Implementation of a Goal-Based Systems Engineering Process Using the Systems Modeling Language (SysML)

Jonathan T. Breckenridge[i]
Jacobs – ESSSA Group / Ducommun Incorporated, Miltec Systems, MSFC, Huntsville, AL, 35763, USA

and

Dr. Stephen B. Johnson[ii]
Dependable System Technologies LLC / Jacobs – ESSSA Group, MSFC, Larkspur, CO, 80118, USA

Abstract: This paper describes the core framework used to implement a Goal-Function Tree (GFT) based systems engineering process using the Systems Modeling Language. It defines a set of principles built upon by the theoretical approach described in the InfoTech 2013 ISHM paper titled “Goal-Function Tree Modeling for Systems Engineering and Fault Management” presented by Dr. Stephen B. Johnson. Using the SysML language, the principles in this paper describe the expansion of the SysML language as a baseline in order to: hierarchically describe a system, describe that system functionally within success space, and allocate detection mechanisms to success functions for system protection.

I. Introduction

As described in the Infotech@Aerospace 2013 paper “Goal-Function Tree Modeling for Systems Engineering and Fault Management” by Stephen B. Johnson, a Goal-Function Tree (GFT) is defined as *a top-down functional decomposition of system goals and functions, arranged by major system phase/configuration; to define what functions the system must perform for the system to successfully perform its mission/objectives*. The GFT utilizes a hierarchical decomposition of system intentions, or “Goals”, within success space, the identification of the system’s State Variables at each level of the tree, and a structure inherently related to the system’s physics and real behaviors. The system’s Goals are hierarchically decomposed in success space to show the relationship between the differing levels of the tree. This relates the fact that for one level of Goals to be successful, all hierarchically decomposed Goals of a lower level must first be successful. For a Goal succeed, that Goal’s State Variables must be held within a specified range. State Variables are defined as inputs and outputs to a system’s Functions as in the mathematical expression y=f(x) where x is the input State Variable to Function f, and y is the State Variable output from f. This strict correlation between Goals, State Variables, and Functions enforces a strong connection of the functional decomposition to the system’s physical laws and causation. The State Variables are the connection between Function and design as they exist in both function space and design space. Additionally due to the fact that every Goal inherently describes the nominal range for its allocated State Variables, an off-nominal Goal can be assigned to require an action the system must take if the nominal range is violated (i.e. For every nominal Goal, there is a potential off-nominal Goal). The GFT is an important new representation within the emerging paradigm of Model-Based Systems Engineering (MBSE), as it enables representation of the designer’s intention for a system within a rigorous state-based, goal-based methodology. The physical and logical rigor of the GFT enables it to be used for a variety of purposes, including robust system planning, analysis of failure coverage, and generation of failure scenarios.

The GFT approach was developed on the National Aeronautics and Space Administration’s (NASA) Space Launch System Program (SLSP). Some of the modeling constructs developed and described below are specifically associated with the SLSP, but the concepts are general in nature. This paper will describe the concepts involved in developing a GFT using SysML as a baseline and show all of the additions that are needed to be made to standard SysML in order to create a GFT.

[i] Assistant Fault Management Analysis Lead, Mission and Fault Management, Space Launch System Program, Marshall Space Flight Center (MSFC), AIAA Member.
[ii] Fault Management Analysis Lead, Mission and Fault Management, Space Launch System Program, Marshall Space Flight Center (MSFC), AIAA Member.

II. GFT Development Process

Using all of the diagrams, constructs, and construct connections described in Section III, this section describes a process for creating the GFT in its entirety. This process was developed using Enterprise Architect (EA), a modeling tool developed by Sparx Systems, with the SysML 1.2 module installed. Although all aspects of this process are explicit in the use of EA, there is nothing keeping it from being used with other SysML tools such as MagicDraw. In addition, the examples and explanations that follow are specific to NASA launch vehicle design and have a much greater emphasis on things like “Mission” and “Crew”. Though “Mission Success” and “Crew Safety” are specific to human-rated space vehicle design, the ideas that they convey can be used in many systems. For example a Dishwasher has a “Mission” to clean dishes while the “Crew” could refer to the person using the dishwasher or even the dishes themselves.

A. GFT Model Process

The GFT Model Process is recursive; it continually repeats itself until the desired level of abstraction is reached for the system being modeled. The process description is divided into two main parts: the Nominal GFT Process and the Off-Nominal GFT Process.

1. Nominal GFT Process

The Nominal GFT Process is distributed across four different modeling views; each view has its own set of diagrams, which are described in detail in Section III.A. Additionally there are distinct constructs and construct connections that are used within each modeling view. These are described in Sections III.B and Section III.C, respectively. Figure 1 below shows the process graphically.

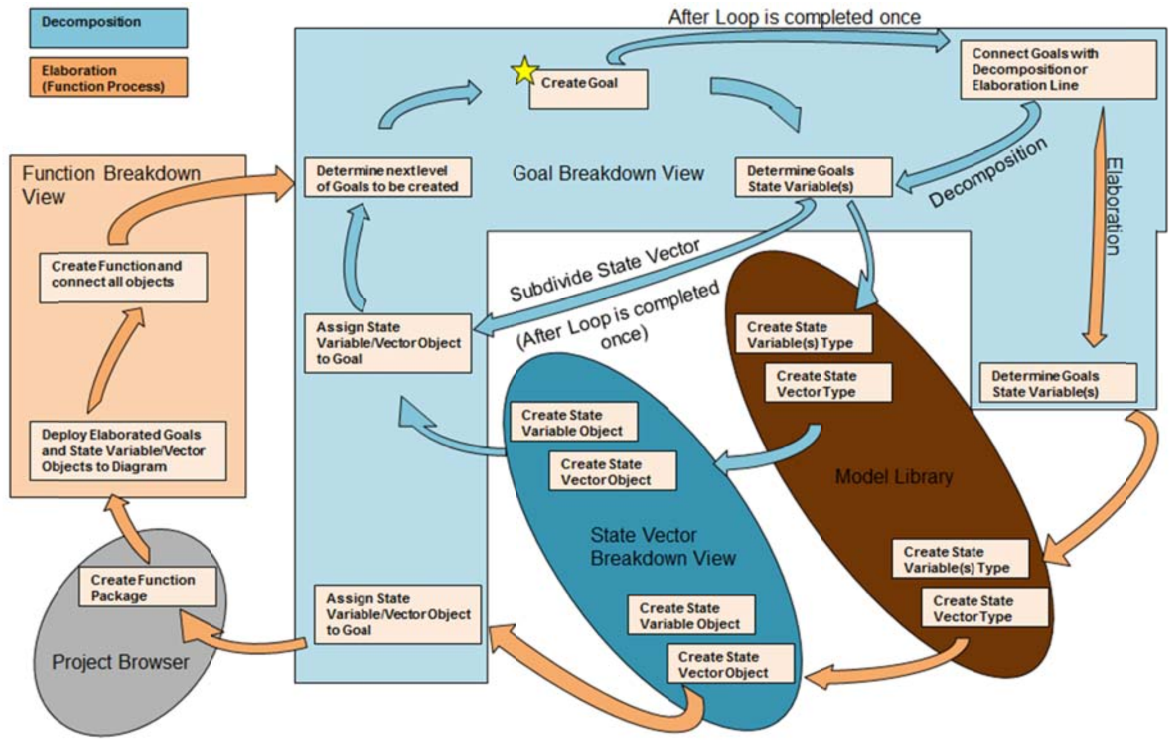

Decomposition
Elaboration (Function Process)
After Loop is completed once
Create Goal
Connect Goals with Decomposition or Elaboration Line
Decomposition
Elaboration
Function Breakdown View
Create Function and connect all objects
Deploy Elaborated Goals and State Variable/Vector Objects to Diagram
Determine next level of Goals to be created
Goal Breakdown View
Determine Goals State Variable(s)
Subdivide State Vector (After Loop is completed once)
Assign State Variable/Vector Object to Goal
Create State Variable(s) Type
Create State Vector Type
Determine Goals State Variable(s)
Create State Variable Object
Create State Vector Object
Model Library
State Vector Breakdown View
Create State Variable(s) Type
Create State Vector Type
Create State Variable Object
Create State Vector Object
Create Function Package
Project Browser
Assign State Variable/Vector Object to Goal

Figure 1: Nominal GFT Development Process

The Nominal GFT Process starts with the creation of a Goal in the Goal Breakdown View. The first Goal in the GFT will be a high level goal that represents the system's overall purpose. Goal diagrams, constructs, and construct connections are described in Section III.A.1, Section III.B.1, and III.C.1, respectively. Once the top level Goal is created the modeler must determine the Goal’s State Variable(s). This is done outside of the model with input from subject matter experts to ensure that all of the State Variables that are needed for the success of the Goal are captured. Once the State Variables have been determined, they are created within the model library as a variable type and then grouped into a State Vector. State Variable and Vector diagrams, constructs, and construct connections for use within the model library; these are described in Section III.A.4, Section III.B.3, and Section

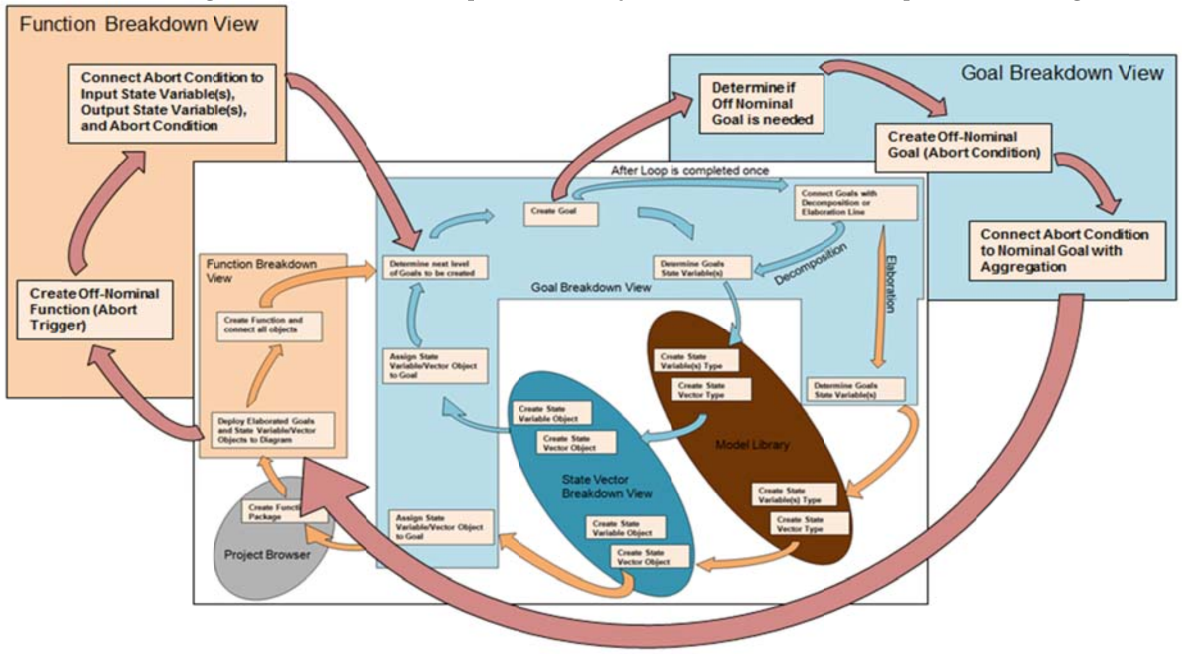
III.C.4, respectively. Now that the Variable types are in the model they can be classified as objects in the State Vector Breakdown View for the specific use by the Goal. State Variable and Vector diagrams, construct, and construct connections, for use within the State Vector Breakdown View is described in Section III.A.3, Section III.B.3, and Section III.C.3, respectively. With the Goal, and Variable/Vector Object existing within the model they can be connected to the associated Goal. The Variable/Vector Object is "Deployed" onto the Goal Diagram and connected to the Goal. *Deployed is the term used to describe the act of taking a construct from one part or view of the model (usually the Model Library) and placing it onto a diagram in another part of the model.*

With a Goal and its associated Variables/Vectors being modeled, the next level of the tree can be determined and created. As Goals are created further down the tree, they must be connected to higher-level Goals using a Decomposition or an Elaboration. If a set of Variables associated with a Goal can be logically subdivided without introducing any new State Variables, then a Decomposition is used. When using a Decomposition, no new State Variables are created within the Model Library or State Vector Breakdown View since they already exist within the model. Therefore the decomposed Goal is linked to a pre-existing State Vector/Variable(s) that is/are deployed to the Goal Diagram from State Vector Breakdown View. The process of breaking down the Goals and their higher level vectors into smaller Vectors and Variables continues until new Variables are needed to continue describing the system. At this point an Elaboration is needed. This in turn implies the need for a Function. Function diagrams, constructs, and construct connections are described in Section III.A.2, Section III.B.2, and Section III.C.2, respectively. Once an Elaboration connection is established, a Function must accompany it in order to create the new Variables. This follows the equation y=f(x) where y is the variable attached to the higher level goal, x is the variable attached to the lower level goal, and f is the function that creates y from x.

The modeler must define the Variables attached to the elaborated Goal outside of the model with inputs from subject matter experts and then incorporate them into the Model Library as Types and the State Vector Breakdown view as Objects. Then the State Variable/Vector Object can be deployed onto the Goal Diagram. However, with the existence of a Function, a Function Package must be created within the Function Breakdown View, deployed to the Goal Breakdown View, and connected as described in Section III.B.2 and Section III.C.2. With the creation of the Function Package, a Function Diagram can be manipulated within the model. This is where the details of the Elaboration of one Variable into another are maintained. With the process being cyclical, it repeats the decomposition process until a new Function is needed. This continues until the system is described to the lowest level needed for the work being done.

2. *Off-Nominal GFT Process*

Once the system is defined using the nominal GFT process, it can be analyzed for placement of Off-Nominal Functions and Goals. Figure 2 below shows this process in conjunction with the nominal process from Figure 1.



Function Breakdown View
Connect Abort Condition to Input State Variable(s), Output State Variable(s), and Abort Condition
Create Off-Nominal Function (Abort Trigger)
Goal Breakdown View
Determine if Off Nominal Goal is needed
Create Off-Nominal Goal (Abort Condition)
Connect Abort Condition to Nominal Goal with Aggregation
After Loop is completed once
Create Goal
Connect Goals with Decomposition or Elaboration Line
Determine next level of Goals to be created
Determine Goals State Variable(s)
Decomposition
Elaboration
Goal Breakdown View
Function Breakdown View
Create Function and connect all objects
Deploy Elaborated Goals and State Variable/Vector Objects to Diagram
Assign State Variable/Vector Object to Goal
Create State Variable(s) Type
Create State Vector Type
Determine Goals State Variable(s)
Create State Variable Object
Create State Vector Object
Model Library
State Vector Breakdown View
Create Function Package
Project Browser
Assign State Variable/Vector Object to Goal
Create State Variable(s) Type
Create State Vector Type
Create State Variable Object
Create State Vector Object

Figure 2: Off-Nominal GFT Development Process

When determining if an Off-Nominal Goal is needed, the modeler must take into account the State Variables that the Off-Nominal Function will monitor. Off-Nominal Goals can exist without an Off-Nominal Function, but if this occurs it indicates that there is no monitoring or detection mechanism to determine if the State Variable(s) is outside of its nominal range. This is known as a "Non-Monitored Condition". Off-Nominal Goals are created in the same way as Nominal Goals, but are connected to other Goals using an Aggregation. Additionally, an Off-Nominal Goal is usually created at the same place that an Elaboration exists. With the Off-Nominal Goal created in the Goal Breakdown View, the Off-Nominal Function is then added to the associated Function Breakdown View that was created by the original nominal Elaboration. This Off-Nominal Function takes the y variables in the equation $y=f(x)$ as inputs, and outputs a notification variable to be used for analysis once the model is completed. Usually (the exception being Caution and Warning, which is merely a notification), the Off-Nominal Function activates a system response of some kind, which for Goal Changes must be modeled as a new GFT, because it represents a new set of system Goals, Functions, and activities.

III. SysML Baseline

Systems Modeling Language (SysML) is commonly used for systems engineering applications and supports the specification, analysis, design, and verification and validation of diverse systems. The language uses a specific set of diagrams and rules to visually describe the system being modeled. In the development of the GFT only a select few SysML diagram types and rules were used. When the limitations of the baseline SysML language were reached, additions were made to the language through the use of the stereotype function within the language. Additionally, a strict set of conventions were developed and implemented to ensure standardized application of the theoretical principles explained in detail within the AIAA InfoTech@Aerospace 2013 conference paper "Goal-Function Tree Modeling for Systems Engineering and Fault Management" presented by Dr. Stephen B. Johnson.

A. GFT Modeling Views

The GFT uses three types SysML Baseline diagrams: Requirement Diagrams, Activity Diagrams, and Block Diagrams. The SysML Requirement Diagram is known as the Goal Diagram and is used to graphically represent the breakdown of the systems Goals. These diagrams make up the Goal Breakdown View of the model which is described in Section III.A.1. The SysML Activity Diagram is used in two ways. First, it shows the detailed information between goals and functions in a Function Diagram which make up the Function Breakdown View of the model described in Section III.A.2. Second, it describes the interrelations between the State Variables and State vector objects which makeup the State Vector Breakdown View described in Section III.A.3. The SysML Block Diagram is used to maintain and create the State Variable and State Vector types that are maintained and created within the Model Library described in Section III.A.3.

The GFT process starts with the definition of four different modeling views: the Goal Breakdown View, the Functional Breakdown View, the State Vector Breakdown View, and the Model Library. Each of these views is an extrapolation of the construct and construct connection sections described below. Finally, a Navigation View enables quick navigation through the system model and its various diagrams, and enables global review of the model structure.

