goal, or erroneously, latter goal. As stated in the GTST note, this portion of the GTST is still under development with much of the state vectors and functions still to be determined.

5.2.1 Booster Details

This section describes in detail some of the lower level Booster goals and functions that were not discussed above. Figure 5-7 shows the branch of the Booster-CS ascent GTST that represents the maintenance goal "Control Left Booster's Contribution to vehicle Thrust". A similar branch would represent the right booster. There are two elaboration goals present in the second layer of the branch: "Maintain Desired Burn Profile in Left Booster" and "Maintain left Booster Structure". The former goal is associated with a state vector whose variables determine the burn profile. As long as the other goal, "Maintain left Booster Structure" is maintained, combustion of the SRB fuel is converted into thrust.

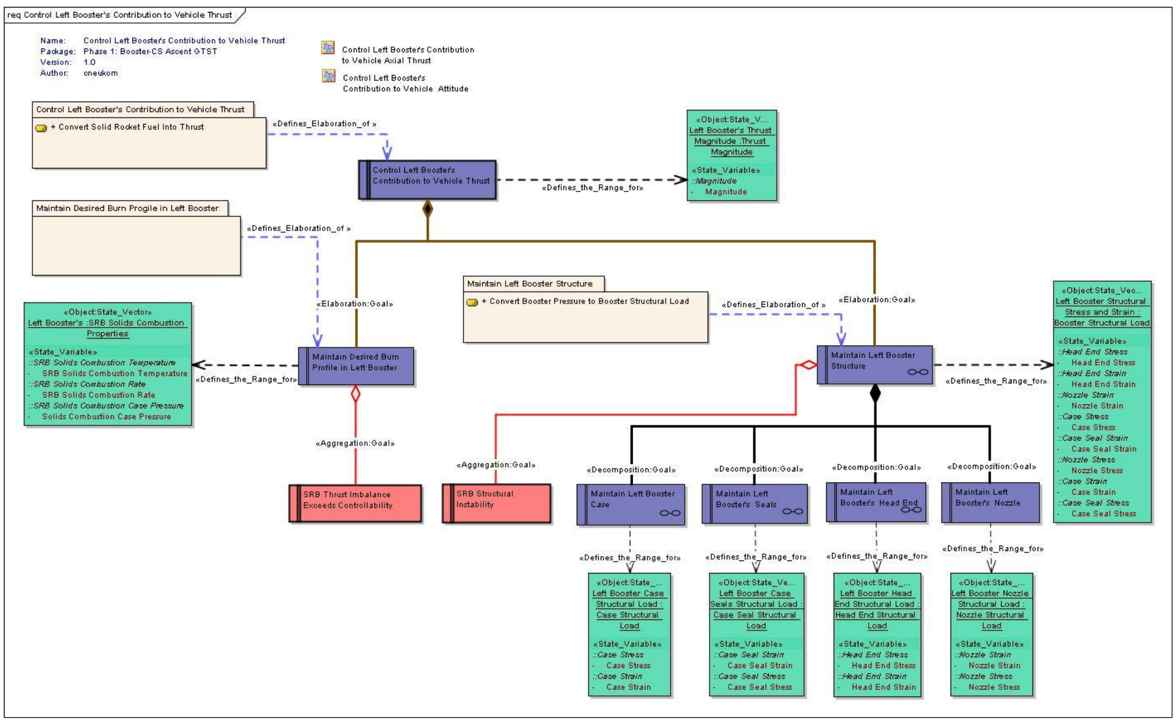

Figure 5-7. "Control Left Booster's Contribution to Vehicle Thrust" Goal

Thrust imbalance may occur if the burn rates of the fuel in the two SRBs are not the same. At the extreme, this could mean that one of the two SRBs failed to ignite, which would most certainly lead to an uncontrollable vehicle. The activity diagram, Figure 5-8, shows the various respective relationships between the Maintenance Goal, the State Vector, the Abort Condition, and the Abort Trigger. As the diagram only shows input from the left booster, a note points out that inputs from both boosters are necessary to determine the thrust imbalance.

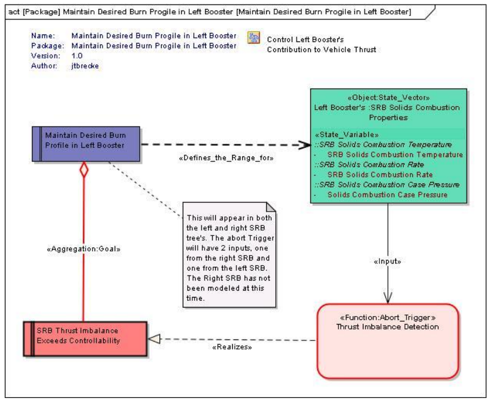

Figure 5-8. "Maintain Desired Burn Profile in Left Booster" Activity Diagram

Figure 5-9 shows the decomposition of the "Maintain Left Booster Structure" goal into the following sub-goals: "Maintain Left Booster Case", "Maintain Left Booster's Seals", "Maintain Left Booster's Head End", and "Maintain Left Boosters Nozzle". Each of these goals defines a state vector of a specific type of structural load, which collectively make up the "Left Booster Structural Load" State Vector in the parent goal. The structural loads that affect the booster parts are a function of the pressure that is exerted by the combustion of the SRB propellant whereas the stress yield limit of each part is affected by the temperature of the ignited SRB.

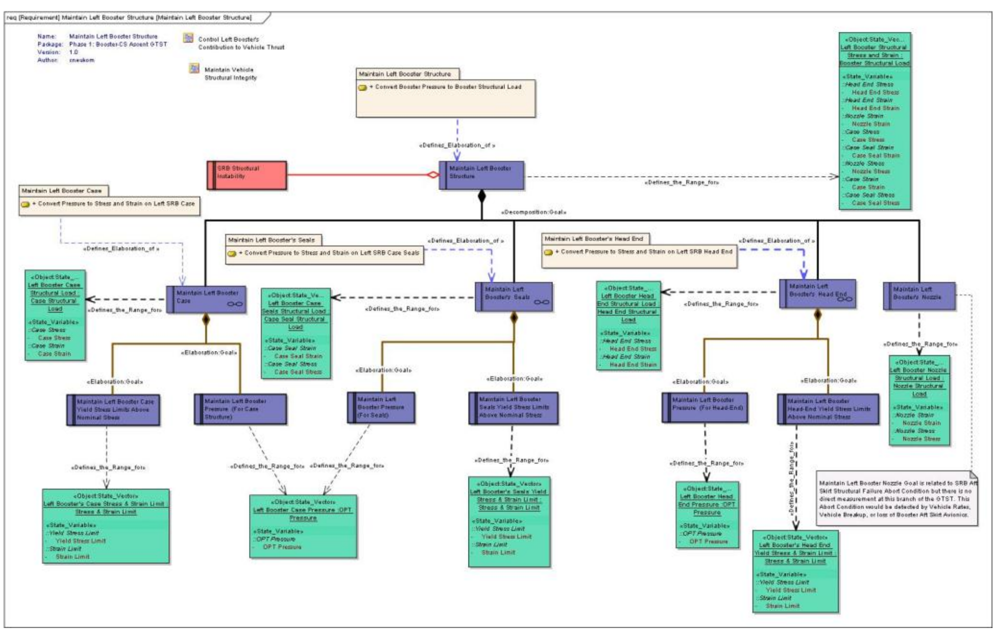

Figure 5-9. Maintain Left Booster Structure

5.2.2 Core Stage Details

This section details the CS specific contributions to the vehicle-level goal of "Control Trajectory", namely "Control Core Stage's Contribution to Vehicle Axial Thrust" and "Control Core Stage's Contribution to Vehicle Attitude". These are illustrated in figures 5-4 and 5-5 respectively. Note the GTST models CSE 1 goals in detail, but it does not model the details for CSEs 2, 3 and 4 because they are duplications of the goal structure defined under CSE 1 goals.

The contribution of axial thrust by each CS Engine is defined by individual engine goals of providing the commanded thrust and maintaining commanded thrust angle. The magnitude of the thrust and the angle of the thrust vector relative to the vehicle centerline axis define the engine's axial thrust contribution. The resulting goal breakdown is displayed in Figure 5-10. The "Provide CS Engine 1 Thrust" and the "Maintain CS Engine 1 Commanded Thrust Angle" goals define the range for the thrust magnitude and angle respectively. They are identical for each CS engine, but are instantiated with a unique engine number to represent the multiplicity.