1. *The Goal Breakdown View*

The Goal Breakdown View contains all of the System Goals and Goal Diagrams, and can be accessed from the project browser. The Goal Breakdown View is broken up into different packages in order to subdivide the modeling view into smaller, more manageable diagrams. In the SLS Project implementation, these packages specific to SLS and are referred to as: the Top Level Goal Breakdown Package, the Mission Success Goal Breakdown Package, the Crew Safety Goal Breakdown Package, and the Abort Package. The Top Level Goal Breakdown Package holds the top level of the tree where most of the Goals are Achievement Goals. These Achievement Goals are decomposed to a level where the system's phases and modes of operations become logically apparent. In the case of the Launch Vehicle example, these are the vehicle mission phases, whereas for a dishwasher it would consist of the dishwasher operating modes such as defining the wash options, loading and unloading modes, and the wash and rinse cycles themselves. Once the system is decomposed into its operational phases a package is created for each within the Mission Success Goal Breakdown Package. Individualizing the operational phases within the Mission Success Goal Breakdown Package is done so that differing teams can work on the different mission phases independently of each other. It also compartmentalizes the system in a functional way so that different nominal ranges can be assigned to the same Goals and Variables/Vectors across operational phases. Next, the Crew Safety Goal Breakdown Package, which is very specific to launch vehicles, will hold the goals that are unique to keeping the crew safe and not to the attainment of nominal orbit (that is, mission success). For example, axial roll rate must not only be controlled for launch vehicle structural reasons, but also for crew safety reasons. Of the two limits, the crew safety limit will typically be reached first due to the fact that the crew will lose the ability to perform their mission adequately due to axial roll rate before it causes a structural failure. The Abort Package is used to maintain all of the Abort Condition Goals, for analytical reasons. Additionally, it is inherently true that an abort is not done in order to successfully complete a mission, but rather for the sole purpose of protecting the crew.

Figure 3 shows an example of a Goal Diagram in the Goal Breakdown View. It shows the breakdown of one of the top level goals in the SLSP GFT developed for NASA. This section of the model starts with the “Deliver Booster CS to CS Separation Point”, which is the first phase of the launch vehicle ascent mission. To successfully complete this goal the vehicle must “Control Trajectory” AND “Maintain Structural Integrity”. Without both of those goals being successful there is no way to successfully complete the higher level goal. Additionally, each of the goals at the second level are broken down into the goals that must be successful for the level two goals to be successful. The diagram also shows the State Variables that must be maintained within a particular range for each of the goals to be successful and the Function Packages that describe the Elaboration of the defined State Variables into new State Variables. State Variables, Function Packages, and Elaborations are defined in later sections of this paper.

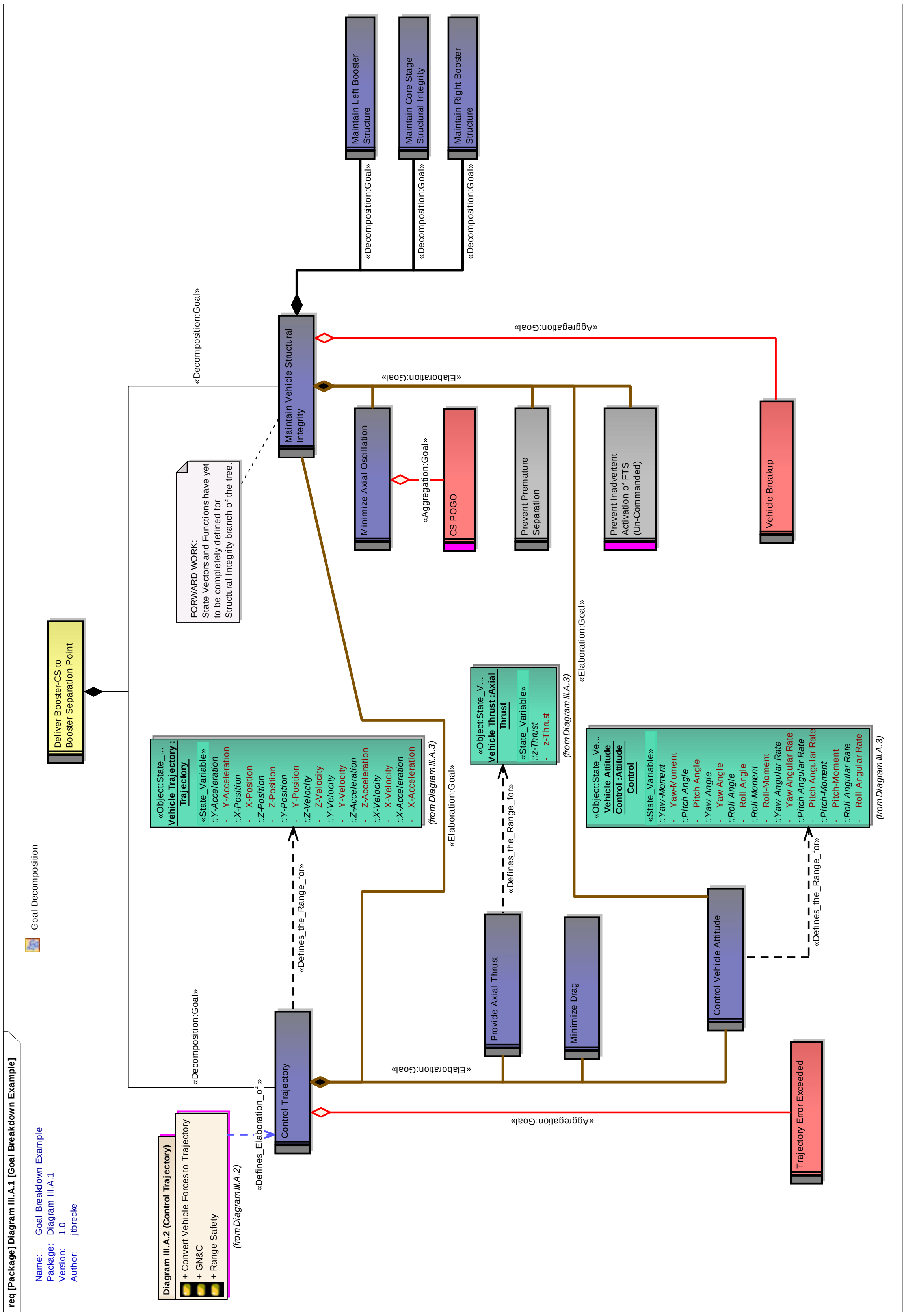


Figure 3: Goal Diagram Example

Nominal Goals in the Goal Breakdown View

To create the diagrams within the Goal Breakdown View, every Goal or SysML Requirement is made composite (a typical SysML option for blocks). This will create a new SysML Requirements Diagram. Each new diagram will have three levels, the top level will be the goal that is being made composite, the second level will be the goals that are being elaborated and/or decomposed from the top level, and the third level will show the next level goals that are being elaborated and/or decomposed. Nominal and Off-Nominal Goals, Function Packages, and System State Variables/Vector Objects will be the only constructs to appear on Goal Diagrams. Only the first and second level of the diagram will show the Function Packages and System State Variables/Vector Objects. Also, only the connections described in Section III.C.1 will be applied to the constructs of a Goal Diagram.

Off-Nominal Goals in the Goal Breakdown View

All of the Off-Nominal Goals appear in the Mission Success branch of the GFT, and are described with an Aggregation connection. However, ONLY Off-Nominal Goals, as described in Section III.B.1, appear in both the Mission Success Goal Breakdown Package AND the Abort Package. The Off-Nominal Goals within the Mission Success Goal Breakdown Package diagrams use the Aggregation connection, while the Off-Nominal Goals within the Aborts Package diagrams use the Decomposition connections.

2. *The Functional Breakdown View*

The Function Breakdown View is used to maintain the Function Packages created during the formation of the Goal Diagrams. These Function Packages are placed within a phase package, and are associated with the systems operational phases that are maintained within the project browser under the Functional Breakdown View. Each Function Package will be associated with an Elaborated Goal and describes all of the details of the creation of new State Variables needed for further Goal Decomposition. The Function Packages, as they appear within the project browser, will consist of nominal and off-nominal Function Constructs, and Function Construct Connections as described in Section III.B.2 and Section III.C.2, respectively. Additionally, each Function Package will create a Function Diagram, or SysML Activity Diagram, for the deployment of System Goals, System State Variables/Vectors, and Nominal/Off-Nominal Functions.

Figure 4 shows an example of a Function Diagram within the Function Breakdown View that is consistent with the Function Package shown in Figure 3. Just as in Figure 3 this diagram is an example pulled from the SLSP GFT and shows the Elaboration of Vehicle Thrust, Vehicle Drag, and Vehicle Attitude Control State Variables into the Vehicle Trajectory State Variables. This diagram shows how the GFT uses the equation y=f(x) by showing that the Thrust, Drag, and Attitude Variables are inputs ‘x’ in to the Nominal Function ‘f’ to create the output State Variables ‘y’ shown here as the Trajectory State Variables. More detail about the constructs and construct connections are defined later in this paper.

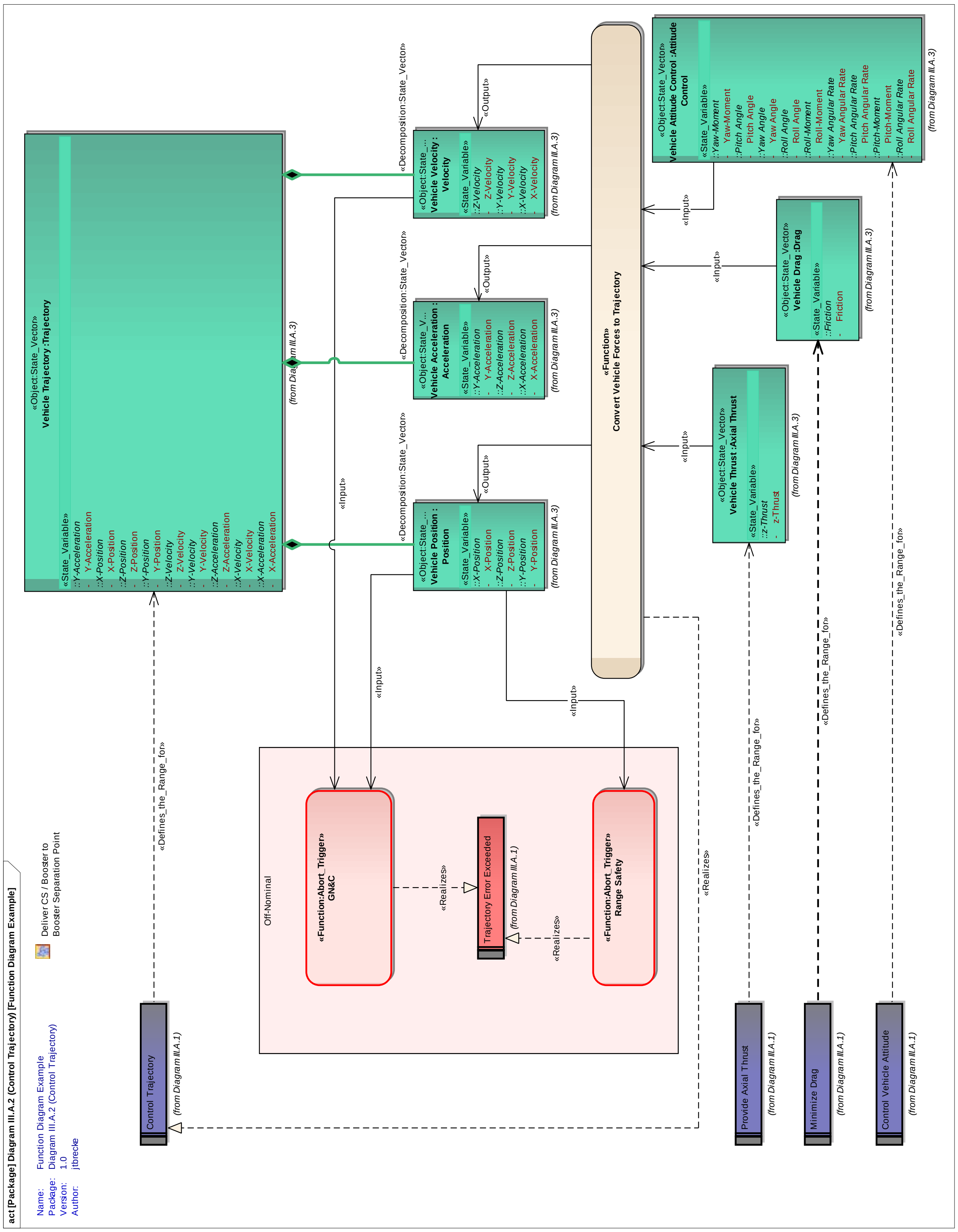


Figure 4: Function Diagram Example

The Function Package on a Goal Diagram will be linked to a Function Diagram. This Function Diagram will show the functional relationship between the original Goal linked to the Function Package and the Elaborated Goals on the Goal Diagram. Therefore, all associated Goals, and their State Variable/Vectors, of the elaboration being explained on the Function Diagram must be deployed to the Function Diagram. After the Goals are deployed, the Function can be created and the construct connections defined in Section III.C.2 can be used to link the lower level Goal State Variable/Vectors to the Function and the Function to the higher level Goal State Variable/Vectors.

Off-Nominal Functions

When an Off-Nominal Goal is aggregated, the Off-Nominal Goal is also deployed to the Function Diagram. On the Function Diagram, the Off-Nominal Function is also deployed, and the State Variable/Vector created by the Nominal Function is then an input to the Off-Nominal Function. This represents the idea that the Off-Nominal Function monitors the State Variable/Vector for deviations from the nominal range set forth by the System State Variable/Vector associated with the System Goal. Occasionally, there is a need to subdivide a State Vector into sub vectors or single variables in order to explain the Goal Elaboration. In this case the construct connections described in Section III.C.3 are used. For Off-Nominal Notification Goals, the Off-Nominal Function has an output State Variable. This State Variable is the notification that a detection is tripped and is associated with an Off-Nominal Functions.

3. *The State Vector Breakdown View*

The State Vector Breakdown View of the GFT Model is a repository for the System State Variable and Vector objects used within the Goal and Function diagrams. The State Vector Breakdown View is separated into packages that represent each of the operational phases of the system being modeled. This is done to allow for different limits to be enforced on the same State Vector across system operational phases. Sub-packages are created within the higher level operational phase package to represent the different system components. In the launch vehicle example, this would be the Main Propulsion System, Thrust Vector Control System, Structural System, and others. It is within these component packages that the State Variables and Vector Objects are maintained. Additionally, each of the component packages hold an activity diagram that is used to detail the interconnections of the State Variables and Vectors independent of, but consistent with the Goal and Function Diagrams. Any System State Variable or Vector Object that is used within the model, such as in a Goal Diagram or a Function Diagram, is **deployed** from this view. Deployment is defined as the act of taking a SysML construct from a place in the model and duplicating it, creating a link to the original, and placing it into another part of the model).

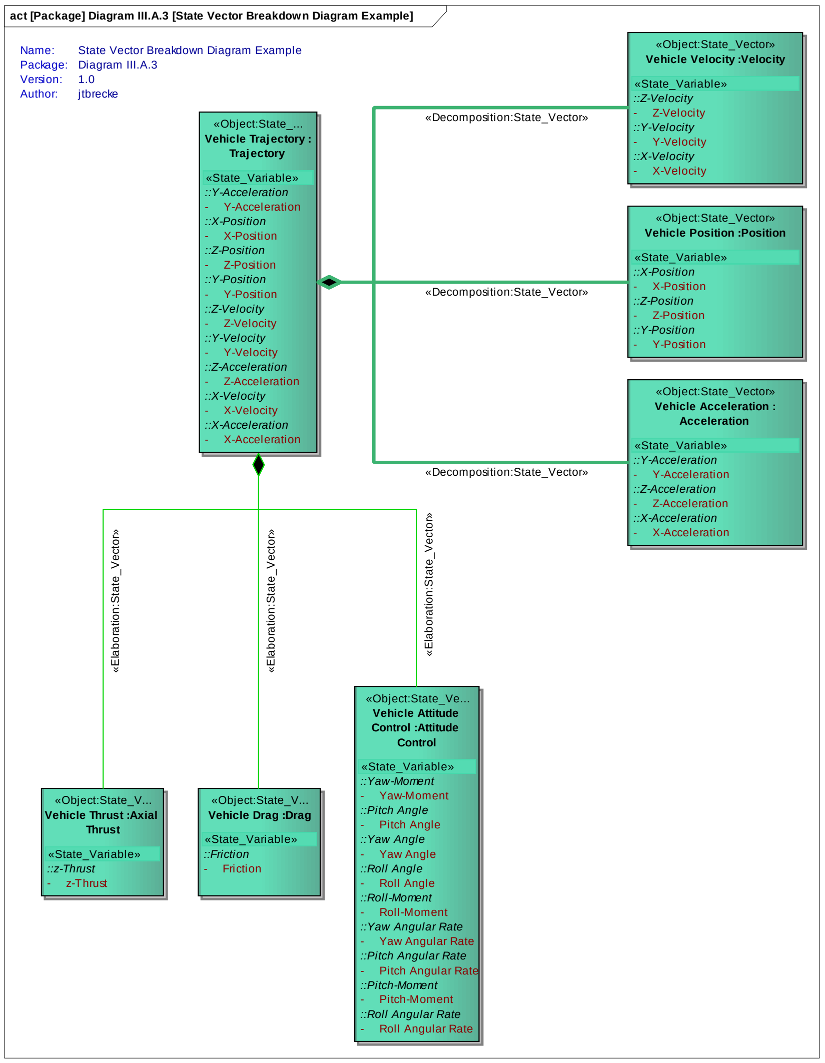


Figure 5: State Vector Breakdown Diagram Example

The State Vector Breakdown View uses two types of construct connections, the **<<Decomposition:StateVector>>**, and the **<<Elaboration:StateVector>>**. These two construct connections are used in the same way that decompositions and elaborations are used within the Goal Diagrams. When a Goal is created and decomposed or elaborated, the Variables and/or Vectors that the elaborated or decomposed Goals are associated with get the same type of linkage within the State Vector Breakdown View. This gives a detailed breakdown of all of the variables and vectors that the system uses to reach its overall Goal without additional Functions or Goals needing to be viewed. It also gives an additional sanity check to the goal breakdown by ensuring that the physical attributes of the system are connected correctly through the goal elaborations and decompositions.