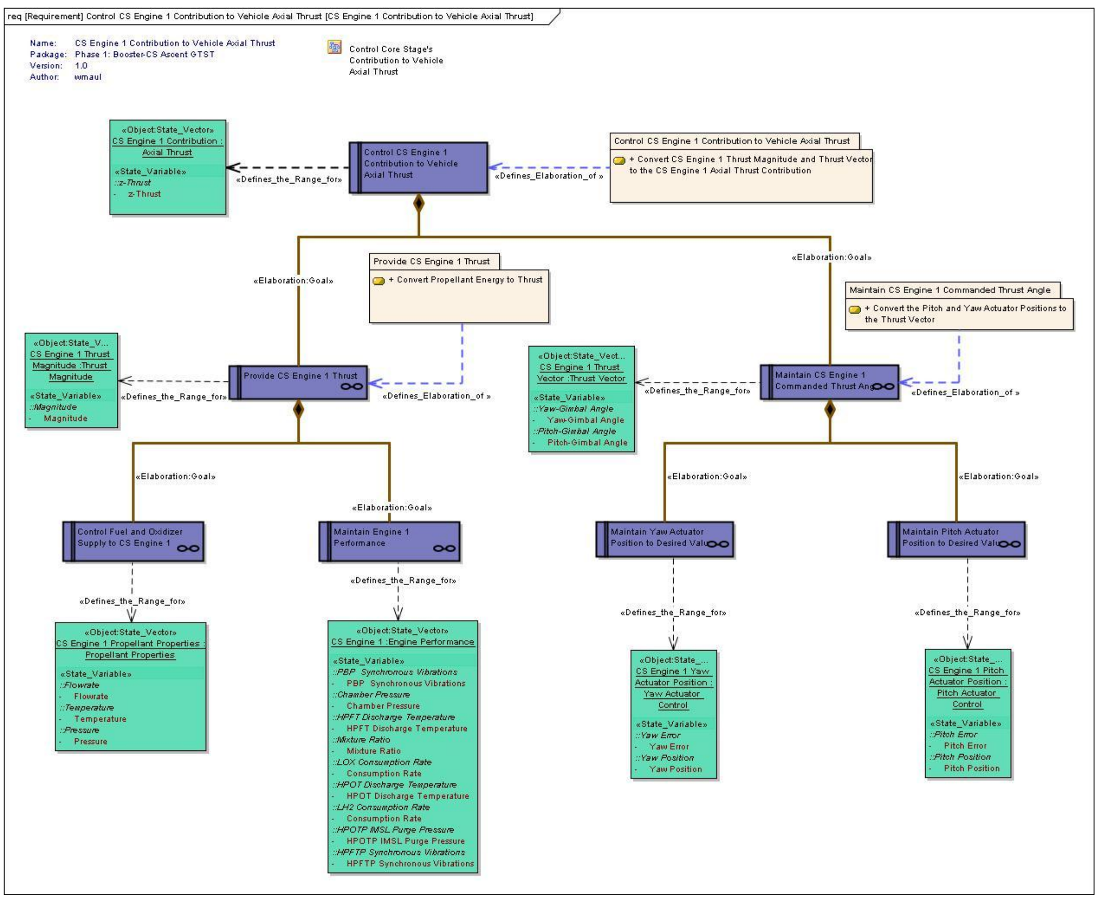

Figure 5-10. "Control CS Engine 1 Contribution to Axial Thrust" Goal

The two supporting goals beneath the "Provide CS Engine 1 Thrust" goal provide propellant supply to the engine and maintain engine performance. The former goal requires propellant tank pressurization and propellant distribution for both the fuel and oxidizer. This portion of the GTST is identical across all four CS engines. This area of the GTST contains the goal to pressurize the fuel propellant tanks. Fuel tank pressurization requires the gasification of LH2 through the CS engine cooling circuit, which in turn requires proper engine operation. In the GTST, the goal cycle is broken and the tree ends at the goal to provide regulated gaseous hydrogen. The GTST cannot contain a loop and therefore once the maintenance goals begin to repeat, the branch should be terminated. These cycles of interdependency are not uncommon in complex, closed systems where many components have multiple goals and operating cycles contain active and passive feedback loops.

The "Maintain CS Engine 1 Commanded Thrust Angle" goal is decomposed into goals maintaining commanded position for two actuators, Pitch and Yaw. The actuators combine to move the CS engine nozzle to the proper angle. Figure 5-11 shows the further breakdown of the goals for each actuator. The supporting goals defined in the GTST include determining the commanded actuator position from the command thrust angle, measuring the actuator position and providing the hydraulic pressure required to drive each actuator. The GTST contains no further detail beyond this point.

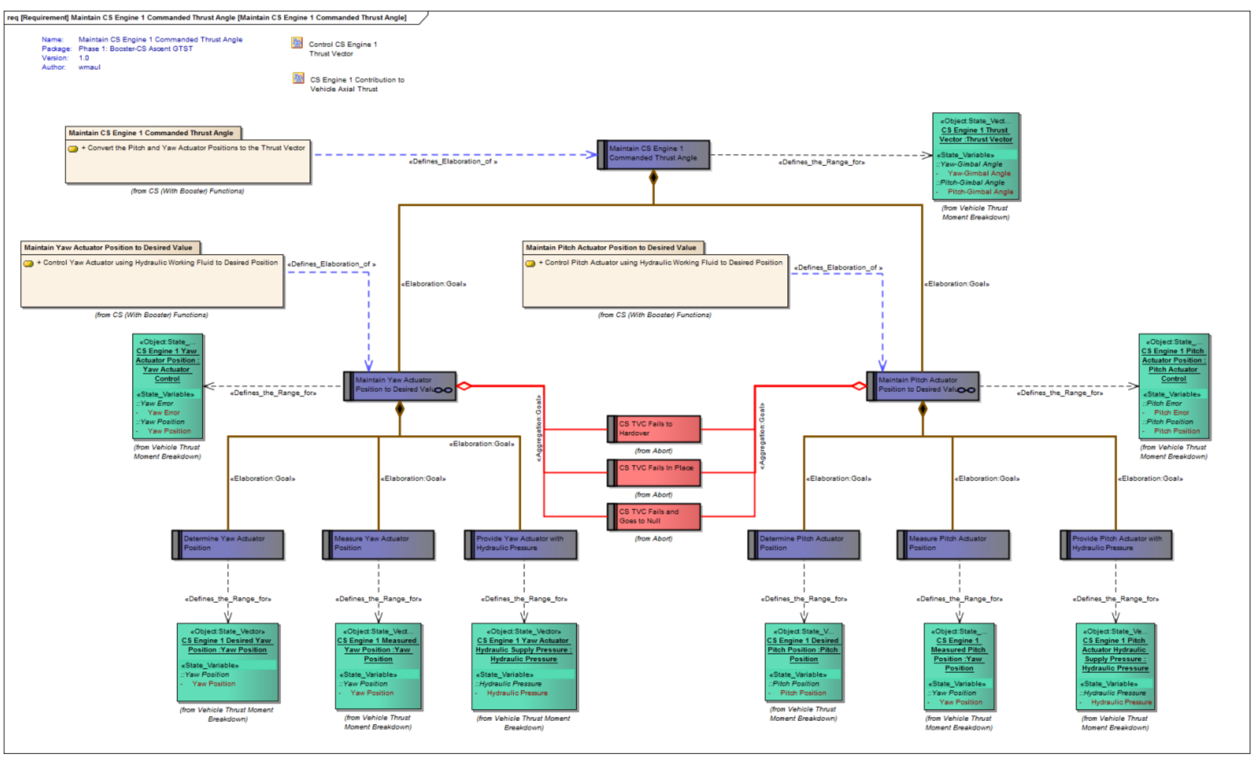

Figure 5-11. "Maintain CS Engine 1 Commanded Thrust Angle" Goal Breakdown View

The “Control Core Stage’s Contribution to Vehicle Attitude” goal breakdown is displayed in Figure 5-12. This goal decomposes into controlling the thrust vector and maintaining the structural position for each CS engine. The “Control CS Engine 1Thrust Vector” goal in this breakdown has the same supporting goals of “Provide CS Engine 1 Thrust” and “Maintain CS Engine 1 Commanded Thrust Angle” as the “Control CS Engine 1 Contribution to Vehicle Axial Thrust” goal in Figure 5-10.

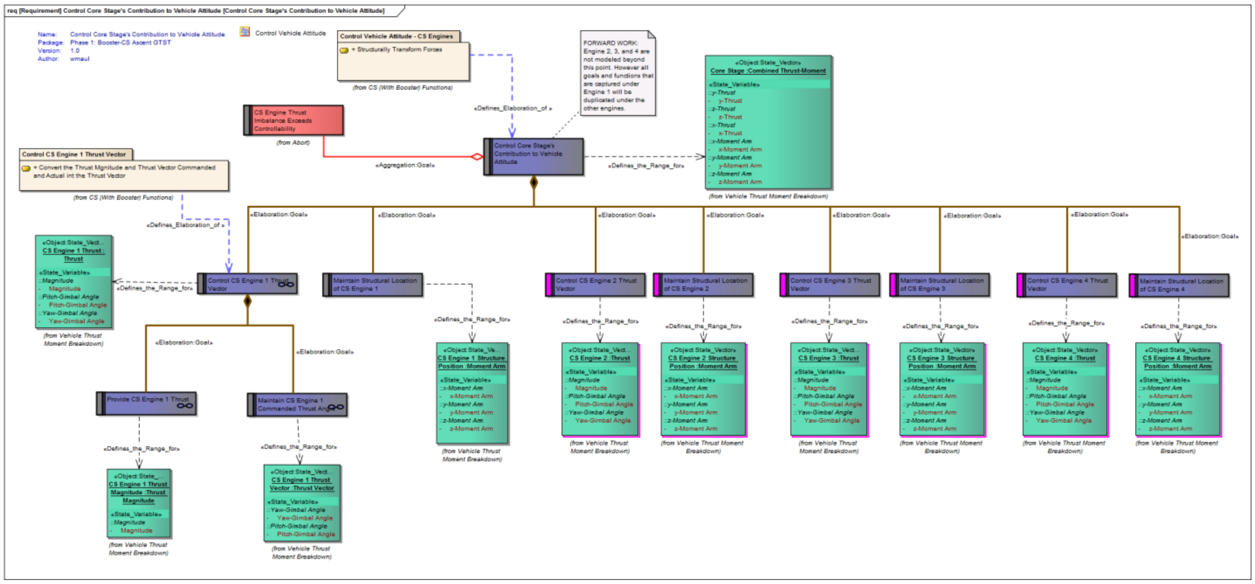

Figure 5-12. “Control Core Stage Contribution to Vehicle Attitude” Goal Breakdown View

Selecting either the “Provide CS Engine 1 Thrust” or “Maintain CS Engine 1 Thrust Angle” goal in Figure 5-10, displays a diagram (shown in Figure 5-13) where the top portion provides two hyperlinks: “CS Engine 1 Contribution to Vehicle Axial Thrust” and “Control CS Engine 1 Thrust Vector”. This indicates that both of these tree segments are part of two distinct tree structures shown in Figure 5-2: “Provide Axial Thrust” goal detailed in the previous sections and “Control Vehicle Attitude” goal.