4. The Model Library

The Model Library is the location in the GFT where the State Variables and Vectors originate from. It is also referred to as the State Variable/Vector Type Library. Before a State Variable or Vector Object can be created within the State Vector Breakdown View and then deployed to a Goal or Function Diagram it must first exist within the Model Library as a “Class” or type. By creating types of State Variables and Vectors, the same kind of Variables and Vectors can be reused in multiple places in the model by utilizing the SysML Instance Classifier operation. The Instance Classifier operation creates the State Variable/Vector Objects that are maintained and deployed from The State Vector Breakdown View. The Model Library View is subdivided into packages associated with groupings of State Variables and Vectors. Examples of this, for launch vehicles, include Propellant, Structural, Trajectory, Thrust, and Control properties among others. These groupings are design-specific due to the fact that the State Variables and Vectors are the essential link between a functional model and a physical model. Additionally, these design-specific packages include a block diagram that shows how the State Variables and Vectors are created. The Model library uses two GFT constructs. The **<<Class:State_Variable>>**, and the **<<Class:State_Vector>>**. The Variable Class construct is created first and given a name to represent the variable being created.

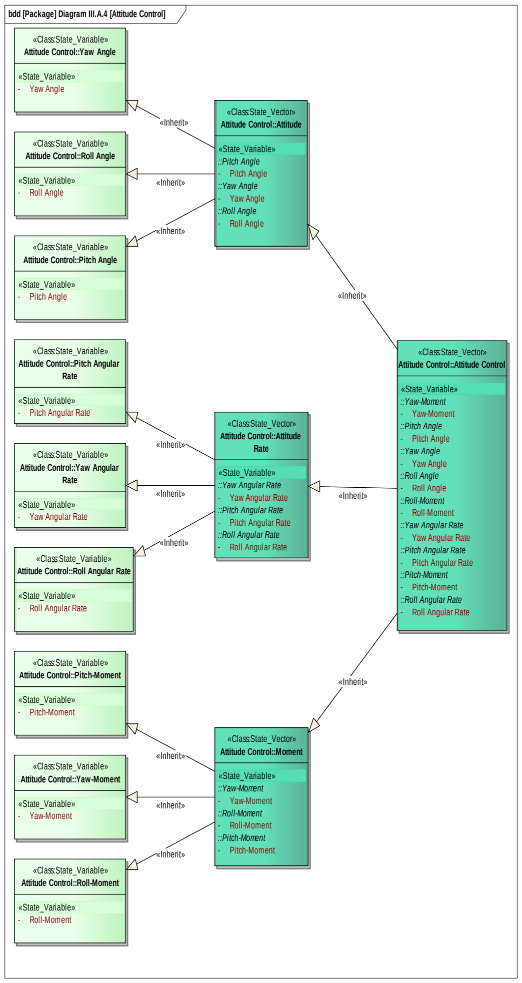

bdd [Package] Diagram III.A.4 [Attitude Control]
«Class:State_Variable» Attitude Control::Yaw Angle
«Class:State_Variable» Attitude Control::Roll Angle
«Class:State_Variable» Attitude Control::Pitch Angle
«Class:State_Vector» Attitude Control::Attitude
«Class:State_Vector» Attitude Control::Attitude Control
«Class:State_Vector» Attitude Control::Attitude Rate
«Class:State_Vector» Attitude Control::Moment
«Inherit»

Figure 6: State Vector Library Diagram Example (Attitude Control)

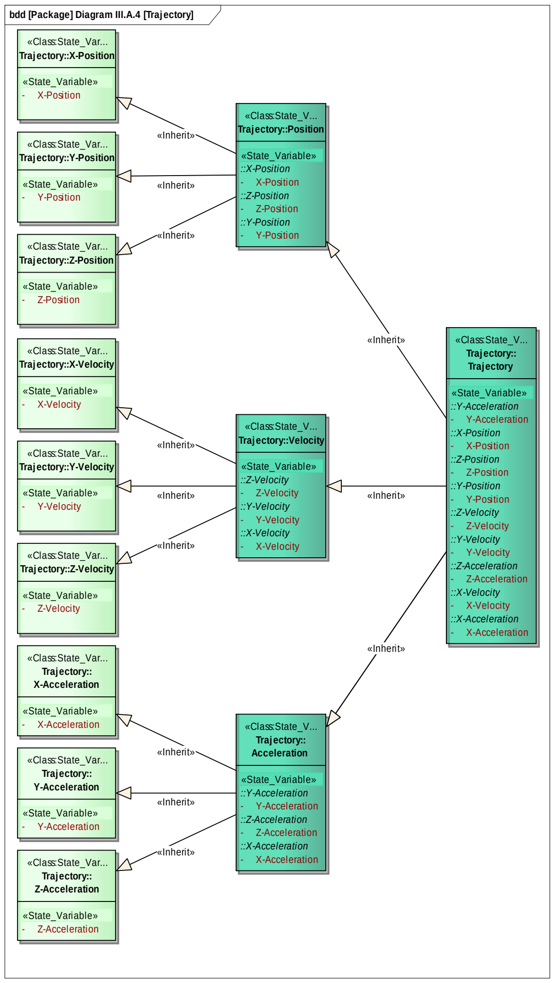

bdd [Package] Diagram III.A.4 [Trajectory]
«Class:State_Var... Trajectory::X-Position
«Class:State_V... Trajectory::Position
«Class:State_V... Trajectory::Velocity
«Class:State_V... Trajectory::Acceleration
«Class:State_V... Trajectory::Trajectory
«Inherit»

Figure 7: State Vector Library Diagram Example (Trajectory)

B. SysML Baseline Constructs

There are 5 SysML constructs used in the GFT:

- Requirements are used to represent System Goals.
- Activities are used to represent System Functions within a Function Diagram
- Packages are used to represent System Functions within a Goal Diagram
- Objects are used to represent specific instances of System State Variables and System State Vectors
- Classes are used to represent differing types of System State Variables and System State Vectors

1. System Goals

System Goals, or simply "Goals", are SysML Requirement constructs with specific Stereotypes that identify their use within the GFT. There are two high-level types of Goals used in the GFT, Nominal and Off-Nominal Goals. Of the two types of high-level Goals, there are three types of Nominal Goals and four types of Off-Nominal Goals.

Nominal Goals

The three Nominal Goals are Achievement Goals, Maintenance Goals, and Prevention Goals. An Achievement Goal is a proposition that must be true in the final state of the Goal and is used to indicate the end of a phase or set of phases that the system must go through to be successful. The Achievement Goal is denoted by a golden color and has the Stereotype **<<Goal_Nom:Achievement>>**. A Maintenance Goal is a proposition that must be true in every state over time for the associated State Variables. One or more Maintenance Goals together make up a time-phase that sometimes culminate in an Achievement goal. The Maintenance Goal is denoted by a purple color and has the Stereotype **<<Goal_Nom:Maintenance>>**. A Prevention Goal is a proposition where the system must inhibit an event or action. Prevention Goals are used in association with Achievement Goals, and indicates the goal of "Not performing something". The Prevention Goal is denoted by a gray color and has the Stereotype **<<Goal_Nom:Prevention>>**.

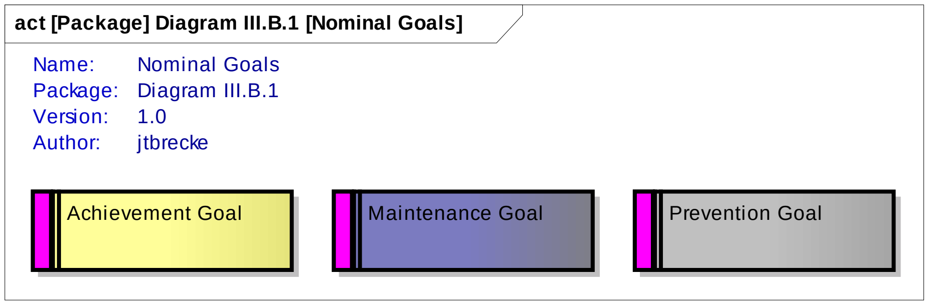


Figure 8: Nominal Goals

Off-Nominal Goals

The four Off-Nominal Goals, which are specifically associated with the SLSP, are Abort Goals, Caution & Warning (C&W) Goals, Redundancy Management (RM) Goals, and Safing Goals. An Abort Goal is a proposition to notify the system for Loss of Mission (LOM) conditions. It implies the creation of a function that performs the task of monitoring for the particular states and behaviors declared in the Abort Goal, and ultimately leading to an abort response to remove the crew (astronauts) from an impending or currently occurring hazardous situation, such as an exploding launch vehicle or loss of vehicle control. Specifically, an Abort Goal will indicate a position in the tree where the system's behaviors will be monitored, so as to ultimately activate an abort to prevent Loss of Crew (LOC). In all cases the mission is lost and the crew must be returned safely to Earth. The Abort Goal is denoted by a red color and has the Stereotype **<<Goal_OffNom:Abort>>**. A C&W Goal is a proposition to notify the system for warning alerts. It implies a function that performs the task of monitoring for the particular issues declared in the C&W Goal and sending a notification to the crew if the monitored behavior occurs. Specifically a C&W Goal will indicate a position in the tree that monitors for a degraded function and a notification to the crew could/should be sent. The C&W Goal is denoted by an orange color and has the Stereotype **<<Goal_OffNom:C&W>>**. An RM Goal is a proposition to provide either passive or active management of redundant system capabilities in order to maintain overall system functionality and still continue the mission. Specifically, an RM Goal will indicate where redundant systems will be used to respond to a failed or degraded state of one of several redundant components in the system. The RM Goal is denoted by a blue color and has the Stereotype **<<Goal_OffNom:RM>>**. A Safing Goal is a proposition that, if a critical failure occurs, will change the state of the system into a "Safe State" that prevents or mitigates further damage to the system. The Safing Goal is denoted by a lavender color and has the Stereotype **<<Goal_OffNom:Safing>>**. An example of a Safing goal is the shutdown of a liquid propellant rocket engine to prevent it from exploding and causing a direct and immediate hazard to the crew. Once shut down, the mission might or might not be lost, depending on when the engine shutdown occurs during a launch vehicle ascent.

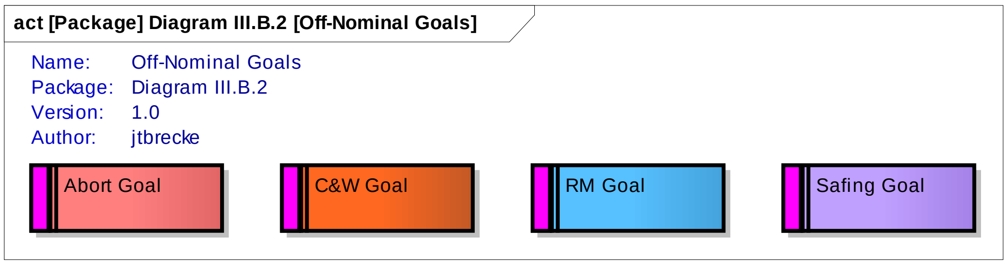


Figure 9: Off-Nominal Goals

As the GFT was initially developed exclusively for launch vehicle design, the above Off-Nominal Goals will not be suitable in all circumstances. Therefore, the analyst will need to define appropriate off-nominal goals for his or her system. In general, Fault Management theory, as described in Chapter 1 of *System Health Management: with*

Aerospace Applications, indicates general categories such as failure detection, failure prognosis, failure recovery, goal change, and others.[iii]

2. *System Functions*

System Functions appear in two diagrams within the GFT: Goal Diagrams and Function Diagrams. A Function as it appears in a Goal Diagram is simply a SysML Package. This package construct is used to define a location in the model where further detail is needed. A Package in a Goal Diagram leads to a Function Diagram. The Function Diagram, in turn, shows explicitly which State Variable/Vectors are used by Nominal and Off-Nominal Functions to create new State Variables/Vectors. Once a Function Diagram is created, the Functions themselves must be created. The actual Function is a SysML activity that transforms one or more input State Variable into a different set of output State Variables. All Functions are "realizations" of their associated Goals and cannot exist independent of a Goal. There is only one type of Nominal Function but there are four Off-Nominal Functions, each correlating to an Off-Nominal Goal: Abort Function, C&W Function, RM Function, and Safing Function with the respecting Stereotype tags: **<<Function:Abort_Trigger>>**, **<<Function:C&W_Trigger>>**, **<<Function:RM>>**, and **<<Function:Safing>>** and their respective color associations.

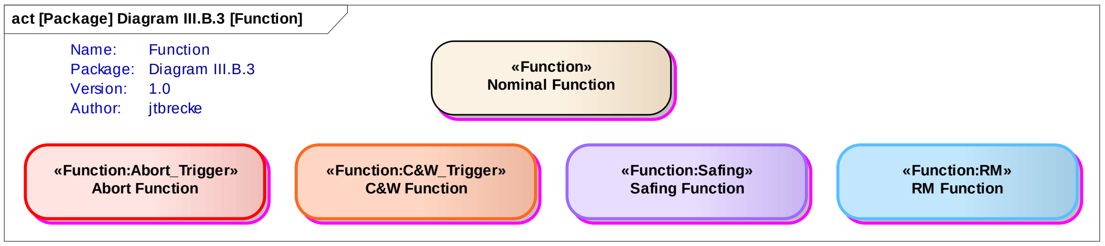

act [Package] Diagram III.B.3 [Function]
Name: Function
Package: Diagram III.B.3
Version: 1.0
Author: jtbrecke
«Function» Nominal Function
«Function:Abort_Trigger» Abort Function
«Function:C&W_Trigger» C&W Function
«Function:Safing» Safing Function
«Function:RM» RM Function

Figure 10: Nominal and Off-Nominal Functions

3. *State Variables and Vectors*

A State Variable is defined as *a physical attribute of a system that must be maintained within an appropriate range for the success of an assigned goal*. In the GFT, SysML Object and Class constructs are used to define State Variables. A SysML Class construct is used to create a generic State Variable that is used in multiple places within the model. The State Variable Class is denoted with a stereotype **<<Class:State_Variable>>**. A SysML Object construct is used to create a specific instance of a State Variable that is used in a specific location within the model. The State Variable Object is denoted with a stereotype **<<Object:State_Variable>>**. An example of the difference between the class the instance is "Pressure" being a State Variable Class and "Helium Tank B Pressure" being a State Variable Object. Both are 'Pressures' but the first is generic and the second is specific. State Variables start out as Classes; when created, an attribute is given to the variable whose name is the same as the variable itself. When a series of variables can logically be combined, depending on the physics of the system being modeled, a State Vector is created. A State Vector is a combination of the attributes from the individual State Variables that make up the State Vector. The State Vector Class is denoted with a stereotype **<<Class:State_Vector>>**. Additionally, this State Vector Class is a generic state vector that can be used in many places in the model. The specific state vector uses the SysML Object and is denoted with a stereotype **<<Object:State_Vector>>**, in much the same way the Class and Object State Variables are used.

To create a State Variable/Vector Object from a Class, the "Instance Classifier" function within the SysML language is used. Also, it is important to note that State Variable/Vector Objects are maintained within a different model view than the State Variable/Vector Classes. This is described in the Modeling Views section of the paper.

[iii] Stephen B. Johnson, et al., eds. *System Health Management: with Aerospace Applications*, Chichester, UK: John Wiley UK, 2011, chapter 1.

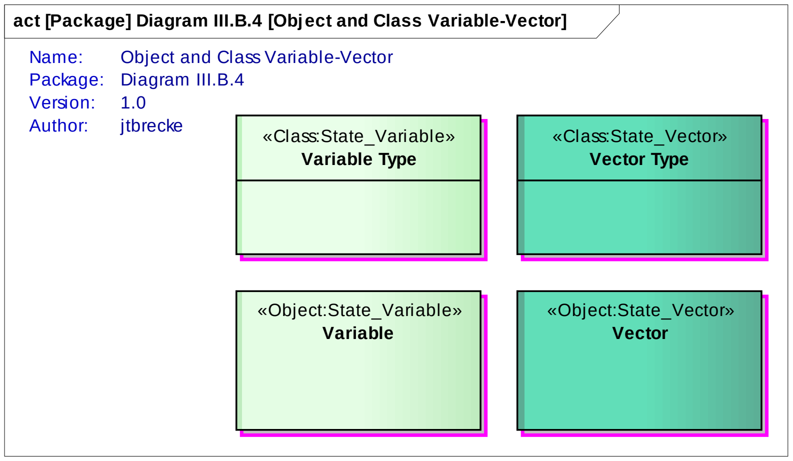

act [Package] Diagram III.B.4 [Object and Class Variable-Vector]
Name: Object and Class Variable-Vector
Package: Diagram III.B.4
Version: 1.0
Author: jtbrecke
«Class:State_Variable» Variable Type
«Class:State_Vector» Vector Type
«Object:State_Variable» Variable
«Object:State_Vector» Vector

Figure 11: State Variable/Vector Classes and Objects

C. SysML Baseline Construct Connections

SysML has a wide variety of construct connections that can be utilized by the modeler, and though the GFT uses many of the baseline connections from the language they are extremely precise in their applications. The following sections will explain in detail the construct connections that are used within the GFT.