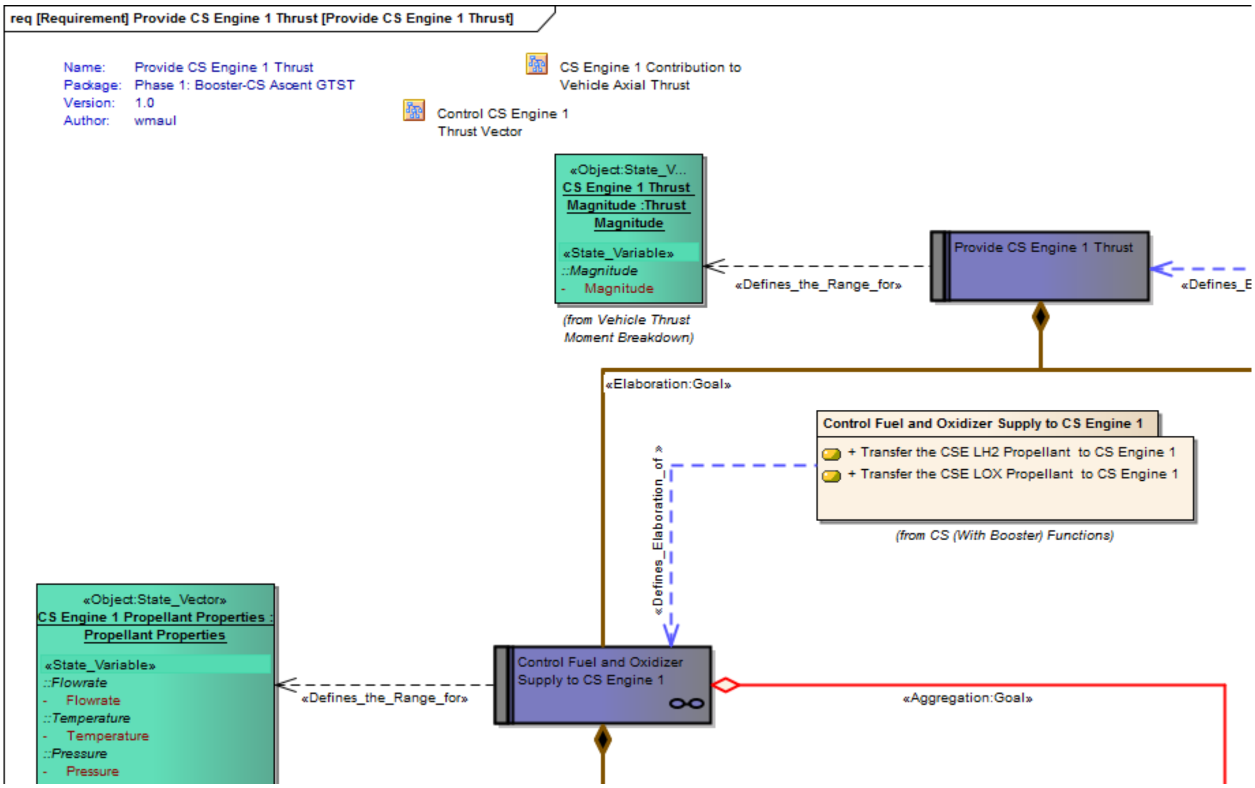

Figure 5-13. Top Portion of the "Provide CS Engine 1 Thrust" Goal Diagram

5.3 Booster-Core Separation

Within the Mission Success Package under the Goal Breakdown view of the SLS Block I GTST, the second phase is titled "Booster-CS Separation". This package holds the details of the decomposition of the "Separate Boosters from Core Stage" goal explained in section 5.1. The "Separate Boosters form Core Stage" goal is decomposed into three parts: pre-separation, physical separation, and post-separation. Pre-separation is essentially a repeat of Phase 1: Booster-CS Ascent described in section 5.2 where the state variables of the lower level goals are such that it would indicate that the vehicle is preparing for separation. Physical separation is a decomposition of the goal to physically separate the boosters from the core stage. This decomposition holds such goals as firing the booster separation pyrotechnics and separation motors. The post-separation decomposition shows the goals that ensure that all necessary actions have taken place to complete a successful separation. In the subsequent sections each of the decompositions are explained. Also, in the diagram below, the "Separate Booster from Core Stage" decomposition is shown.

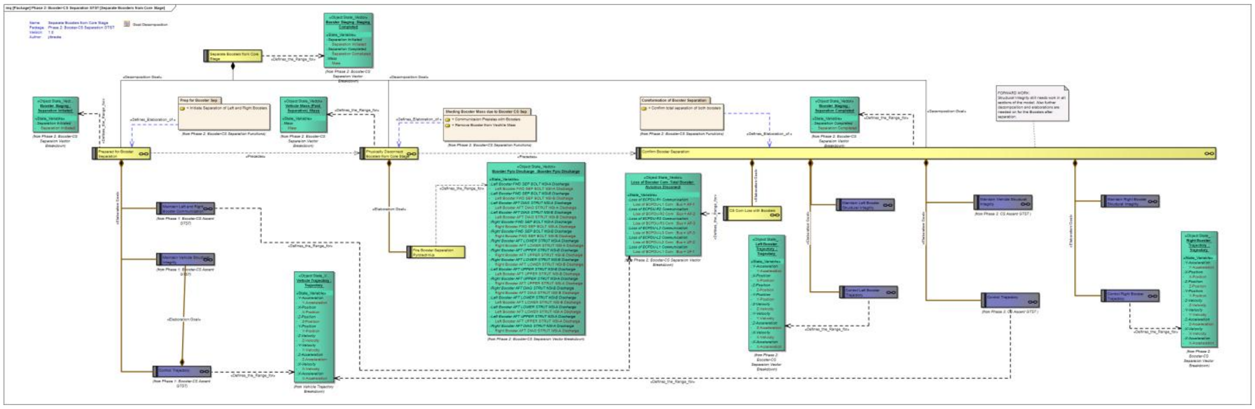

Figure 5-14. Top Level Booster Separation Tree

5.3.1 Pre-Separation

As stated in section 5.3 the first item in the decomposition of the Booster-CS Separation phase of flight is Pre-Separation. This decomposition is essentially a duplication or reuse of previously defined goals from section 5.2 and can be seen in detail within the figure below.