1. Goal Diagram Construct Connections

As stated previously, Goal Diagrams are SysML Requirement Diagrams that show the hierarchical links between different Goals within the GFT. Goal Diagrams are used to show the links from Goal to Goal, Goal to System State Variable/Vector, and Goal to Function Package.

Goal to Goal Connections

The GFT process uses three types of Goal to Goal connections: Goal Decomposition, Goal Elaboration, and Goal Aggregation. Each of these connections are based on SysML norms but are defined specifically for GFT use. A Goal Decomposition is a segmentation into a complete set of sub goals necessary to achieve the higher level goal. To determine when a goal decomposition is needed, the State Vector must be evaluated. If the State Variables within a State Vector can be partitioned into a subset of State Vectors without introducing any new State Variable(s), then a goal decomposition is appropriate. For example, in the State Vector(x_1, x_2, x_3, v_1, v_2, v_3, a_1, a_2, a_3), where x's correspond to position components, v's correspond to velocity components, and a's correspond to acceleration components, the State Vector is often decomposed into three State Vectors, one for position alone, another for velocity alone, and a third for acceleration alone. If a new State Variable, or set of state Variables, are needed then a functional Elaboration is needed. In the previous example, if acceleration is generated by thrust from a rocket engine, then the next lower-level goal that supports the acceleration goal will be a thrust goal, with a new thrust State Vector that has magnitude and direction components. Introducing new State Variables in this way is not a decomposition; rather, it is an elaboration. A new function will then be introduced, in which the input State Variables will represent thrust components, and the output State Variables will include acceleration components.

Goal Decomposition connectors are standard SysML Decomposition lines stereotyped as <<**Decomposition:Goal**>>. A Goal Elaboration is a transformation of the State Vector; it implies that a function or set of functions with new State Variables are needed to continue development of the tree along the particular path. A function is added to transform the newly introduced State Variable(s) into the higher level existing State Variable(s). Goal Elaboration connectors are standard SysML Decomposition lines stereotyped as **<<Elaboration:Goal>>.** Usually an added visual cue is given to the line such as an added line thinness and color change.

A Goal Aggregation is used to indicate that there is a Goal to monitor the attributes of a State Vector to determine if the states observed from the Vector's State Variables will diverge from their nominal ranges. The Goal Aggregation is only used to aggregate the off-nominal Goals within the lower level goals. This gives a clear visual cue to where the off nominal goals are within the GFT and allows for analysis of which State Variables are monitored and protected from going outside of their nominal range. Goal Aggregation connectors are standard SysML Aggregation lines stereotyped as <<**Aggregation:Goal**>>. A SysML Aggregation connection is used due to the fact that the success of the higher level goal does not depend on the success of the off-nominal goal. All Goal to Goal connections point FROM the lower level goal TO the higher level goal.

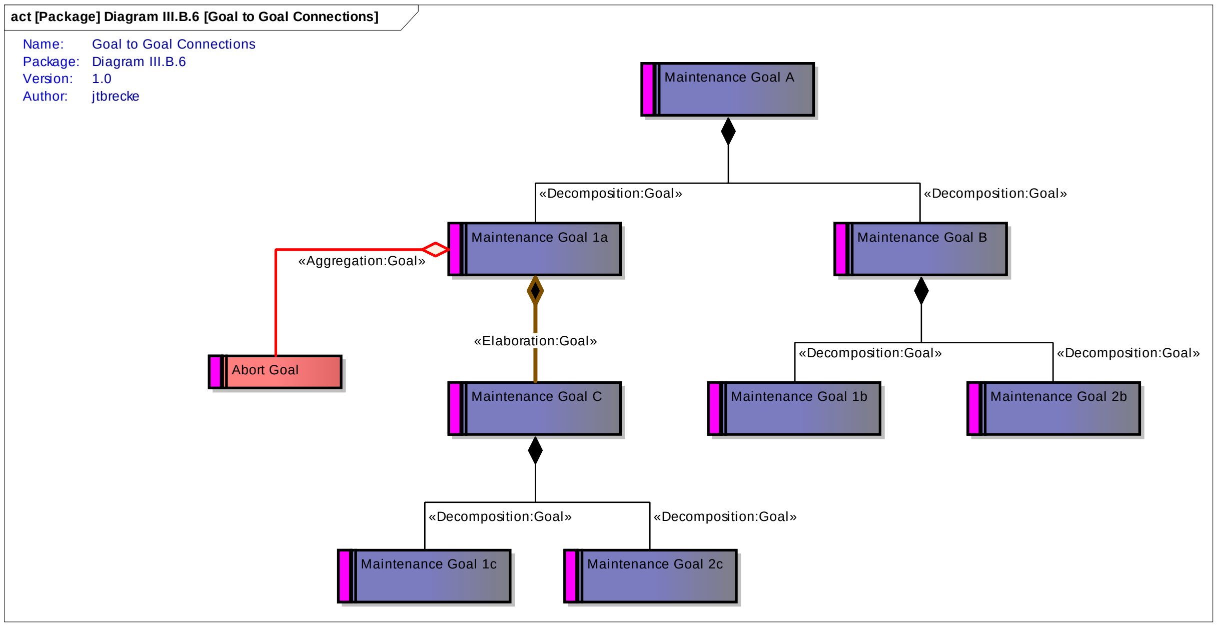


Figure 12: Goal to Goal Construct Connections

Goal to State Variable/Vector Connections

A Goal Diagram also shows the relationship between a Goal and its State Variable/Vector. A SysML Dependency Line, stereotyped as <<**Defines_the_Range_for**>>, is used to show the connection between a Goal and its State Variable or Vector. Only the specific Object version of a State Variable/Vector is used on a Goal Diagram, and the arrow points FROM the Goal TO the State Variable/Vector Object.

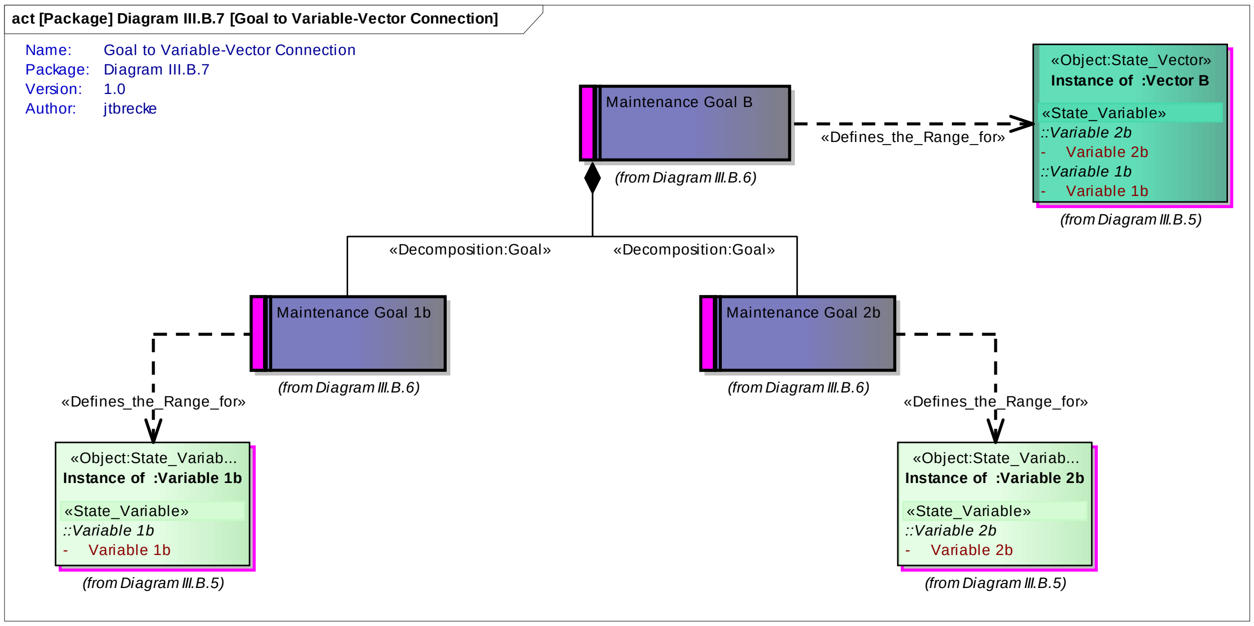


Figure 13: Goal to State Variable/Vector Construct Connections

Goal to Function Package Connections

When a Goal Elaboration is used, and a new State Variable/Vector must be created, a Function Package is created and placed on the Goal Diagram to explicitly define that transformation. The SysML Activity Diagram, or Function Diagram, that is created alongside the Function Package contains the explicit transformation. The Function Package is connected to the Goal that is being elaborated on the Goal Diagram with a standard SysML Dependency Line that is stereotyped as <<**Defines_Elaboration_of**>>.

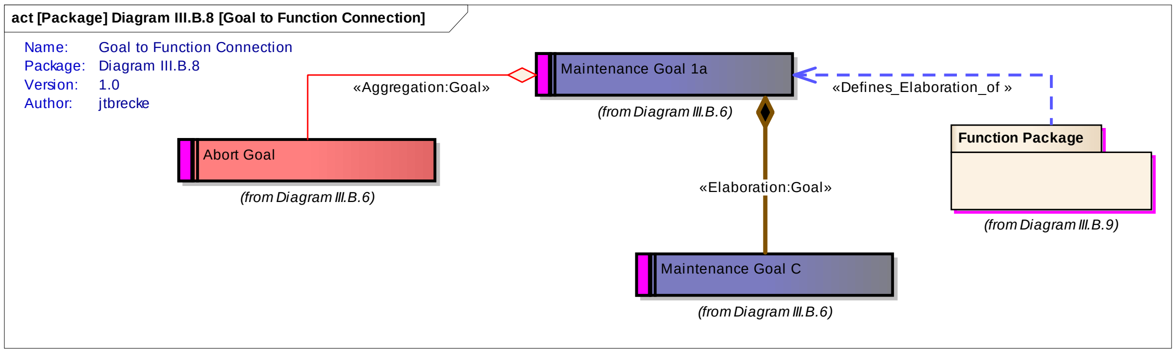


Figure 14: Goal to Function Construct Connections

2. Function Diagram Construct Connections

A Function Diagram shows the connection of the input State Variables to the Function, and of the Function to its output State Variables, thus defining on a single diagram the equation y = f(x). The Goal associated with the Function is the limitation on the range of the output State Variables "y". Function Diagrams consist of many of the previously defined constructs such as Goals and State Variables and Vectors, and connections such as Goal to State Variable/Vector Connections. Therefore, it is only necessary to define connections between State Variables/Vectors to Functions, and Functions to Goals. The connections between State Variables/Vectors to Functions use a standard SysML Object Flow Line. This connection is stereotyped in two ways: <<**Input**>>, and <<**Output**>>. The Input Object Flow is used to show which System State Variable/Vector(s) "x" flow into the Function, and the Output Object Flow is used to show the output state vector "y" flow out of the Function. When relating back to the Goal Diagram the higher level Goal is associated with the State Variable/Vector "y" and the lower level Goal is associated with the State Variable/Vector "x".

The connection of the Function to the Goal is denoted with a standard SysML Realization line and is stereotyped <<**Realizes**>>. This is used to graphically show which goal creates the function

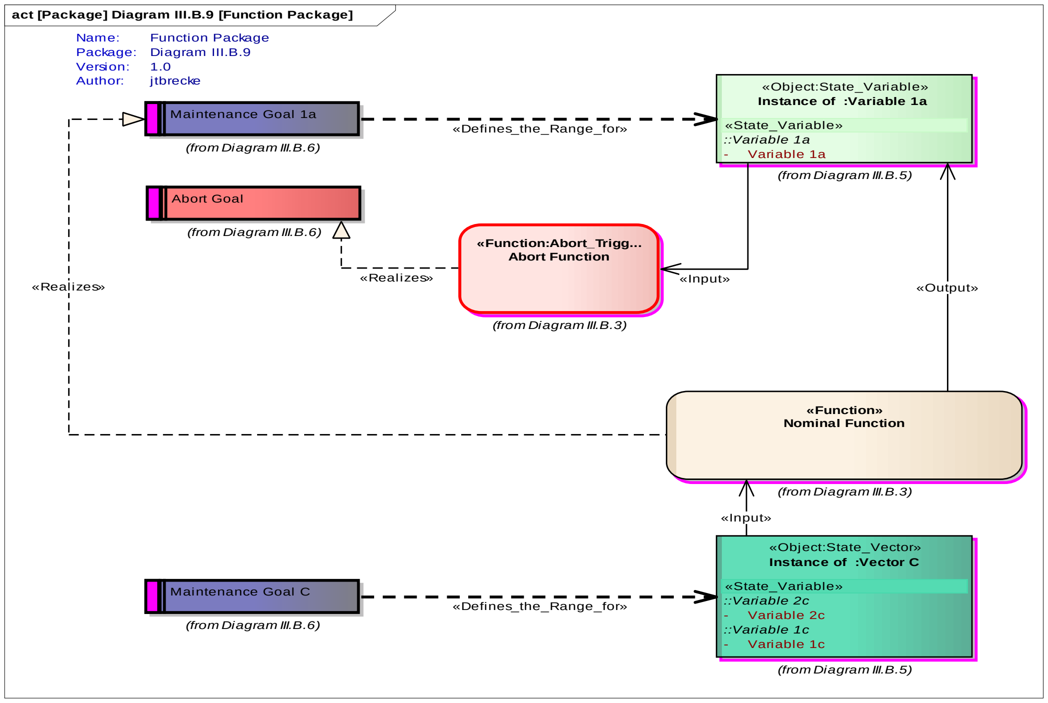


Figure 15: Function Package Diagram Construct Connections

3. System State Vector Object Construct Connections

The State Vector object decompositions and elaborations are created and maintained separately from the Goal Diagrams. These Vector Breakdown Diagrams are created in concert with the Goal Diagrams so that they are relatable and consistent with each other. These diagrams are explained in detail within the Modeling View section of this paper. However, there are some construct connections that must be defined first.

When a Goal and its associated State Vector is decomposed within the Goal Diagram the State Vector must also be decomposed. A State Vector Decomposition shows the partitioning of one State Vector into its sub State Vectors. For example, a single State Vector consisting of all positions, velocities, and accelerations together can be decomposed to three State Vectors: one for position, one for velocity, and one for acceleration. If taken to the lowest level possible, a State Vector can be eventually decomposed to its individual State Variables. In our example, the position State Vector can be decomposed to the three position State Variables (in whichever coordinate system is appropriate). This is done within the State Vector Breakdown View of the model with a standard SysML Decomposition line stereotyped as **<<Decomposition:StateVector>>**. Just as Goals can be Elaborated and not just Decomposed so to can a State Vector. A State Vector Elaboration shows the connection between two State Vectors that have been elaborated by a Function. It is used when one or more new State Variables are introduced that did not exist in the higher level State Vector. This is done with a standard SysML decomposition Line stereotyped as **<<Elaboration:StateVector>>**.

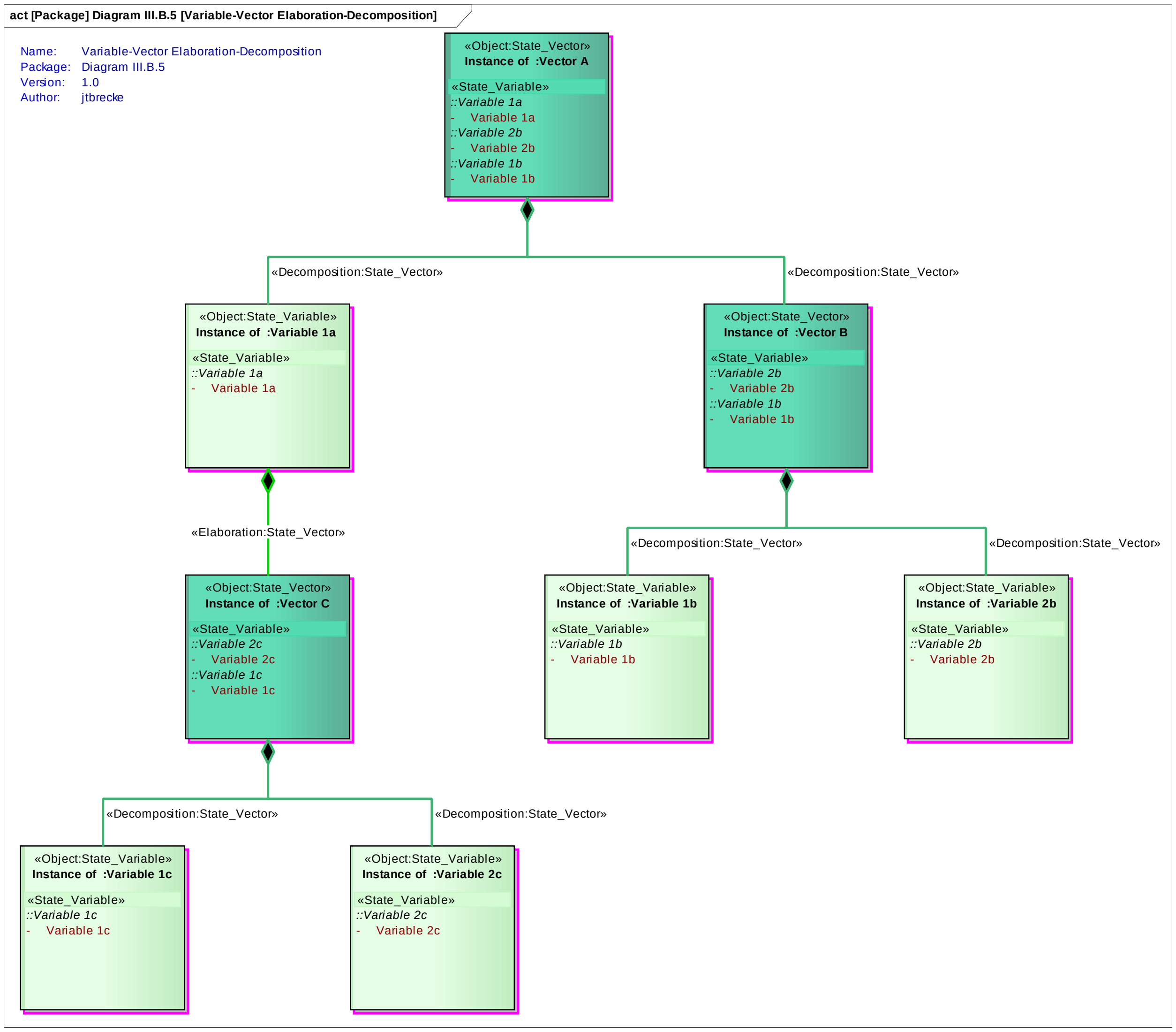

act [Package] Diagram III.B.5 [Variable-Vector Elaboration-Decomposition]
Name: Variable-Vector Elaboration-Decomposition
Package: Diagram III.B.5
Version: 1.0
Author: jtbrecke
«Object:State_Vector» Instance of :Vector A
«State_Variable» ::Variable 1a - Variable 1a ::Variable 2b - Variable 2b ::Variable 1b - Variable 1b
«Decomposition:State_Vector»
«Decomposition:State_Vector»
«Object:State_Variable» Instance of :Variable 1a
«State_Variable» ::Variable 1a - Variable 1a
«Object:State_Vector» Instance of :Vector B
«State_Variable» ::Variable 2b - Variable 2b ::Variable 1b - Variable 1b
«Elaboration:State_Vector»
«Decomposition:State_Vector»
«Decomposition:State_Vector»
«Object:State_Vector» Instance of :Vector C
«State_Variable» ::Variable 2c - Variable 2c ::Variable 1c - Variable 1c
«Object:State_Variable» Instance of :Variable 1b
«State_Variable» ::Variable 1b - Variable 1b
«Object:State_Variable» Instance of :Variable 2b
«State_Variable» ::Variable 2b - Variable 2b
«Decomposition:State_Vector»
«Decomposition:State_Vector»
«Object:State_Variable» Instance of :Variable 1c
«State_Variable» ::Variable 1c - Variable 1c
«Object:State_Variable» Instance of :Variable 2c
«State_Variable» ::Variable 2c - Variable 2c

Figure 16: State Variable/Vector Object Construct Connections

4. Block Diagram Construct Connections

Block Diagrams are used in the GFT to create types of State Variables. These types of State Variables are used within the Block Diagram to create State Vectors by passing the variable attributes from a set of variables to a single vector. This allows for the creation and maintenance of all of the State Variable/Vectors in one location. From this location, the analyst can then instantiate the Variable/Vector type to a specific State Vector Object in the relevant SysML diagrams. This process is explained further within the Modeling Views section of this paper.

The GFT Block Diagram only uses System State Variable and Vector Classes. The only construct connection used on this diagram is the stereotyped SysML Generalization line, **<<Inherit>>**. The **<<Inherit>>** stereotype was created during the development of the GFT and is based on the SysML Generalization line to facilitate the "source" inheriting the attributes of the "target". SysML defines Generalization as *a taxonomic relationship between a more general classifier and a more specific classifier. Each instance of the specific classifier is also an indirect instance of the general classifier. Thus, the specific classifier inherits the features of the more general classifier*[iv]. The term "Generalization" implies a mechanism for combining similar classes of objects into a single, more general class. "Inheritance" is a more limited concept, referring to the mechanism that permits subclasses to share attributes with superclasses.[v] Currently the SysML language combines these two ideas into one type of connection; for GFT purposes it was necessary to separate and specifically use the "Inheritance" concept. This allows for passing of the State Variable attribute from one State Variable/Vector to another. The connection is created to go FROM the vector needing the variable attributes TO the Vector, or Variable, that has the variable attributes.

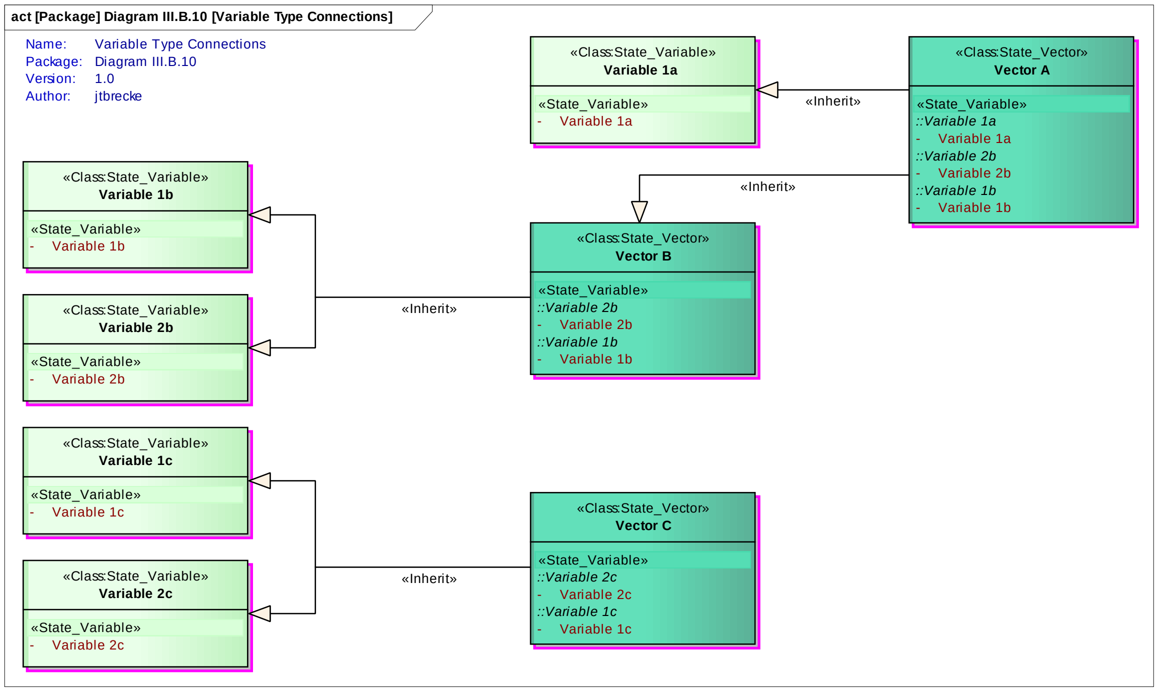

act [Package] Diagram III.B.10 [Variable Type Connections]
Name: Variable Type Connections
Package: Diagram III.B.10
Version: 1.0
Author: jtbrecke
«Class:State_Variable» Variable 1a
«State_Variable» - Variable 1a
«Class:State_Vector» Vector A
«State_Variable» ::Variable 1a - Variable 1a ::Variable 2b - Variable 2b ::Variable 1b - Variable 1b
«Inherit»
«Class:State_Variable» Variable 1b
«State_Variable» - Variable 1b
«Class:State_Vector» Vector B
«State_Variable» ::Variable 2b - Variable 2b ::Variable 1b - Variable 1b
«Class:State_Variable» Variable 2b
«State_Variable» - Variable 2b
«Class:State_Variable» Variable 1c
«State_Variable» - Variable 1c
«Class:State_Vector» Vector C
«State_Variable» ::Variable 2c - Variable 2c ::Variable 1c - Variable 1c
«Class:State_Variable» Variable 2c
«State_Variable» - Variable 2c

Figure 17: State Variable/Vector Model Library Construct Connections

IV. Conclusions

In conclusion, the GFT gives systems engineers a representation and process to define a system functionally in a top down perspective that links the physical design to the functional design. It also allows for incorporation of nominal and off nominal design during the functional expression of the system. The GFT, by linking the goals of a system both to the functions and to the physical attributes of the system that must be controlled, also establishes a mathematical and logical structure for creating requirements for the system that allows for systematic verification and validation. Although the concepts described within this paper are applicable to SysML, with the current version of the language it becomes difficult to reliably perform the steps described in a repeatable manner. This is due to SysML's heavy reliance on the physical components of a system and less so with the functional components and system requirements. Although SysML was the best available language to start with in creating the concepts that make up the GFT, it is not ideal because it is not optimized to model the inherent tree structure required by GFT

[iv] The OMG UML specification (*UML Superstructure Specification*, *v2.1.1*, *p. 73*)
[v] "Generalization and Inheritance, Concepts of Generalization and Inheritance" *Universal Teacher Publications*, [Online Article],http://www.universalteacherpublications.com/univ/free-asgn/2008/mcs32/page1.htm [cited 8 July 2013]

implementation. By using SysML there is an additional layer of complexity involved in keeping track of the tree due to the inherency of GFT branch crossing. Put another way, either the GFT replicates the same tree branch in several locations, or it creates several connections from those several locations to the tree branch. An example of this is that to maintain structural integrity of a launch vehicle one must maintain control, but it is also true that to maintain control one must maintain structural integrity. Additional work would be needed to (a) create a tool that would inherently utilize the concepts described above, or (b) utilize the particular aspects of SysML that do work and expand on the language to incorporate GFT concepts into the core SysML architecture.

Although SysML is not the perfect tool to use to create and represent a GFT, it was the tool available during the development of the concepts presented in this paper. SysML is also an emerging standard for systems engineering modeling, so it was worthwhile to determine if SysML's capabilities are sufficient to represent new systems engineering methods such as the GFT. To properly define all of the stereotypes that were needed, an Enterprise Architecture Toolbox was created on top of the SysML standard toolboxes. This tool box allowed for the correct constructs and construct connections to be used in the development of the GFT. Figure 18 shows the graphical representation of the Toolbox created for use in development of a GFT.

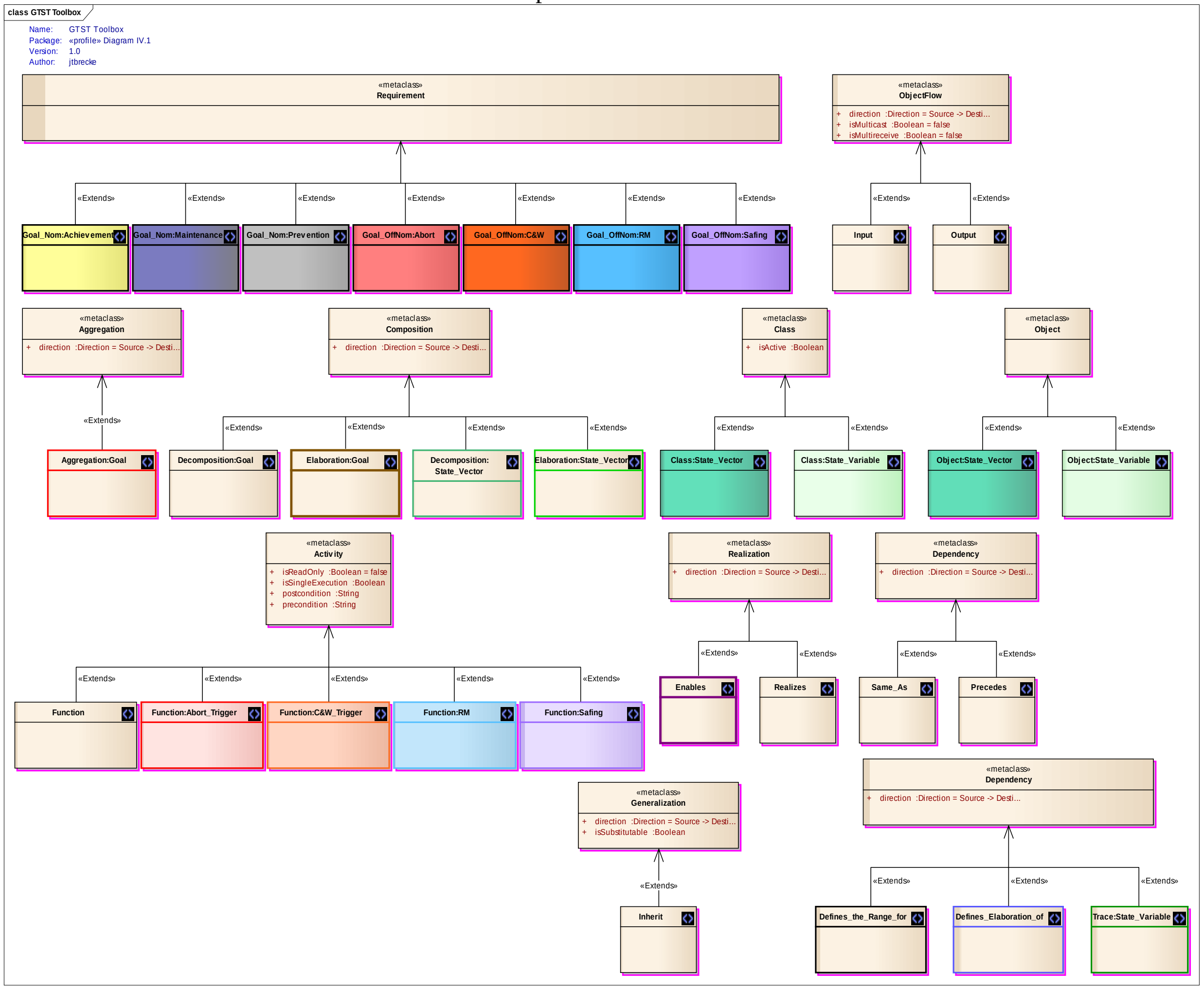


Figure 18: GFT Toolbox Profile

The profile diagram shown in Figure 18 gives the modeler the ability to use all of the required new stereotypes developed for the GFT. Details of each are listed in Table 1.

Table 1: GFT Stereotype List

GFT Stereotype	**SysML Metaclass**	**Description**
<<Goal_Nom:Achievemnt>>	Requirement	Nominal Goals used within the GFT
<<Goal_Nom:Maintenance>>	Requirement	
<<Goal_Nom:Precention>>	Requirement	
<<Goal_OffNom:Abort>>	Requirement	Off Nominal Goals used within the GFT

<<Goal_OffNom:C&W>>	Requirement	
<<Goal_OffNom:RM>>	Requirement	
<<Goal_OffNom:Safing>>	Requirement	
<<Input>>	ObjectFlow	Object Flows used to show how and what State Variables/Vectors are used by Nominal and Off Nominal Functions
<<Output>>	ObjectFlow	
<<Aggregation:Goal>>	Aggregation	Used to connect Off Nominal Goals to Nominal Goals
<<Decomposition:Goal>>	Composition	Used to connect Nominal Goals to Nominal Goals
<<Elaboration:Goal>>	Composition	
<<Decomposition:State_Vector>>	Composition	Used to connect State Vectors to State Vectors consistently with Goal Diagrams
<<Elaboration:State_Vector>>	Composition	
<<Class:State_Vector>>	Class	Used to create "Types" of State Variables/Vectors to be specifically classified elsewhere in the model
<<Class:State_Variable>>	Class	
<<Object:State_Vector>>	Object	Used to create specific State Variables/Vectors out of State Variable/Vector "Types"
<<Object:State_Variable>>	Object	
<<Function>>	Activity	Nominal and Off Nominal Functions used within the GFT
<<Function:Abort_Trigger>>	Activity	
<<Function:C&W_Trigger>>	Activity	
<<Function:RM>>	Activity	
<<Function:Safing>>	Activity	
<<Realizes>>	Realization	Used to link Functions to Goals
<<Inherit>>	Generalization	Used to pass State Variables to Vectors and lower level State Vectors to higher level State Vectors
<<Defines_the_Range_for>>	Dependency	Connects State Variables/Vectors to Goals
<<Defines_Elaboration_of>>	Dependency	Connects Function Packages to Goals in the Goal Diagram

Implementation of a Goal-Based Systems Engineering Process Using the Systems Modeling Language (SysML)

Jonathan T. Breckenridge
jonathan.t.breckenridge@nasa.gov

NASA Marshall Space Flight Center
EV43 Integrated System Health Management and Automation Branch