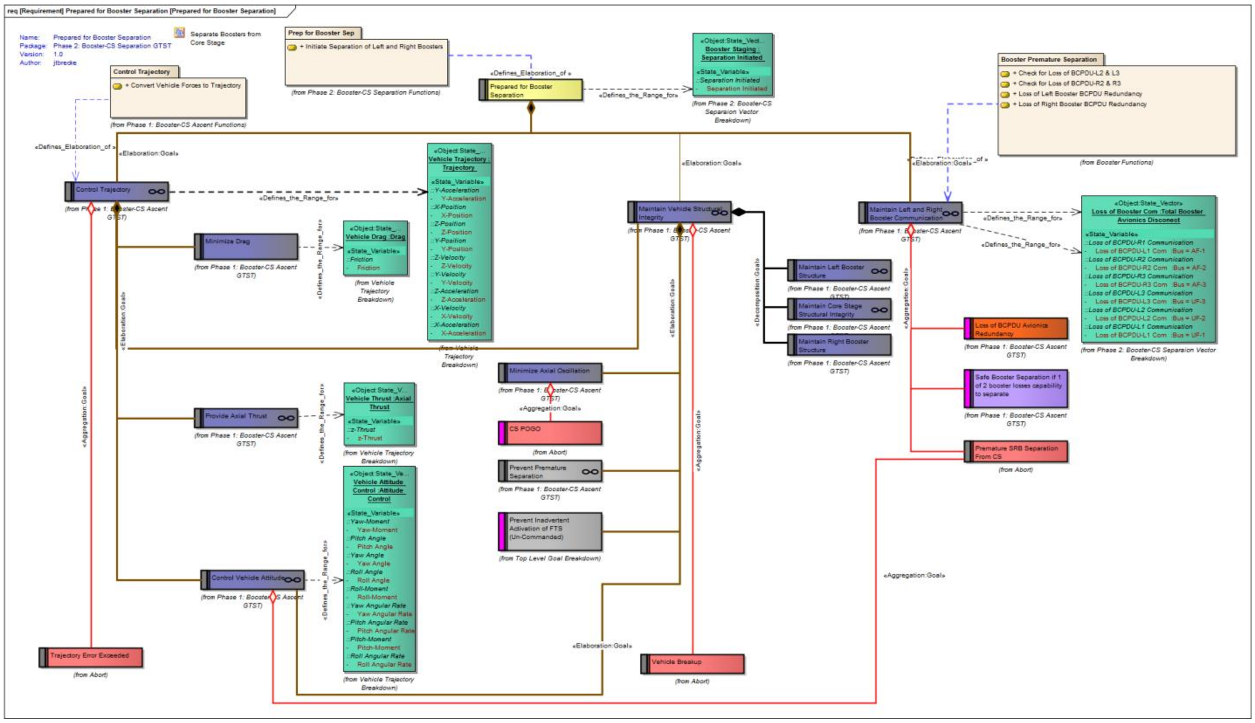

Figure 5-15. Booster Pre-Separation Subphase Decomposition

Due to the nature of the SysML language and the EA tool itself the goals placed on this diagram are deployed from other part of the GTST and as such are not part of the separation tree alone. Rather, they are considered part of the Booster-CS Ascent tree explained in section 5.2. This allows for quick development of the tree by using established artifacts and connections. Additionally it is found to be true that prior to physical separation, that though the phase has changed, the configuration of the vehicle has not. Thus a pre-separation sub-decomposition is needed.

5.3.2 Physical Separation

The physical separation sub-decomposition is made up of only achievement goals. This is based on the fact that physical separation can be thought of as taking place at an instantaneous point in time. That point in time is when the state variables associated with the decomposed goals indicate that separation is occurring. The diagram below shows the breakdown of goals under the "Physically Disconnect Boosters from Core Stage" goal that is decomposed from the "Separate Booster from Core Stage" goal in Figure 5-14.

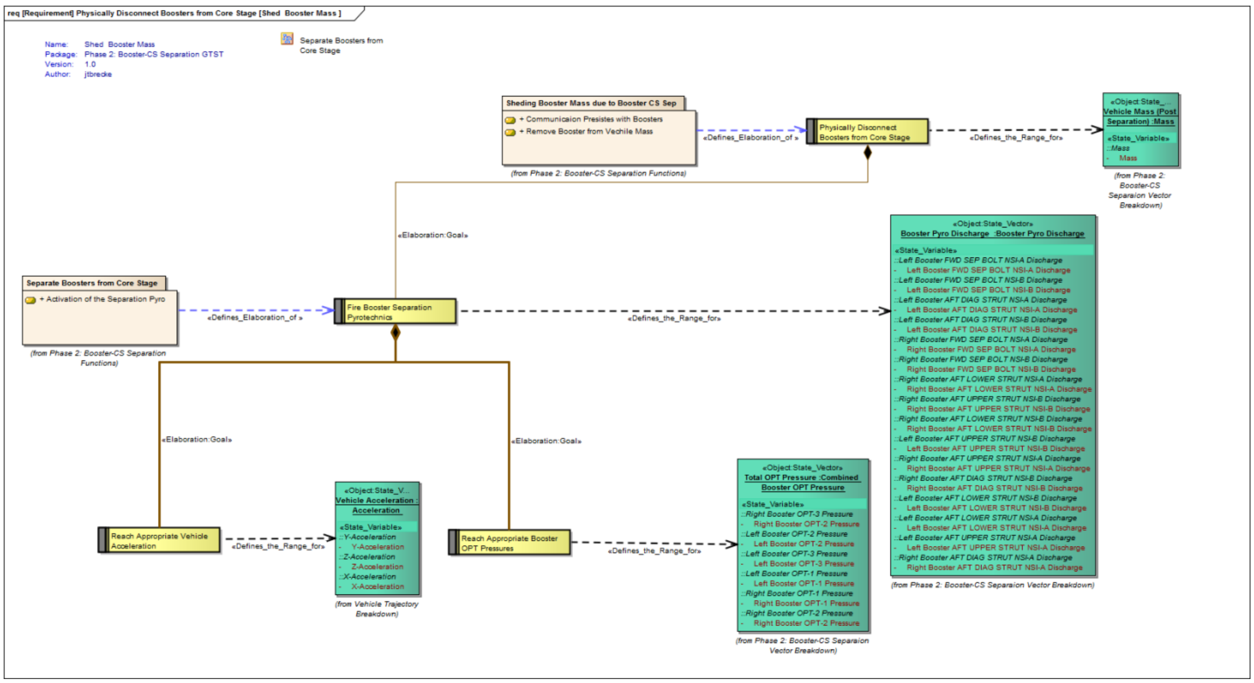

Figure 5-16. Booster Separation Subphase Decomposition

5.3.3 Post-Separation

The final sub-decomposition of the "Separate Booster from Core Stage" goal is the post-separation decomposition. This decomposition is separated into three parts. Two of the parts are associated with each of the left and right boosters, and the third is associated with the continuation of the core stage onto its ascent phase. The booster portions of this decomposition are still in development as of the publication of GTST Version B, but these are meant to indicate the need for the boosters to perform a particular set of goals so as not to interfere with the continuation of the core stage alone in its ascent. The third section of this decomposition is essentially a duplication of the core stage ascent phase of the GTST. Just as in section 5.3.1 in how the Booster-CS Ascent goals were reused, the same is done in the post-separation decomposition only the next phase, CS Ascent, is used. In addition to the duplication of the CS Ascent goals, an additional goal is needed to ensure that separation has occurred properly. The additional goal that is added, “CS Com Loss with Boosters”, allows for an association to the abort condition goal “Failure to Separate”. The nominal and off-nominal goals describe the capability for the vehicle to determine if separation have occurred. The diagram below shows the decomposition of the “Confirm Booster Separation” goal that is decomposed from the “Separate Booster form Core Stage” Goal in Figure 5-14.

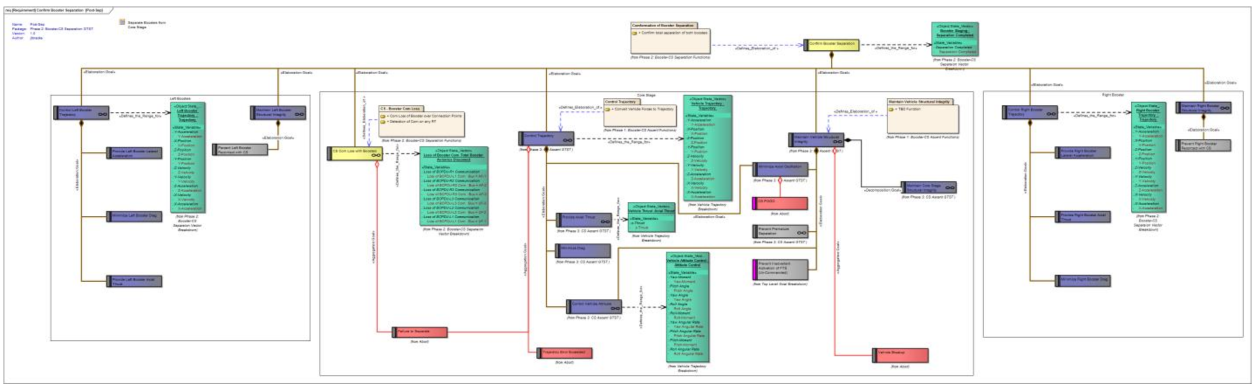

Figure 5-17. Booster Post-Separation Subphase Decomposition

5.4 Core Stage Ascent

This portion of the GTST represents the SLS vehicle operation after Booster-Core Separation is complete and up to separation of the CS and the ICPS. It is located under the achievement goal of “Deliver CS-ICPS to ICPS Separation Point”. This achievement goal is divided into two high-level maintenance goals of “Control Trajectory” and “Maintain Vehicle Structural Integrity”. These maintenance goals define continuous and persistent functions performed throughout this phase of the mission. Note that these goals are same as the Booster-Core Ascent phase goals, except that the supporting goals related to the SRBs are removed. The “Control Trajectory” goal contains functions and goals to propel the vehicle along the planned flight path during this period, while the “Maintain Vehicle Structural Integrity” goal contains those functions and goals to retain the vehicle composition and form. Figure 5-18 displays the first two goal tiers beneath the “Deliver CS-ICPS to ICPS Separation Point” achievement goal.