Miltec, A Ducommun Company and Jacobs ESSSA

19 August 2013
AIAA Infotech@Aerospace
Boston, Massachusetts

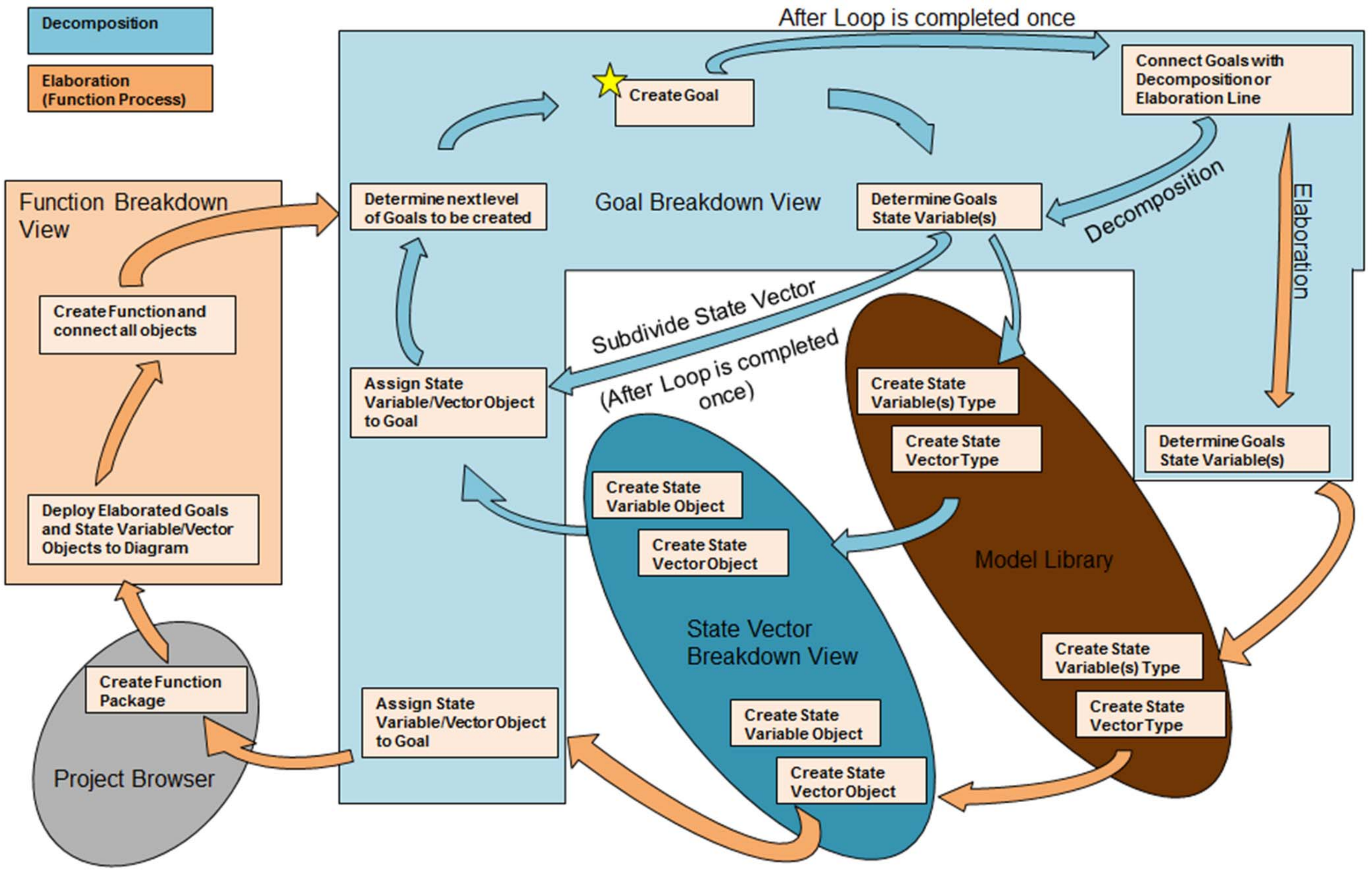
What is a Goal Function Tree (GFT)?

- **A top-down hierarchical decomposition of system goals and functions --- hierarchies are models of "intention"**
  - Arranged by major system phase/configuration,
  - Defines the functions the system must perform and goals the system must achieve for the system to successfully perform its mission/objectives
  - Relationship between goals and functions defined through rigorous use of state variables

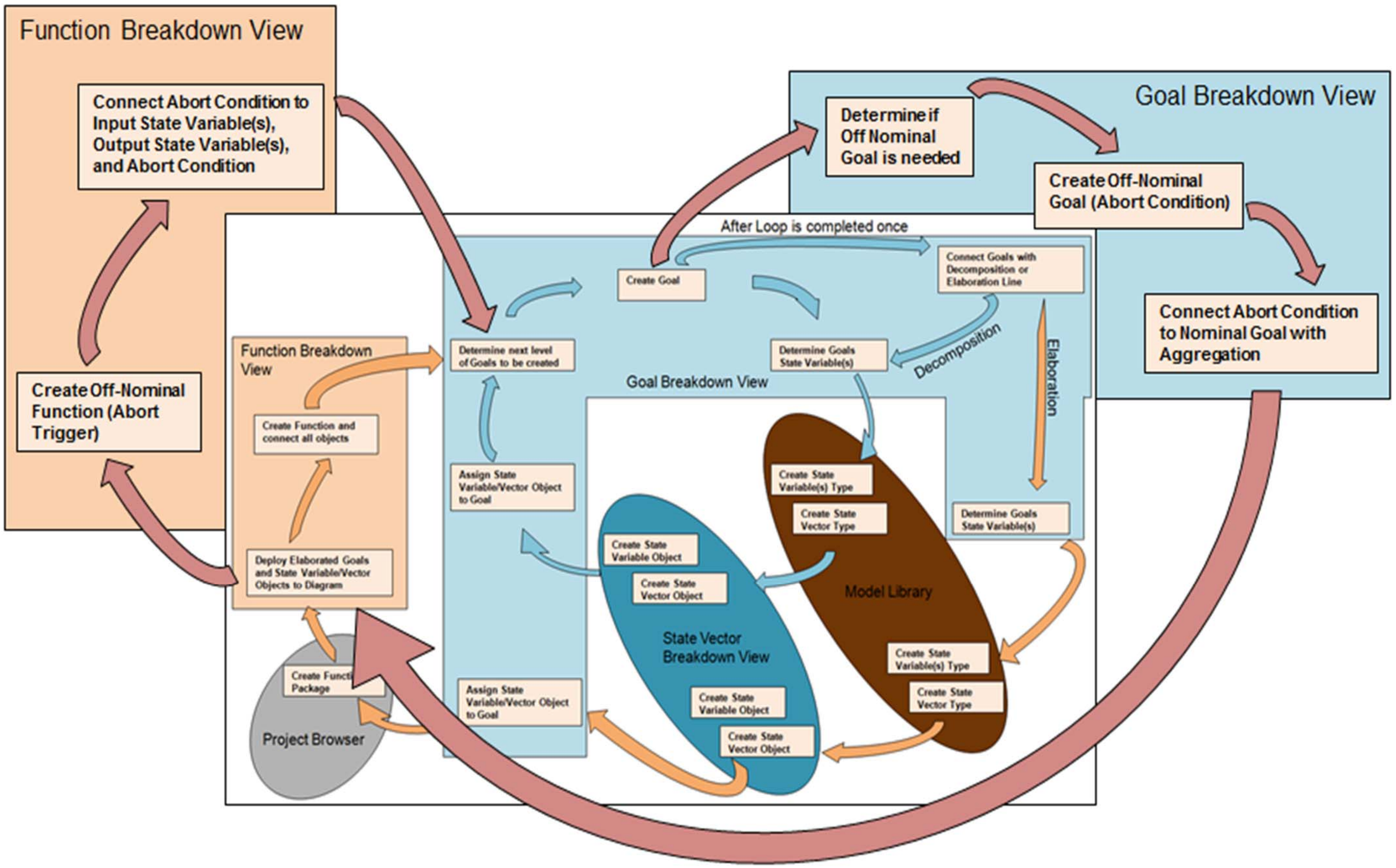
- **The GFT extends the classical function decomposition through the use of state variables to explicitly contain goals, and makes the GFT causally, physically correct**

- **It is based on function, not design, though inherently you *must* make some design assumptions**
  - **Example: Goal is to place humans into trans-lunar trajectory**
  - **Two possible solutions:**
    - **Star Trek-style quantum transporter**
    - **Space Launch System-style chemical rocket**
  - **These two solutions yield very different function decompositions, SLS has stages, the transporter does not, etc.**

Nominal GFT Development Process


Decomposition
Elaboration (Function Process)
After Loop is completed once
Create Goal
Connect Goals with Decomposition or Elaboration Line
Function Breakdown View
Determine next level of Goals to be created
Goal Breakdown View
Determine Goals State Variable(s)
Decomposition
Elaboration
Create Function and connect all objects
Subdivide State Vector
(After Loop is completed once)
Assign State Variable/Vector Object to Goal
Create State Variable(s) Type
Create State Vector Type
Determine Goals State Variable(s)
Create State Variable Object
Create State Vector Object
Deploy Elaborated Goals and State Variable/Vector Objects to Diagram
Model Library
State Vector Breakdown View
Create State Variable(s) Type
Create Function Package
Assign State Variable/Vector Object to Goal
Create State Variable Object
Create State Vector Type
Create State Vector Object
Project Browser

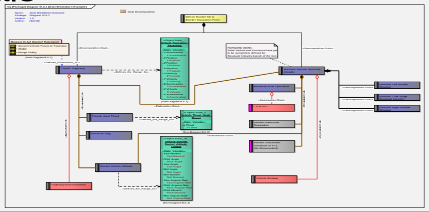
Off-Nominal GFT Development Process


Function Breakdown View
Connect Abort Condition to Input State Variable(s), Output State Variable(s), and Abort Condition
Create Off-Nominal Function (Abort Trigger)
Goal Breakdown View
Determine if Off Nominal Goal is needed
Create Off-Nominal Goal (Abort Condition)
Connect Abort Condition to Nominal Goal with Aggregation
After Loop is completed once
Create Goal
Connect Goals with Decomposition or Elaboration Line
Determine next level of Goals to be created
Determine Goals State Variable(s)
Decomposition
Elaboration
Goal Breakdown View
Function Breakdown View
Create Function and connect all objects
Assign State Variable/Vector Object to Goal
Determine Goals State Variable(s)
Deploy Elaborated Goals and State Variable/Vector Objects to Diagram
Create State Variable(s) Type
Create State Vector Type
Model Library
Create State Variable Object
Create State Vector Object
State Vector Breakdown View
Create State Variable Object
Create State Vector Object
Create State Variable(s) Type
Create State Vector Type
Create Function Package
Project Browser
Assign State Variable/Vector Object to Goal

GFT Modeling Views

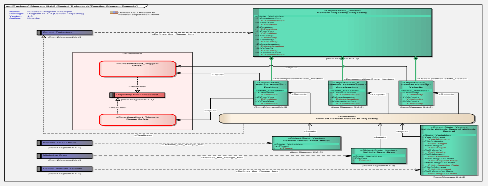
The Goal Breakdown View
SysML Requirements Diagram

- Goal Diagrams
  - Goal to Goal Connections
  - Goal to State Variable Connections
  - Goal to Function Package Connections
- Subdivided Packages
  - Top Level Goals
  - Mission Success Goals
  - Crew Safety Goals
  - Abort Goals



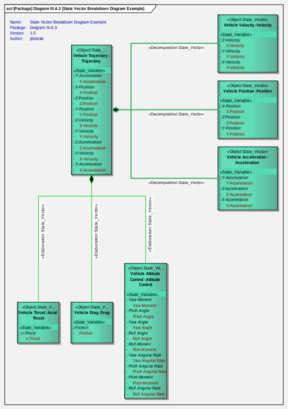
The Functional Breakdown View
SysML Activity Diagram

- Function Diagram
  - Goal to State Variable Connections
  - State Variable to Functions Connections
- Subdivided Packages
  - One Package per Function/Diagram
  - Functions grouped by Mission Phase



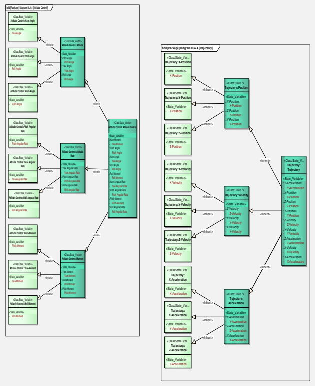
The State Vector Breakdown View
SysML Activity Diagram

- State Variable/Vector Object Diagram
  - State Variable/Vector to Variable/Vector Connections
  - Consistent with Goal and Function Diagrams
- Subdivided Packages
  - One Package per Operational Phase
  - Multiple Sub-Packages for each Sub-System Component
    - i.e. MPS, TVC, GN&C



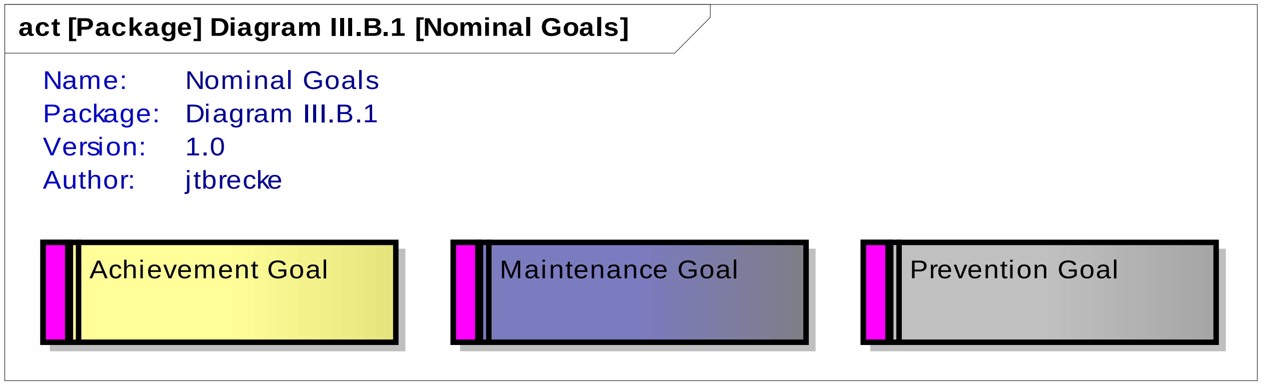
The Model Library
SysML Class Diagram

- State Variable/Vector "Type" Diagram
  - Variable to Vector Roll-up Connections
  - Vector to Vector Roll-up Connections
- Subdivided Packages
  - Multiple Diagrams per Package
  - Packages are grouped by Variable/Vectors
    - i.e. Propellant, Structural, Thrust, Trajectory



GFT Nominal System Goals

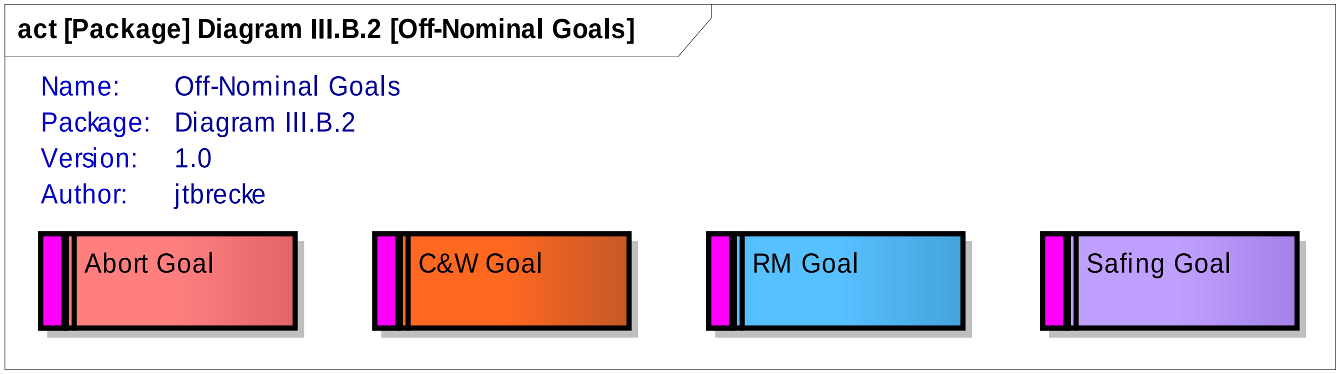
- **All Goals in the GFT Model are SysML Requirement constructs with specific Stereotypes**
  - The color shown is part of the Stereotype definition
- **In the GFT Model there are 2 types of goals used in the GFT process**
  - Nominal Goals
  - Off Nominal Goals
- **Nominal Goal Definition**
  - Achievement Goal ***<<Goal_Nom:Achievement>>***
    {Stereotyped SysML Requirement}
    - An Achievement Goal is a proposition that must be true in the final state of the goal.
      - They are used to indicate the end of a phase or a set of phases.
  - Maintenance Goal ***<<Goal_Nom:Maintenance>>***
    {Stereotyped SysML Requirement}
    - A Maintenance Goal is a proposition that must be true in every state through which the agent passes.
      - These are often Safety Goals – the goal of staying way from off nominal states.
      - One or more Maintenance Goals together make up a phase that culminates in an Achievement Goal.
  - Prevention Goal ***<<Goal_Nom:Prevention>>***
    {Stereotyped SysML Requirement}
    - A Prevention Goal is a proposition where the system must inhibit an event or action.
      - These are used in association with Achievement Goals, and indicates the goal of "Not performing something" .