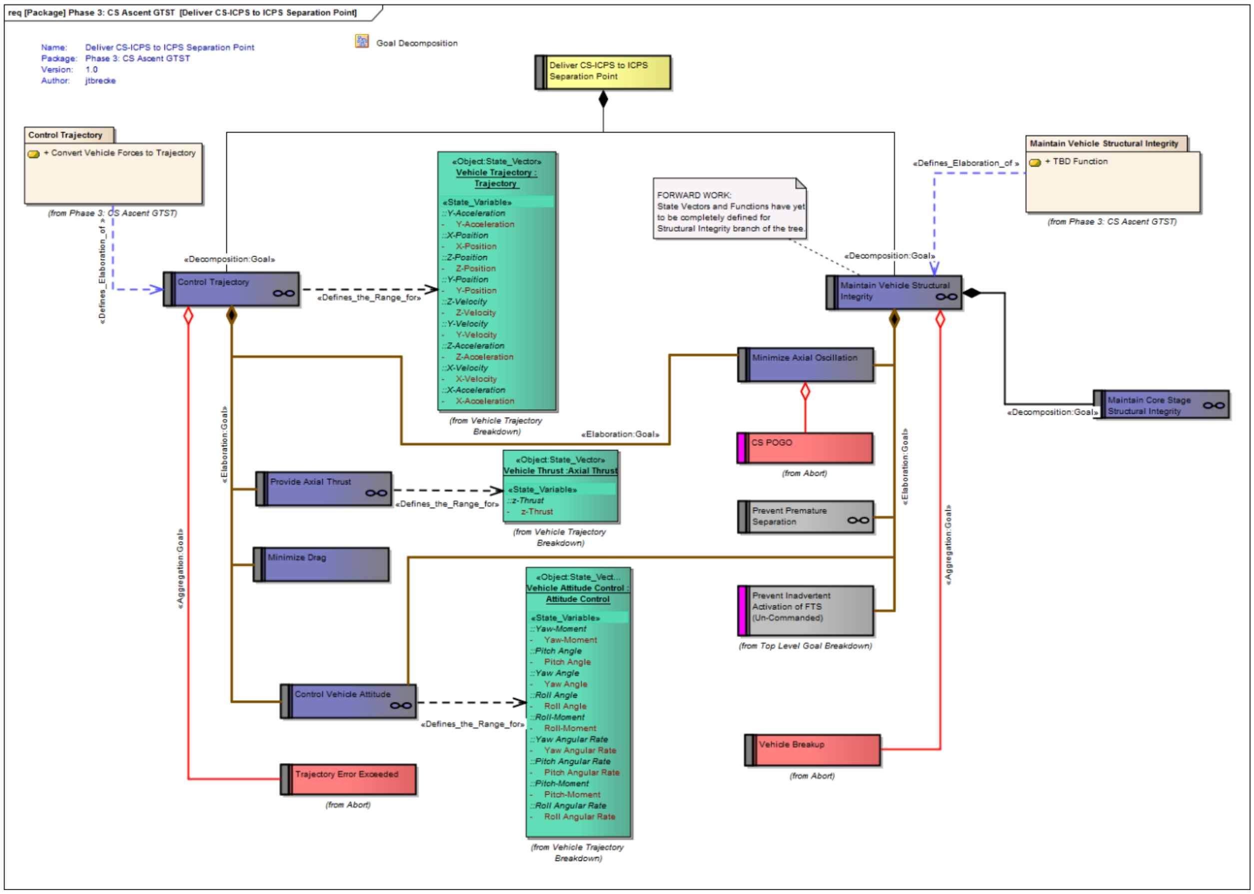

Figure 5-18. Top Tier of CS Ascent during Flight Operations

The details about the lower portions of this tree are provided in Section 5.2.2.

The CS Ascent portion of the GTST contains over 55 maintenance goals and 3 prevention goals. The current GTST structure has a depth of 9 levels below the "Deliver CS-ICPS to ICPS Separation Point" achievement goal. In addition, 29 abort conditions have been placed in the tree indicating the loss of critical function could lead to an abort scenario. Redundant portions of the tree were not completed as this was deemed unnecessary as indicated in the modeling notes. This portion of the GTST lacks details and state vector and function definition under the structural related goals.

5.5 Later Phases

Version B of the GTST model contains the decomposition of the SLS Block I vehicle of the Booster-Core Stage Ascent, Booster-Core Stage Separation, and Core Stage Ascent phases only. Core Stage –ICPS separation and ICPS Ascent are slated for later versions of this model. These phases, although not modeled within Versions B, are briefly described here.

5.5.1 Core Stage-ICPS Separation

The Core Stage-ICPS Separation phase of the GTST is associated with the separation of the ICPS from the Core Stage section of the vehicle. This section of the model, just like the Booster-Core Stage Separation section, will contain three distinct sub-decompositions: pre-separation, separation, and post-separation. The pre-separation sub-decomposition will be a duplication of the previous phase, CS Ascent, to indicate the goals that would need to be successful for the vehicle to get to a point where separation would need to occur, just as described in section 5.3.1. The separation sub-decomposition would be as previously stated in section 5.3.2 and show the achievement goals needed to take place so that a physical separation could take place between the Core Stage and the ICPS. Finally just as in section 5.3.3, the sub-decomposition of post-separation would contain a duplication of the goals in the ICPS Ascent phase of the tree with the addition of necessary goals to show that separation had successfully occurred. Additionally, in the post-separation sub-decomposition there would need to be goals associated with the ICPS engine nozzle extension correctly extending. This would then lead to the ICPS Ascent phase of flight.

5.5.2 ICPS Ascent

The ICPS Ascent phase of the GTST is the decomposition of all of the necessary goals needed to ensure that the ICPS completes its phase of the mission. This section of the GTST will be similar to the CS Ascent phase detailed in section 5.4. A few known differences to the CS Ascent phase are that the ICPS section there would need to contain goals necessary to ensure the successful ignition and re-ignition of the ICPS engine, and the existence of only one engine instead of four.

6.0 GTST-BASED ANALYSIS AND USAGE

6.1 Abort Trigger Safety Net Coverage Analysis

The white paper "**Detection Coverage of the Abort Trigger "Safety Net"**" located at (https://nasa-ice.nasa.gov/Windchill/servlet/AttachmentsDownloadDirectionServlet?oid=OR:wt.doc.WTDocument:101091609&cioids=wt.content.ApplicationData:101091642&role=PRIMARY) describes the initial assessment of a safety net detection coverage analysis. Additional information has been generated on the subject since the paper was published. The dominant control of the safety net will be availability of sensors to enable the safety net. The Abort Condition Detection Hardware study presented on 4 December 2012 located at to the SLS Chief Engineer did not add any additional sensors to the core stage. This results in using existing sensors for determining a Vehicle Breakup (VB) trigger rather than implementing a new specific sensor (example would be adding break wires). This result did not prohibit the use of existing sensors in an innovative way to generate a VB trigger that has not been used in the past. The Initial Goal Tree Success Tree (GTST) assessment has also been completed since the white paper was published.

The GTST demonstrated that coverage of functions assigned to components of the SLS is near-complete. One factor in showing coverage completeness is the completeness of the top levels of the GTST. By modeling system goals and tying these goals to state variables, the GTST inherently shows what goals and functions must be accomplished for the SLS to deliver its crewed payload to orbit, along with potential variables to monitor to determine the success of those goals. The GTST representation thus provides confidence that all relevant top-level goals have been identified, and that their decomposition into lower-level goals does not miss any of the relevant state variables that the SLS system must control.

The Safety Net Abort Triggers include the GN&C attitude error, rate error, and trajectory abort triggers, and vehicle breakup abort triggers. The safety net detection coverage question is whether this set of triggers will detect ALL SLS-originated failure effects. One simple check in the GTST is to simply follow all paths up the tree from bottom to top to ensure that at least one of these triggers exists along the path. If there are any paths in which there are no abort triggers, then we have identified a failure effect propagation path that will directly threaten the crew with no safety net detection mechanism. Simple inspection of the GTST shows that there are no currently modeled paths that do not contain at least one safety net trigger.

The second issue that must be addressed is more subtle. The question is whether there are any sets of paths with safety net abort triggers that are tied to *multiple goals, where each of these different goals link to the same state variable(s)*. The most obvious example of this is goals related to acceleration state variables. In this case, there exist goals to limit or control SLS vehicle accelerations so as to prevent: breakup of the SLS vehicle structure, breakup of the MPCV structure, excessive acceleration on the crew. Each of these different goals can and do have different control limits or thresholds to prevent their respective goals from being violated. It is therefore possible that in the system nominal control design, and in the associated abort trigger threshold design, for the limits on acceleration to be set in such a way so as to ensure that one or two of these goals are achieved, but not the third. Thus a search for these situations using the GTST was performed, to identify potential safety net "escapes."

This is exactly what occurs for the one possible "escape" from the safety net that has been identified: axial oscillation. It is possible for there to be axial oscillations that create hazard to the safety of the crew, without breaking either the SLS or MPCV structures, and hence not being detected by vehicle breakup sensing capabilities. This could occur if the axial oscillation limits for crew safety are lower than for SLS or MPCV structures. Searching the GTST identifies and confirms the existence of potential conflicting threshold limits associated with different goals that are linked to the same state variable. Once identified, they can then be analyzed and assessed for potential safety net escapes, as was done in this case.

As design details become available later in the acquisition process, the safety net detection coverage analysis will continue to use the updated GTST to identify any changes in the detection coverage.

6.2 Abort Condition Specification

This section describes how the GTST influenced the development of the list of abort conditions and triggers. Previous efforts in developing these lists relied primarily on "engineering judgment"; the knowledge that engineers accumulate and apply, based on their experience with past hardware and control failures and on existing sensors and measurements that might protect from such failures. The principles used to develop the GTST on the other hand are based more on the physics of the system, creating a hierarchical decomposition of goals, identifying state variables at each level, and introducing functions that describe the behavior of the system with the state variables being inputs and outputs.

State variables are the values that the system requires to maintain the success of the system and when they cannot be maintained within some range, this will "trigger" an abort condition. We learned that the state variables and the method in which this data is obtained are two separate concepts and also that the data needed to determine an abort condition is often not directly measurable, thus requiring that a different measurement is needed to trigger an abort.

The GTST analysis with its state vectors made it very clear what the abort condition indicators are for each abort condition. For example, for the abort condition, "SRB Case Pressure Burst", the abort condition indicator is stress and strain on the SRB case. There are currently no sensors available in the design that directly measure stress and strain but a state variable that is the main contributor to stress and strain is the case pressure that can be measured from the Operational Pressure Transducer sensors. As discussed in section 5.2.1, the GTST analysis also brought to light how the temperature affects the stress and strain limit of the SRB case.

In the initial "engineering judgment" AC/AT list, the use of the term "trigger" was misused and the data needed to indicate the abort condition was identified inconsistently. The original list contained a mix of specific sensors or the output of data from a piece of hardware (for example, attitude rate from the GNC). Using what we learned in part from the GTST, we moved to a more consistent representation in the ACL, using the term "abort condition indicator" to describe the actual data value(s) needed, indicator being almost synonymous with state variable. Therefore the abort conditions were rewritten to use terms from the GTST such as Ullage Pressure and Tank Thermal Load to describe the Abort Condition Indicators, separating them from the method in which the data was supplied. This also helped with the Abort Condition Detection Hardware Study where the hardware needed to sense this data was evaluated, rather than the Conditions themselves.

6.3 Abort Trigger Gaps and Overlaps

In the previous section, we discussed how developing the GTST helped to better understand what an individual abort condition really was by tying it to a goal node and using the state variables as abort condition indicators. This also allowed us to separate the choice of hardware and sensors from the condition itself.

Moving from the individual abort conditions to the full tree, the GTST allowed us to perform multiple analyses:

- Compare, assess, and improve the initial set of abort conditions based on "engineering judgment" against those identified in the GTST modeling effort and vice versa.
- Understand the relationships between the different abort conditions and their relative locations in the tree
    - Assess whether each abort condition was at the correct level. Each goal and associated abort condition was inspected to determine if an abort condition could be detected closer to the root cause and thus provide a longer (better) abort warning time.
    - Identify secondary abort triggers. Traversing the goal paths revealed which secondary (or higher) triggers could be used as a backup if the primary trigger does not detect a failure effect.
    - Check for the existence of "safety net" abort conditions and triggers. These are conditions and triggers associated with goals that are at or near the root (top) of the GTST which would catch and cause an abort if all other paths failed. In the case of this GTST, all paths followed led past either the "Vehicle Breakup" or "Trajectory Error Exceeded" triggers, which will operate as the last resort of aborting due to a failure if earlier triggers did not catch it. See section 6.1.
    - Understand the hierarchical relationships between the abort triggers and conditions and make assessments as to whether it would be acceptable to have more than one Abort Condition or Trigger along the same goal path, and in doing this, assess the relative value of the various triggers to see if they are all appropriate and necessary for the SLS design to save the crew.

- Search for gaps in trigger coverage by identifying paths where there were no abort conditions or triggers with several possible thresholds based on differing requirements or needs.
- Assess how the GTST structure is related to the definition of the abort scenarios (consisting of unique failure effects, failure responses, and mission phase, time, and configuration). Each path in the tree is related to a “failure scenario” in the Safety and Mission Assurance Probabilistic Risk Assessment.
- Support the Abort Condition Detection Hardware (ACDH) Study and the future abort analysis matrix. In particular, we used insight from the GTST to map abort triggers into each failure scenario-Loss of Mission Environment combination in the ACDH Study Matrix.

The following example shows how the GTST supports these analyses. Figure 6-1 shows a Goal Tree representing one of the flight phases in its entirety. We initially created these kinds of views of the GTST for navigation purposes but we soon realized how useful these views are to visualize the various relationships among the abort conditions and for gaining better understanding about how a failure in one branch of the tree can affect a goal in another branch.

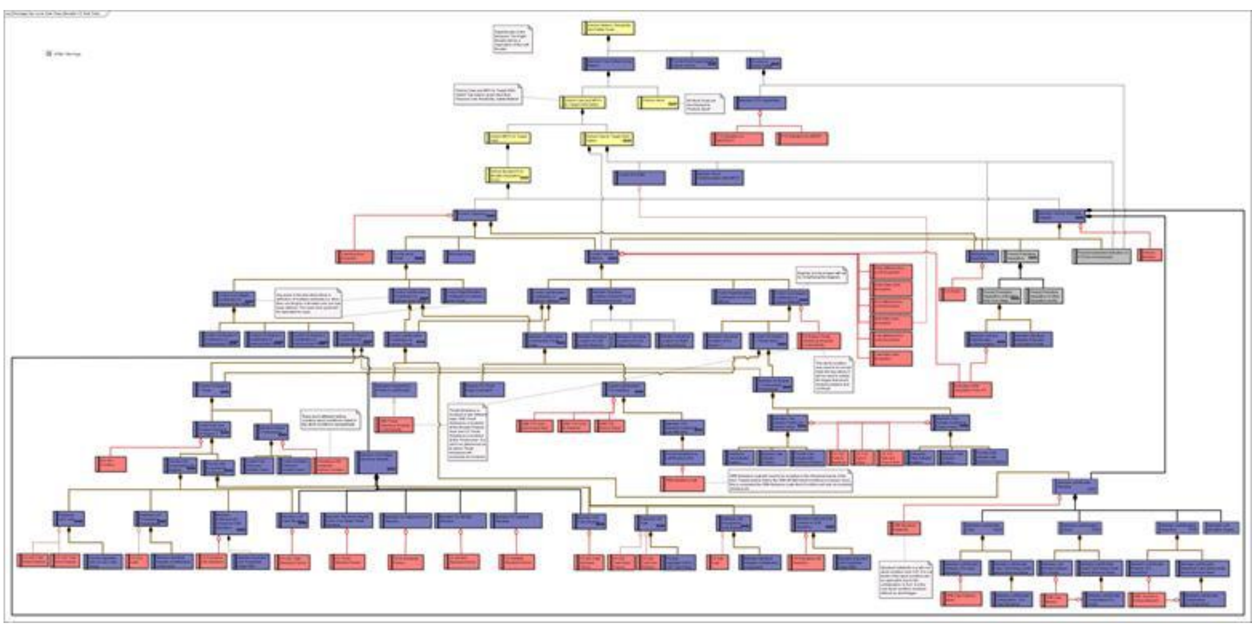

Figure 6-1. Booster-CS Ascent Top Level GTST

Using this tree, it is possible to visually traverse the tree from leaf-level goals to the root goal and note the abort conditions on the path. It becomes apparent that detecting a failure near a root cause may provide additional time for the crew to abort but such triggers near the leaf goals only provide detection for a small set of GTST branches. [Note: There are some unusual cases related to the non-linearity of fluid flows and combustion where this may not always be true, such as rupture of propellant tanks versus engine combustion fluctuations.] The GTST also highlights some goals at branch points where many of the lower level goals feed into, for example, Control Attitude. Most of the failures will eventually lead to loss of attitude control if they progress along their failure effect propagation paths (which inherently compromise system goals as you move along GTST paths).

Although most of the goals below a certain level are either maintenance or prevention goals (in other words functional goals), it is apparent that as one proceeds toward the leaves (the lowest levels of the tree), the goals are more and more tied to specific hardware. Of course, without knowledge of hardware, such specific goals could not even be defined. Construction of the GTST at a certain level of sophistication requires some knowledge of hardware and architecture. However, early in the design it is necessary to keep goals as general as possible to allow for later changes in hardware.

Insight gained from traversing the GTST has helped determining the necessary trigger hardware for the SLS for providing adequate protection of the crew. In the following example, the GTST was traversed from leaf goal to root goal for the Booster-CS ascent GTST and the abort conditions associated with the goals along the paths were noted. The following dash (-) separated lists of goals shows such a path from the leaf-level maintenance goal “Maintain Expected LOX Propellant Usage Rate” to the root-level achievement goal “Deliver Booster-CS to Booster Separation Point”.

Maintain Expected LOX Propellant Usage Rate - Maintain Sufficient LOX Volume for CSE Operation - Provide LH2 Propellant to CS Engines - Control Fuel and Oxidizer Supply to CS Engine 1 - Provide CS Engine 1 Thrust - Control CS Engine 1 Contribution to Vehicle Axial Thrust - Control Core Stage's Contribution to Vehicle Axial Thrust - Provide Axial Thrust - Provide Axial Thrust - Deliver Booster-CS to Booster Separation Point.

There are two additional paths with the same start and end point but they are not listed here. Shown below are three lists of abort conditions that are associated with a number of goals along each of the three paths. *Note: Abort conditions between parentheses are associated with the same goal.*

- CS Premature LO2 Depletion – Run Box Violation – Trajectory Error Exceeded
- CS Premature LO2 Depletion – Run Box Violation – CS Engine Thrust Imbalance Exceeds Controllability – (Pitch Rate Limit Exceeded, Yaw Rate Limit Exceeded, Roll Rate Limit Exceeded, Pitch Attitude Error Limit Exceeded, Yaw Attitude Error Limit Exceeded, Roll Attitude Error Limit Exceeded, Premature SRB Separation From CS) – Trajectory Error Exceeded

- CS Premature LO2 Depletion – Run Box Violation – CS Engine Thrust Imbalance Exceeds Controllability – (Pitch Rate Limit Exceeded, Yaw Rate Limit Exceeded, Roll Rate Limit Exceeded, Pitch Attitude Error Limit Exceeded, Yaw Attitude Error Limit Exceeded, Roll Attitude Error Limit Exceeded, Premature SRB Separation From CS) – Vehicle Breakup

Although this analysis was able to point out some additional possible triggers, such as low hydrazine tank pressure and temperature caused by a hydrazine leak, which could lead to loss of SRB TVC, it was determined that such detection would most likely not be sufficient cause for triggering an abort but possibly could be considered for a caution and warning if the probability of such a failure to occur would warrant it.