Ducommun Miltec
JACOBS ESSSA Group

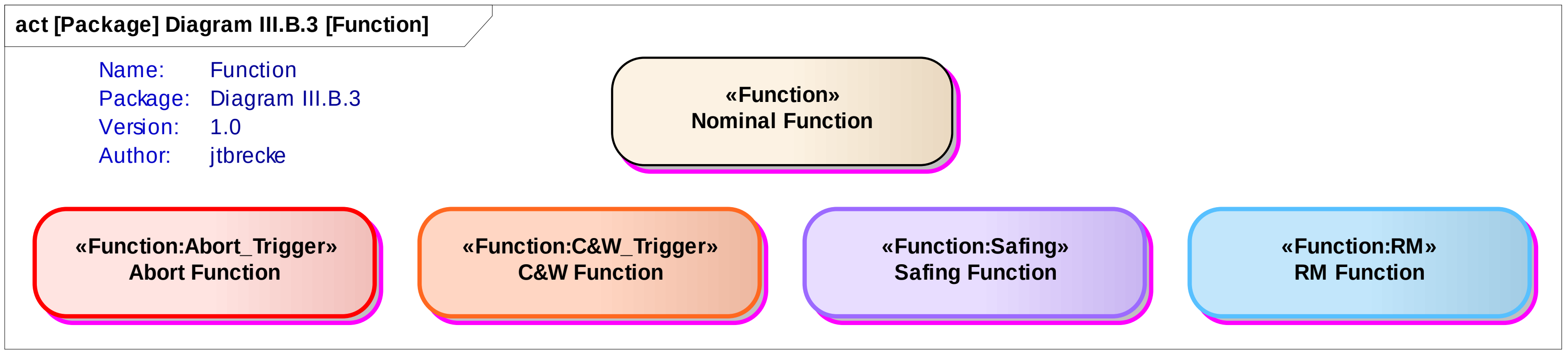
GFT Off-Nominal System Goals

- **Abort Goal <<*Goal_OffNom:Abort*>>**
  {Stereotyped SysML Requirement}
  - An Abort Goal is a proposition to notify the system for LOM conditions
    - It ultimately leads to the creation of a function that performs the task of monitoring for the particular issues declared in the Abort Goal
    - Specifically an Abort Goal will indicate a position in the tree where a goal change will be needed to prevent a LOC Situation
- **Caution & Warning (C&W) Goal <<*Goal_OffNom:C&W*>>**
  {Stereotyped SysML Requirement}
  - A C&W Goal is a proposition to notify the system for warning alerts to crew and/or mission support personnel.
    - It ultimately leads to the creation of a function that performs the task of monitoring for the particular issues declared in the C&W Goal.
    - Specifically a C&W Goal will indicate a position in the tree where there is a degraded function and a notification could/should be made.
- **Redundancy Management (RM) Goal <<*Goal_OffNom:RM*>>**
  {Stereotyped SysML Requirement}
  - An RM Goal is a proposition to provide either passive or active management of system capabilities in order to maintain overall functionality.
    - Specifically an RM Goal will indicate where redundant systems will respond to a failed or degraded state of the system
- **Safing Goal <<*Goal_OffNom:Safing*>>**
  {Stereotyped SysML Requirement}
  - A Safing Goal is a proposition that, if a critical failure occurs, will change the state of the system into a "Safe State" so that the higher critical goals can still be achieved with no further damage to the system.



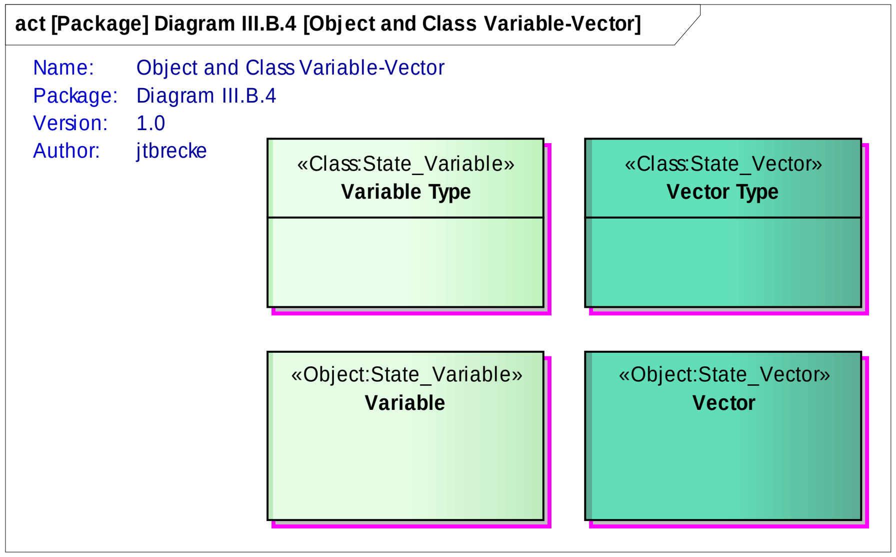
GFT Nominal and Off-Nominal Functions

- **In the GFT Model SysML activities constructs are used to define functions, there are 2 types used in the GFT process**
  - **Nominal** Functions are non-stereotyped SysML activity constructs
  - **Off-Nominal** Functions are stereotyped SysML activity constructs as shown below
- **Nominal Function**
  - A function is an activity that transforms one or more State Variables into a different set of State Variables. All functions are realizations of their associated goals and cannot exist independent of a goal.
- **Off-Nominal Functions**
  - A function is an activity that transforms one or more State Variables into a different set of State Variables
  - All functions are realizations of their associated goals and cannot exist independent of a goal.
  - Off-Nominal Function Types
    - Abort Functions (Triggers) ***<<Function:Abort_Triggers>>***
      {Stereotyped SysML Requirement}
    - C&W Functions (Triggers) ***<<Function:C&W_Triggers>>***
      {Stereotyped SysML Requirement}
    - RM Functions ***<<Function:RM>>***
      {Stereotyped SysML Requirement}
    - Safing Functions ***<<Function:Safing>>***
      {Stereotyped SysML Requirement}


act [Package] Diagram III.B.3 [Function]
Name: Function
Package: Diagram III.B.3
Version: 1.0
Author: jtbrecke
«Function» Nominal Function
«Function:Abort_Trigger» Abort Function
«Function:C&W_Trigger» C&W Function
«Function:Safing» Safing Function
«Function:RM» RM Function

GFT State Variable/Vectors: Objects and Classes

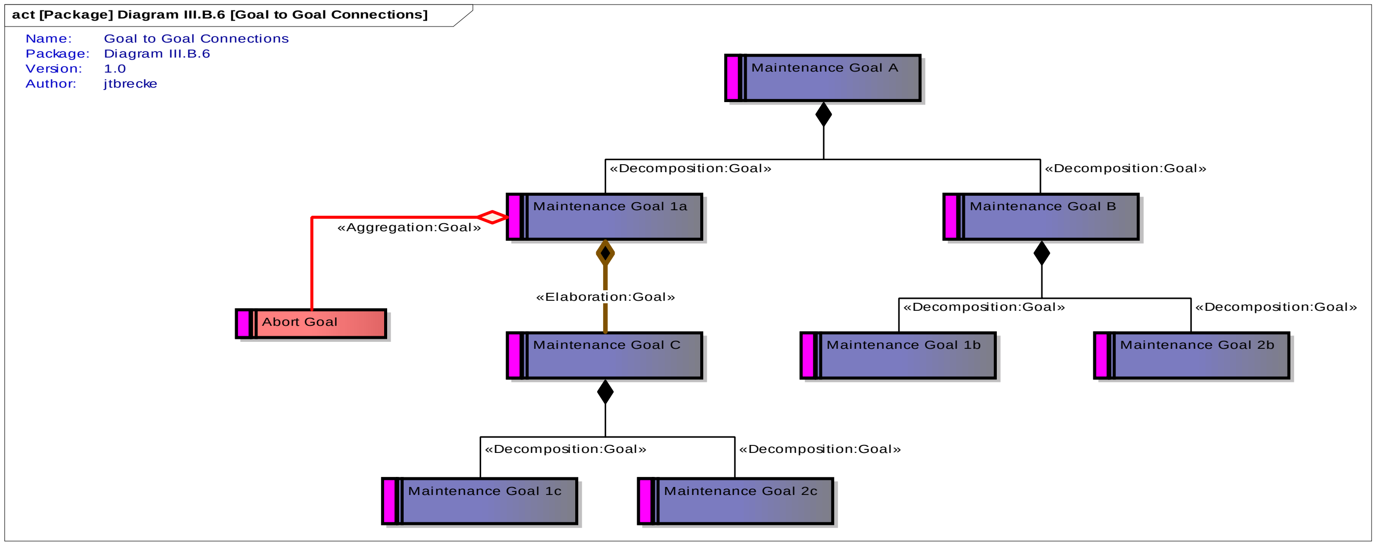
- **State Variables**
  - A State Variable is a physical attribute of a system that must be maintained within an appropriate range for the success of the assigned goal.
- **In the GFT Model, SysML Object and Class constructs are used to define state variables for which there are 2 types used in the GFT process**
  - ***<<Object:State_Variable>>***
    {SysML *Object*}
    - An Object State Variable is a stereotype used when calling out a specific instance of a state variable.
  - ***<<Class:State_Variable>>***
    {SysML *Class*}
    - A Class State Variable is a stereotype used when creating a type of variable to be used in several places in the model
- **State Vector**
  - State Vectors are used with both the *Object* and *Class* Stereotype and are a set of State Variables. State Variables are "attributes" associated with the State Vector.


act [Package] Diagram III.B.4 [Object and Class Variable-Vector]
Name: Object and Class Variable-Vector
Package: Diagram III.B.4
Version: 1.0
Author: jtbrecke
«Class:State_Variable» Variable Type
«Class:State_Vector» Vector Type
«Object:State_Variable» Variable
«Object:State_Vector» Vector

Ducommun Miltec
JACOBS ESSSA Group

GFT Goal Diagram Construct Connections

- **Goal to Goal Connection**
  - Goal Decomposition ***<<Decomposition:Goal>>***
    {Standard SysML Decomposition Line}
    - A goal decomposition is a segmentation into a complete set of sub goals necessary to achieve the higher level goal.
    - To determine when a goal decomposition is needed the State Vector must be evaluated. If the State Variables within a State Vector can be partitioned into subset State Vectors without introducing any new State Variable(s) then a goal decomposition is appropriate. If a new State Variable, or set of State Variables, are needed then a Functional Elaboration is needed.
  - Goal Elaboration ***<<Elaboration:Goal>>***
    {Stereotyped SysML Decomposition Line}
    - A goal elaboration is a transformation of the state vector and indicates a function or set of functions are needed to continue development of the tree along the particular path.
    - A goal elaboration is required when new State Variables must be introduced to achieve the higher level goal. A function is added to transform the new State Variable(s) into the higher level existing State Variable(s).
  - Goal Aggregation ***<<Aggregation:Goal>>***
    {Stereotyped SysML Aggregation Line}
    - 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 goals within the lower level goals.



GFT Goal Diagram Construct Connections

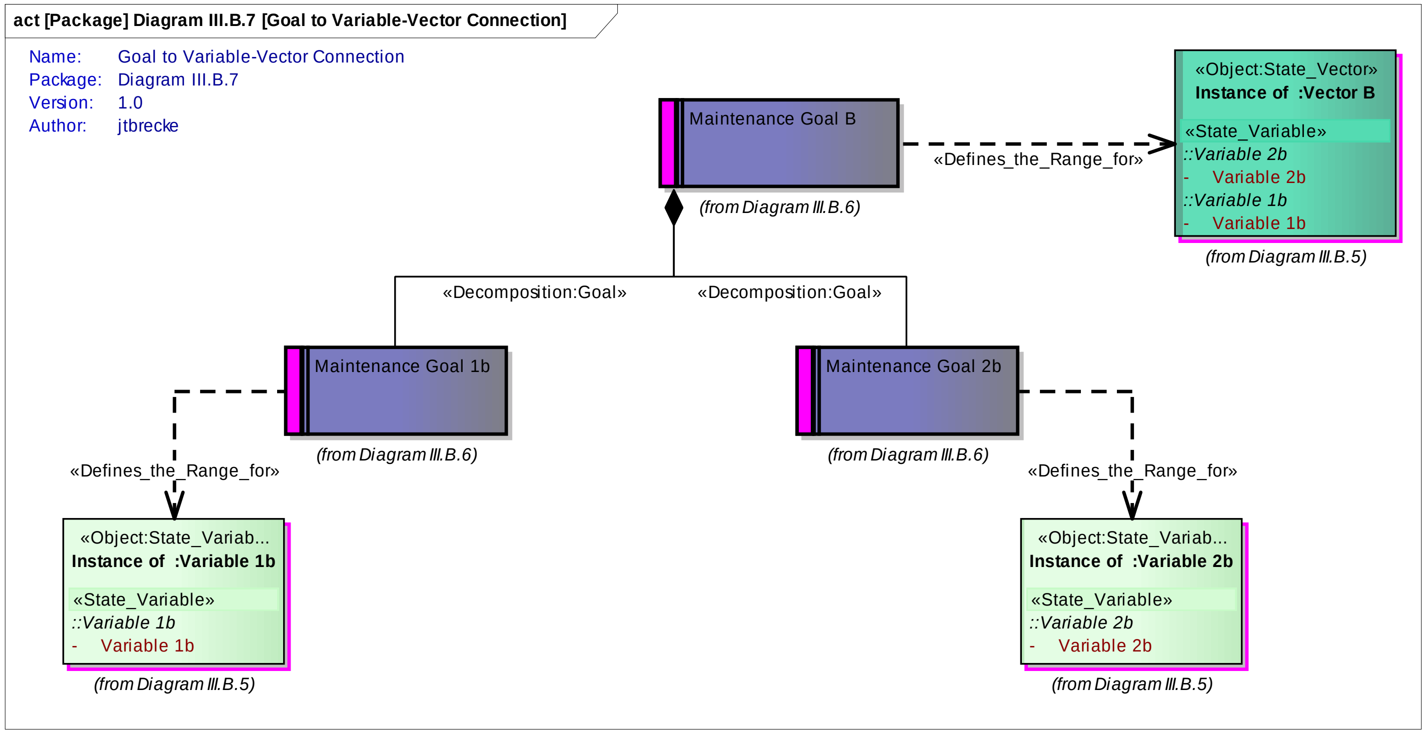
♦ Goal to State Variable/Vector Connections

- ***<<Defines_the_Range_for>>***

  {Stereotyped SysML Dependency Line}

  – Goals define the range of a State Variable or a set of State Variables, a State Vector

  – Only the specific Object version of a State Variable/Vector is used on a Goal Diagram



GFT Goal Diagram Construct Connections

♦ Goal to Function Package Connections

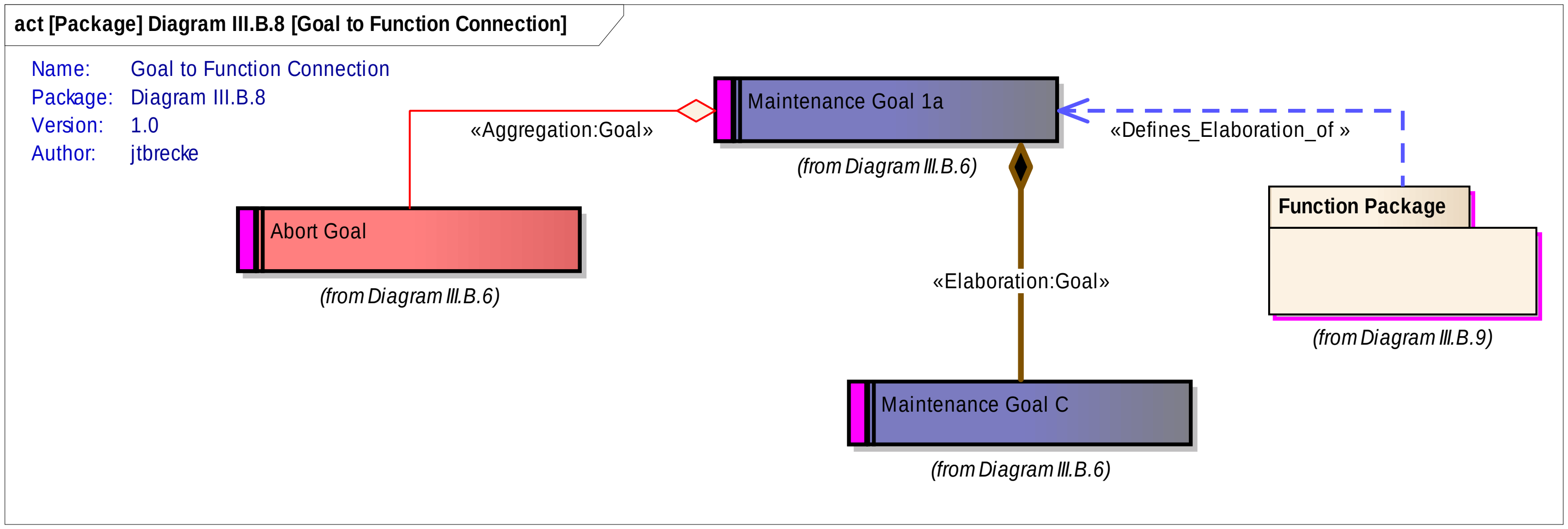
- ***<<Defines_Elaboration_of>>***

  {Stereotyped SysML Dependency Line}

  – When the Goal Elaboration ***<<Elaboration:Goal>>*** is used, and new State Variable/Vector(s) must be created, a package must be created to explicitly define that transformation