6.4 Abort Trigger Mapping to Abort Condition Detection Hardware Study Matrix

One critical use of the GTST was in support of the Abort Condition Detection Hardware (ACDH) Study. The ACDH Study created and utilized an Excel Spreadsheet in which the calculations of the Loss of Crew (LOC) benefits of anticipated and candidate abort triggers were performed. The matrix contained rows that were failure scenario (FS)-Loss of Mission Environment (LOME) combinations. The columns consisted of the anticipated and candidate abort triggers. Expected abort triggers were those expected to be approved and in the SLS design, whereas candidate abort triggers were those being assessed to determine how much additional LOC benefit is obtained by adding these candidates.

The GTST supported this effort, by making it straight-forward to determine which abort triggers were potentially activated in any FS-LOME combination. These abort triggers are associated with abort conditions in the GTST, and every FS-LOME combination could be associated in the GTST with a specific path "up the tree". Along each of these paths existed specific abort conditions and abort triggers, which are the triggers that should be mapped as columns into the rows of the ACDH Study Matrix. This mapping was much easier, because the GTST explicitly identified the anticipated and candidate abort triggers along each path, and hence to be represented in the matrix rows. In addition, the ordering of the abort triggers up the path is related to the identification of abort triggers in the ACDH Study Matrix as primary or secondary triggers.

7.0 FORWARD WORK

The GTST will be updated for Critical Design Review (CDR), with a corresponding update to this document. Possible updates to the model and analyses are listed here.

7.1 Trace to SLS Vehicle Functional Analysis

The SLS Systems Engineering and Integration organization has developed a Vehicle Functional Analysis Model (VFAM), which identifies and represents SLS functional requirements. Given that the GTST team developed a semi-independent set of functions and implied requirements (constraints on state variable outputs of functions), a comparison between the GTST and the VFAM would provide a cross-check of both models, as well as of SLS requirements. We plan to perform this comparison to, and trace from, the VFAM to the GTST shortly after PDR.

7.2 Trace to SLS M&FM Design Model

The M&FM (design) Model is the SysML representation of M&FM-built algorithms. For abort algorithms, the GTST represents the requirements that call out the need for the algorithm. As such, there should be a trace from the GTST to the M&FM (design) Model. We plan to perform this trace after completing the VFAM to GTST comparison and trace.

7.3 Redundancy Management and Avionics Modeling

For PDR, the GTST does not contain explicit information regarding system redundancy, or the avionics implementations that contain significant portions of this redundancy. Determining how to model avionics and redundancy is an important task that we expect to perform after the trace work described in 7.2 and 7.3. However, it is not yet determined whether we will perform full modeling of the RM and avionics. That determination will be made in the months soon after PDR. The GTST can provide similar insights regarding system redundancy and avionics functionality as it did with aborts, but whether it is worth the time and effort to do so must be determined.

7.4 Failure Scenario Cross-Check

As described earlier in this report, the paths up the GTST from bottom to top are related to Failure Scenario-Loss of Mission Environment combinations that are used to calculate abort effectiveness. A comprehensive cross-check of these FS-LOME combinations versus the GTST-implied scenarios would be a valuable comparison to determine if there are any missing scenarios. This cross-check should be performed within a year after PDR.

7.5 Probability Representation

Just as it is possible to cross-check FS-LOME combinations with the GTST, it is possible to associate the probabilities of FS-LOME combinations with the paths up the GTST. It may be worthwhile to at least determine how this could be done within the GTST framework, with a few sample cases. However, given that the probability calculations for LOC and LOM are the responsibility of S&MA, a comprehensive development of reliabilities within the GTST is not likely to be a worthwhile effort. A few cases may be appropriate to investigate and analyze based on specific M&FM analysis needs.

7.6 ICPS Scenarios

When ICPS information becomes available, the GTST will be extended to include ICPS-CS Separation, and ICPS operation phases. These new portions of the GTST will be modeled to address abort-related issues, as with the PDR Tree.

7.7 Pre-Launch Scenarios

It has not yet been determined if the GTST will be extended to address pre-launch scenarios. This determination will be made within one year of PDR to ensure implementation before CDR, if necessary.

APPENDIX A
ACRONYMS AND ABBREVIATIONS
AND GLOSSARY OF TERMS

A1.0 ACRONYMS AND ABBREVIATIONS

AC	Abort Condition
ACDH	Abort Condition Detection Hardware
ACL	Abort Conditions List
ACR	Abort Conditions Report
AT	Abort Trigger
CDR	Critical Design Review
CMP	Configuration Management Plan
CS	Core Stage
CSE	Core Stage Engine
DAPA	Design Analysis and Performance Assessment
EA	Enterprise Architect
FDDR	Failure Detection, Diagnostics, and Response
FM	Fault Management
FS	Failure Scenario
FTS	Flight Termination System
GN&C	Guidance, Navigation, and Control
GNC	Guidance, Navigation, and Control
GTST	Goal Tree/Success Tree
HTML	Hypertext markup Language
ICPS	Interim Cryogenic Propulsion Stage
JPL	Jet Propulsion Laboratory
LH2	Liquid Hydrogen
LO2	Liquid Oxygen
LOC	Loss of Crew
LOM	Loss of Mission
LOME	Loss of Mission Environment
LOX	Liquid Oxygen
M&FM	Mission and Fault Management
MBSE	Model-Based Systems Engineering
MCR	Monitored Conditions Report
MDS	Mission Data System

MPCV	Multi-Purpose Crew Vehicle
MSFC	Marshall Space Flight Center
OPR	Office of Primary Responsibility
PDR	Preliminary Design Review
PRA	Probabilistic Risk Assessment
S&MA	Safety and Mission Assurance
SLS	Space Launch System
SRB	Solid Rocket Booster
SysML	Systems Modeling Language
TBD	To Be Determined
TBR	To Be Reviewed
TVC	Thrust Vector Control
UML	Unified Modeling Language
VB	Vehicle Breakup
VFAM	Vehicle Functional Analysis Model
VFAR	Vehicle Functional Analysis Report
VM	Vehicle Management

A2.0 GLOSSARY OF TERMS

Term	Description
Decomposition	Breaking down a goal based on the variables in its state vector to lower level goals to further describe the system.
Deploy	Placing an already-created Enterprise Architect artifact from the project browser onto a diagram for use in that diagram.
Elaboration	A form of decomposition where new state variables are needed to add lower level goals.
Stereotype	A SysML capability to expand an existing model element by additional properties and semantics.

APPENDIX B
HISTORICAL ORIGIN AND EXTERNAL SOURCES

B1.0 HISTORICAL ORIGIN OF THE GTST

The GTST for SLS originated from questions that arose on the Ares I project. These questions relate directly to the list of purposes shown above in section 3.1. All of the items on this list are topics that the Failure Detection, Diagnostics, and Response (FDDR) team on Ares I had difficulties addressing. There is no need to replicate that list again, but a common theme among the first three items (abort trigger detection coverage, physical and logical relationships of abort conditions, and potential missing abort conditions) is that they cannot be resolved using “bottom up” approaches.

On Ares I, candidate abort conditions were generated using engineering judgment and documented in the form of the Abort Conditions List (ACL) of the Abort Conditions Report (ACR). This list was a matrix containing the conditions and a variety of condition attributes. As the list was developed, a number of issues arose, which the team had great difficulty in addressing. One particularly troublesome issue was trying to determine what was meant by a “monitored condition”. Some conditions were potentially monitored “at” the location of the abort condition, such as a TVC Hardover Abort Condition, monitored by direct measurement of the gimbal angles. This condition was ultimately classified to be “non-monitored”. This was because the gimbal angle position sensors were of insufficient criticality, meaning it would have been costly to upgrade them and add more sensors, and because failure effects of a TVC gimbal hardover would be visible and adequately monitored and detected by the Guidance, Navigation, and Control Abort Triggers: attitude error and attitude rate error in this case. These attitude and rate triggers were “causally downstream” from the gimbal position. A good case could be made that this meant the condition was actually monitored, though not directly.

This case was only one of many examples showing the existence of a physical, causal relationship between abort conditions, but in list form it was difficult to determine these relationships in a consistent and complete way. If there was more than one way (more than one abort trigger) to monitor an abort condition, how does one identify the several possible abort triggers that could monitor the condition, and determine which of these triggers should be implemented? If more than one trigger is implemented, are we creating unneeded system complexity, adding to cost and perhaps decreasing reliability? Does monitoring mean “directly monitored” as opposed to “indirectly monitored”, and if so, can we define more precisely what “directly monitored” means?

Determining the completeness of the ACL was a second issue. How did one determine whether the ACL was complete? Were there other abort conditions that the team was missing?

A third issue was ascertaining whether the abort conditions in the ACL, and hence their associated potential abort triggers “covered” (could detect) all possible ways that Ares-generated failures could threaten the crew? It was occasionally stated as a fact (when in fact this was a hypothesis) that the GN&C and vehicle breakup triggers would detect all threats to the crew, and this was sometimes used as an argument that little more was needed in terms of abort triggers. Was it really true that these two sets of triggers did in fact detect all threats to the crew? How did one prove this? And even if it was true, would these detections be sufficient to protect the crew? There was, during Ares I, no easy way to answer these questions.

To answer these questions, a new analytical method and approach was needed. This is the genesis of the GTST. It was hypothesized that a top-down approach in success space (the logical complement of failure space) may be able to address these questions. Unlike failure space, in success space, it does not matter how a system goal is not achieved. Focusing on success space, we can ignore the specific failure modes that can cause failure of the goal. It matters only that the goal is not being achieved, because if the goal is not achieved, then higher-level goals will also be compromised. Operationally with abort triggers, we need only determine when the goals of keeping the crew safe or achieving orbit are not or will not be achieved.

Even with a top-down GTST, there are still issues of completeness. How do we ensure that our tree of system goals is complete? How do we link our tree of goals to the system design? It is essential to develop a representation in which we can assess completeness, and can link the representation to the system design. Otherwise we would be unable to answer these critical questions, which are both essential to the design question we want to address: do we have both a complete and optimal (or realistically, near-optimal) suite of abort triggers to protect the crew, and do they do so in a cost-effective way that does not create more losses of mission due to false positives than is absolutely necessary? This is where the insight to use a state variable-based model was essential.

Each goal was linked to a state variable or variables, which if properly controlled, defined the goal’s success. The goal is also associated with functions that transform or convert the incoming state variables into the outgoing state variables. The functional representations are abstract state variable models tied to the system design and operations concept. State variable-based models provide a functional decomposition of the system; combining the system goal tree approach with a state-variable/functional model ensures completeness of the model and identifies failure detection functions to monitor the state variables.

Defining all goals with respect to state variables has several crucial benefits. It inherently links the GTST to the physics and behavioral logic of the system. This enables an approach to check completeness of each level of the goal tree, by identifying lower-level state variables required to support higher-level state variables. It ensures that the construction of the tree is causally correct, in both a physical and logical sense. Finally, state variables are the major link between a functional/intentional model like the GTST and the system design. The state variables exist both in the GTST and in the design, because the design is simply the way in which the functions and goals described in the GTST are implemented. The state variables enable a direct and obvious mapping between the functions and goals on one hand, and the design on the other. Because design solutions may impose new functional relationships, there is an iterative development relationship that exists between the GTST and the design. The use of state variables in the GTST enables a close and traceable linkage between the design and the GTST.

A GTST constructed using state variables as described above has the characteristic that all paths up the tree are causal, physical/logical paths. They are "causal" in the sense that the inability to achieve system goals moving from bottom to top of the tree will mirror the effect of failures through the real design. This feature implies that each path up the tree is, in essence, defining a failure scenario. This in turn has significant short term and long term implications for how we use the tree to map into calculations of abort effectiveness, which are performed on a failure scenario basis, as described in Sections 6.3 and 6.4.

While the GTST has its name because it is both a goal tree, and is a representation in success space, it probably would have been better described as a Goal-Function Tree. This is for two reasons. The first and most important is that "GFT" is a more descriptive phrase for the major features of this type of model. The second reason is to distinguish it from prior methods, which will be described in the next section. Thus while in retrospect, GFT is a better term to describe this approach, we will continue to use GTST as this is the phrase that has been used to date on SLS, with tasks and documents with that name. In the future, we believe it will be better to move to the terminology of GFT.

Finally, although this has not been exploited for the current GTST for SLS, the GTST method using state variables has important long-term implications for systems engineering. It may provide a basis for a far more rigorous and systematic methodology for the specification of system requirements than exists currently. Because the GTST is a model, it can be considered a subset of the several Model-Based Systems Engineering (MBSE) approaches that are in various phases of development across government, industry, and academia.

B2.0 GTST REFERENCES

The SLS GTST is not the first time that goal trees have been created or used, though it does appear that SLS's use of state variables in goal trees is novel.

A recent description of a goal methodology is van Lamsweerde (2009). As implied by the title of this book, *Requirements Engineering: From System Goals to UML Models to Software Specifications*, van Lamsweerde's approach has much in common with the SLS methodology. Van Lamsweerde believes that goal tree specification using "goal diagrams", which just happen to be tree structures, is essential, as well as the criticality of using formal models to represent goals, in his case using Unified Modeling Language (UML). In common with van Lamsweerde, the SLS approach also uses tree structures for goal specification, and uses SysML, which is a systems engineering extension of UML. Chapter 8 of van Lamsweerde's text provides much detail on the structure and use of goal diagrams, which are primarily AND gate structures, with OR-gate modifications for redundancy, and other features for potential goal conflicts, so-called "soft goals", etc. Much of this detail is useful, but the major "take-away" from van

Lamsweerde's work is that goal tree structures are essential to understand and model requirements of complex systems, and that formal modeling using trees can then be linked to many other UML or SysML modeling structures.

Van Lamsweerde correctly notes that tree structures for representation of goals is common in artificial intelligence, going back at least to (Nilsson 1971). Its introduction into requirements engineering, according to van Lamsweerde, came with the work of (Dardenne et al., 1991) and (Mylopoulos et al., 1992).

Kim and Modarres (1986) introduced a Goal Tree-Success Tree representation for use in operator advisory systems. This approach developed in the mid-1980s for nuclear power systems modeling, and was tied to expert system development, which is the same conceptual origin as used in artificial intelligence. For Modarres, the goal tree portion represented the top-level goals, and the success tree accounted for lower level goals that needed to be accomplished for the higher level goals to be achieved. It too used an AND-gate structure as primary, with OR-gates for alternative methods to achieve a goal. In the advisory system application, achievement of success of lower level goals (the lower level "success trees") was required to achieve high-level goals. Modarres (1999) continued to develop the GTST approach, creating "master logic diagrams" of top level goals as columns and "support functions" as rows to begin assessing how failures affected goals. While attacking the right problem (failures and relationship to goals and functions) this approach developed into an overly complex and largely unusable methodology and model. These complexities may have been avoided if a state variable methodology had been considered. However, it does point to the difficult issue of mapping from goals and functions to design, which is a many-to-many complex problem. Modarres's approach seems to confuse function space with design space, which causes unnecessary problems.

Another GTST-related development is the state analysis and Mission Data System (MDS) under development at Jet Propulsion Laboratory (JPL). Ingham et al. (2005) is an early description of this approach. The JPL work uses a state-based approach to develop an autonomous, on-board deep space computer and computing architecture. This architecture employs a Goal-Tree-like structure to represent goals, for the same reasons that artificial intelligence applications use tree structured search spaces. Fragments of the tree can be shifted around based on changes to the vehicle configuration, representing changes to the mechanisms needed to operate system functionality. A separate set of models represents system constraints. As a state-based approach, these representations utilize state variables and hence have many parallels to the GTST representation. The key difference between the GTST and the MDS state analysis is that GTST is aiming towards design development, whereas MDS aims toward an operational, deep-space architecture. The representation constructs are similar, but the applications and analyses differ.

The uniqueness and effectiveness of the SLS approach appears to be due largely to the adoption of the state variable methodology, of which a variant is in development by van Lamsweerde's group. Both van Lamsweerde and the SLS GTST approach strictly separate intention and function from design, whereas Modarres's methods confused them. There are also close parallels between the JPL MDS and state analysis approach, and the SLS GTST approach. In sum, the SLS GTST is a unique and independent development from those of Modarres, Van Lamsweerde, and JPL, but it shares some of their beneficial characteristics and is used in a quite different manner for Fault Management analysis.

APPENDIX C
OPEN WORK

All resolved TBDs, TBRs, and forward work items should be listed on the Change Request (CR) the next time the document is updated and submitted for formal review, and that will serve as the formal change record through the configuration management system.

C1.0 TO BE DETERMINED

Table C1-1 lists the specific To Be Determined (TBD) items in the document that are not yet known. The TBD is inserted as a placeholder wherever the required data is needed and is formatted in bold type within carets. The TBD item is sequentially numbered as applicable (i.e., **<TBD-001>** is the first undetermined item assigned in the document). As each TBD is resolved, the updated text is inserted in each place that the TBD appears in the document and the item is removed from this table. As new TBD items are assigned, they will be added to this list in accordance with the above described numbering scheme. Original TBDs will not be renumbered.

Table C1-1. To Be Determined Items

TBD	Section	Description
TBD-001		

C2.0 TO BE RESOLVED

Table C2-1 lists the specific To Be Resolved (TBR) issues in the document that are not yet known. The TBR is inserted as a placeholder wherever the required data is needed and is formatted in bold type within carets. The TBR issue is sequentially numbered as applicable (i.e., **<TBR-001>** is the first unresolved issue assigned in the document). As each TBR is resolved, the updated text is inserted in each place that the TBR appears in the document and the issue is removed from this table. As new TBR issues are assigned, they will be added to this list in accordance with the above described numbering scheme. Original TBRs will not be renumbered.

Table C2-1. To Be Resolved Issues

TBR	Section	Description
TBR-001		