  – The link connects the Goal being Elaborated to a Function Package

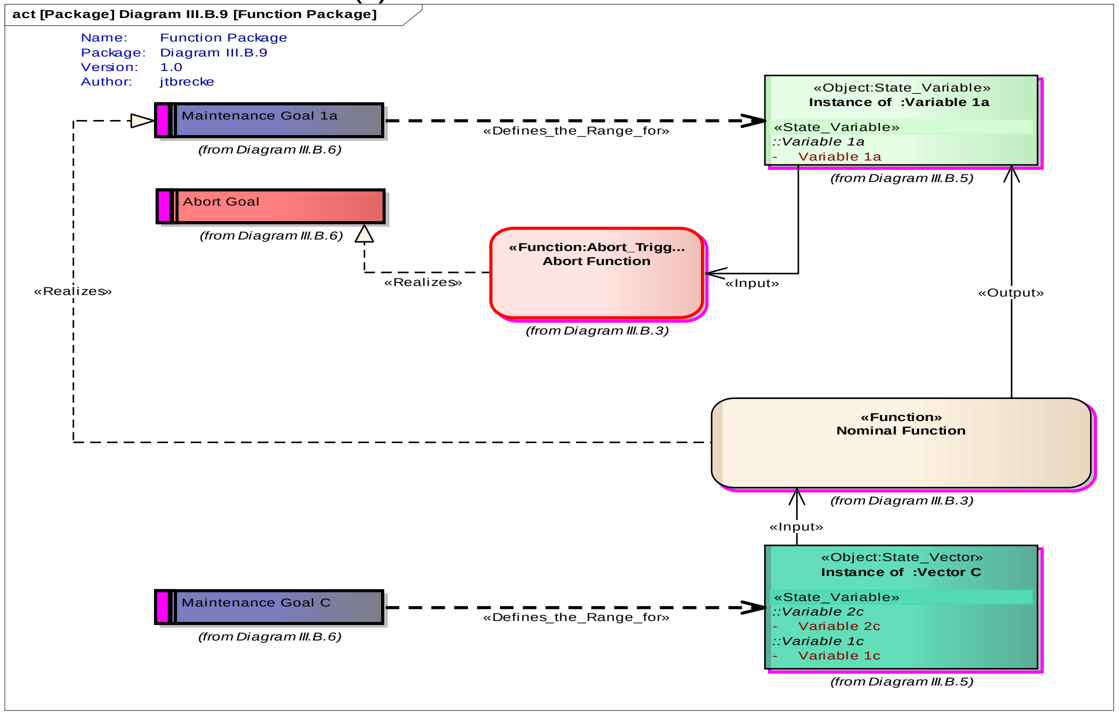
    - The activity diagram that is created alongside the package is where the description of the transformation process is shown, this is also known as the Function Diagram



Ducommun Miltec JACOBS ESSSA Group

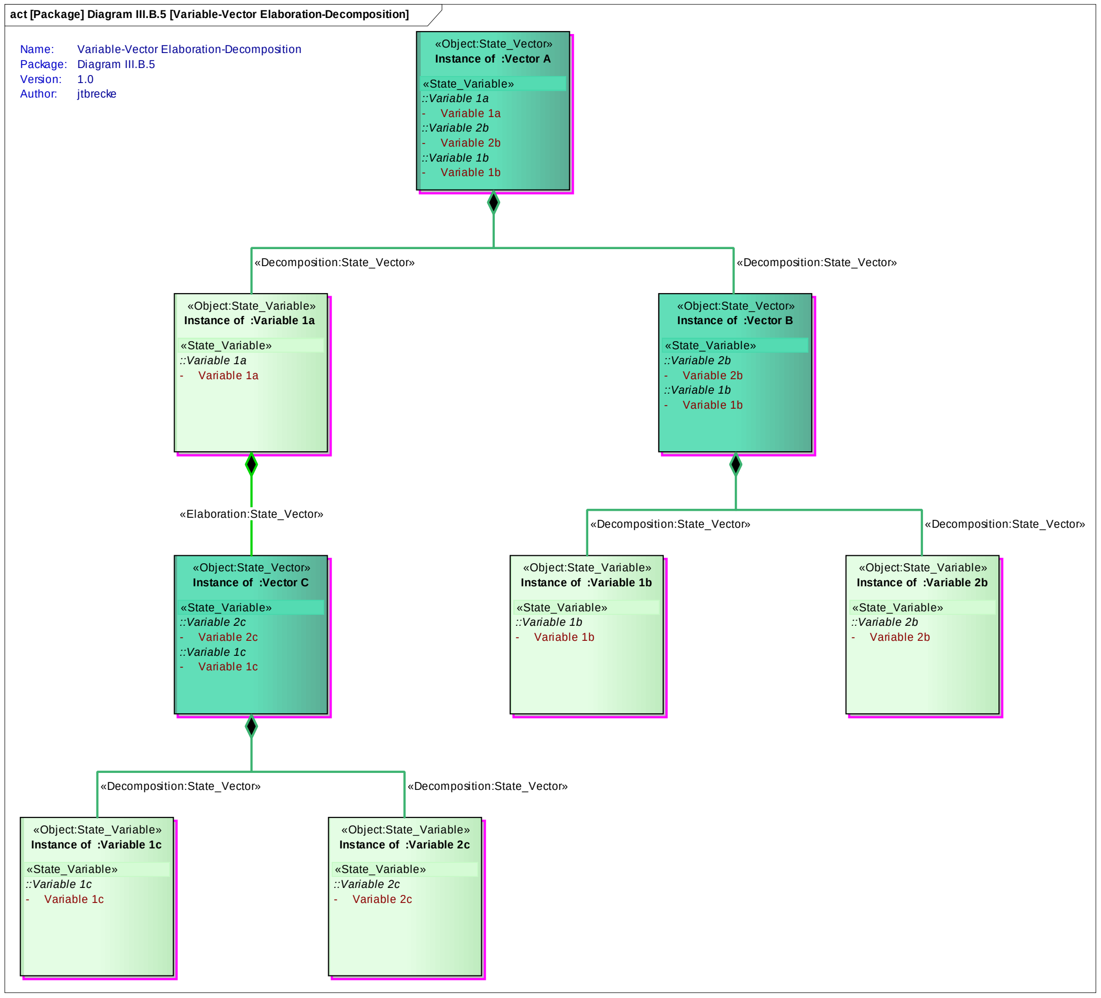
GFT Function Diagram Construct Connections

- **Function Diagrams show the connection between ‘y’ and ‘x’ to ‘f’ in the equation y=f(x)**
  - <<Input>> *{Standard SysML Object Flow Line}*
    - Construct Connections linking the input State Variable ‘x’ to the Function ‘f’
  - <<Output>>*{Standard SysML Object Flow Line}*
    - Construct Connection linking the Function ‘f’ to the output State Variable ‘y’
- **Functions can not exist with out a goal**
  - <<Realizes>> {Standard SysML Realization Line}
    - Construct Connection linking a nominal or Off-Nominal Function to show that there exists a function that gives the goal its State Variable/Vector(s)



GFT State Vector Object Construct Connections

- **In concert with Goal Diagram Elaboration and Decomposition Construct Connections, links connect State Variable and Vector objects within the State Vector Breakdown View of the model.**
  - In addition to the State Vector Breakdown View being the library for State Vector Objects used on Goal Diagrams it also shows the connections of the State Vectors to each other
  - State Vector Decomposition **<<Decomposition:StateVector>>**
    - A State Vector Decomposition shows the partitioning of one state vector into its sub state vectors
    - If taken to the lowest level possible a state vector can be eventually decomposed to its individual state variables
  - State Vector Elaboration ***<<Elaboration:StateVector>>***
    - A State Vector Elaboration shows the connection between two state vectors that have been elaborated by a function
    - A State Vector Elaboration is used when one or more new State Variables are introduced that did not exist in the higher level State Vector

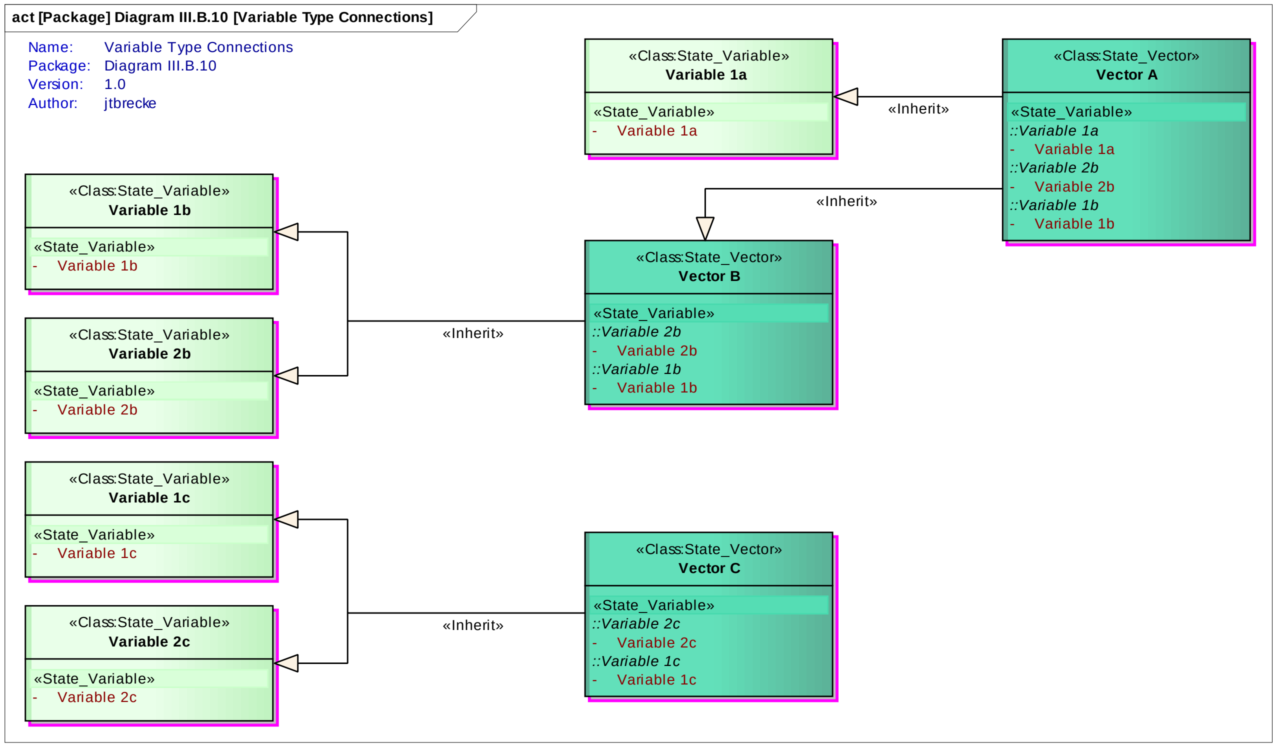


act [Package] Diagram III.B.5 [Variable-Vector Elaboration-Decomposition]
Name: Variable-Vector Elaboration-Decomposition
Package: Diagram III.B.5
Version: 1.0
Author: jtbrecke
«Object:State_Vector» Instance of :Vector A
«State_Variable» ::Variable 1a - Variable 1a ::Variable 2b - Variable 2b ::Variable 1b - Variable 1b
«Decomposition:State_Vector»
«Decomposition:State_Vector»
«Object:State_Variable» Instance of :Variable 1a
«State_Variable» ::Variable 1a - Variable 1a
«Object:State_Vector» Instance of :Vector B
«State_Variable» ::Variable 2b - Variable 2b ::Variable 1b - Variable 1b
«Elaboration:State_Vector»
«Decomposition:State_Vector»
«Decomposition:State_Vector»
«Object:State_Vector» Instance of :Vector C
«State_Variable» ::Variable 2c - Variable 2c ::Variable 1c - Variable 1c
«Object:State_Variable» Instance of :Variable 1b
«State_Variable» ::Variable 1b - Variable 1b
«Object:State_Variable» Instance of :Variable 2b
«State_Variable» ::Variable 2b - Variable 2b
«Decomposition:State_Vector»
«Decomposition:State_Vector»
«Object:State_Variable» Instance of :Variable 1c
«State_Variable» ::Variable 1c - Variable 1c
«Object:State_Variable» Instance of :Variable 2c
«State_Variable» ::Variable 2c - Variable 2c

GFT State Vector Type Construct Connections

- **A set of State Variable/Vector Types are created before the State Variable/Vector Objects can be created,**
  - 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
- **The State Vector Types are contained within the State Vector Library**
  - The ***<<Class:State_Variable>>*** are created, and an attribute assigned representing the variable itself.
  - ***<<Inherit>>*** is used once all of the variables are created

    {Stereotyped SysML Generalization line}
- **The connection is created going from the vector, *<<Class:State_Vector>>*, to the variables**


act [Package] Diagram III.B.10 [Variable Type Connections]
Name: Variable Type Connections
Package: Diagram III.B.10
Version: 1.0
Author: jtbrecke
«Class:State_Variable» Variable 1a
«State_Variable» - Variable 1a
«Class:State_Vector» Vector A
«State_Variable» ::Variable 1a - Variable 1a ::Variable 2b - Variable 2b ::Variable 1b - Variable 1b
«Inherit»
«Class:State_Variable» Variable 1b
«State_Variable» - Variable 1b
«Class:State_Variable» Variable 2b
«State_Variable» - Variable 2b
«Class:State_Vector» Vector B
«State_Variable» ::Variable 2b - Variable 2b ::Variable 1b - Variable 1b
«Class:State_Variable» Variable 1c
«State_Variable» - Variable 1c
«Class:State_Variable» Variable 2c
«State_Variable» - Variable 2c
«Class:State_Vector» Vector C
«State_Variable» ::Variable 2c - Variable 2c ::Variable 1c - Variable 1c

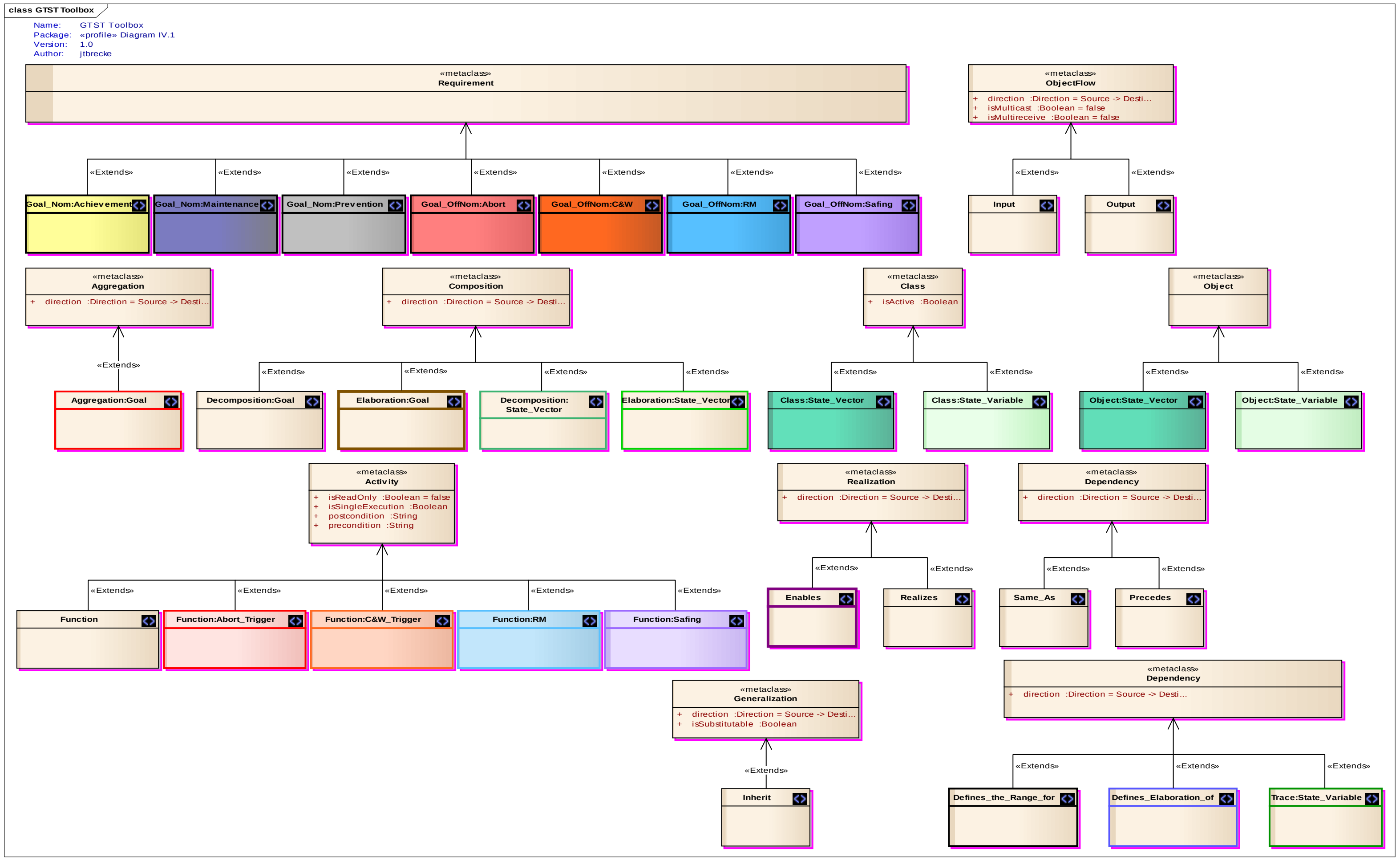
GFT Conclusions

- **The GFT gives systems engineers a representation and process to define a system functionally in a top down perspective that links the physical design to the functional design.**
  - It allows for incorporation of nominal and off nominal design during the functional expression of the system.
  - It establishes a mathematical and logical structure for creating requirements for the system that allows for systematic verification and validation
- **However, with the current version of the SysML language it becomes difficult to reliably perform the steps described in a repeatable manner.**
  - SysML relies heavy on the physical components of a system and less so with the functional components and system requirements.
  - SysML is not optimized to model the inherent tree structure required by GFT implementation.
  - Due to the inherency of cross branching in a GFT model, SysML adds an additional layer of complexity.
    - An example of this is that to maintain structural integrity of a launch vehicle one must maintain control, but it is also true that to maintain control one must maintain structural integrity.
- **Additional work would be needed to:**
  - Create a tool that would inherently utilize the concepts described above, or;
  - Utilize the particular aspects of SysML that do work and expand on the language to incorporate GFT concepts into the core SysML architecture

BACKUP

GFT Toolbox

- **Although SysML is not the perfect tool to use to create and represent a GFT, it was the tool available during the development of the concepts presented**
  - To properly define all of the stereotypes that were needed, an Enterprise Architecture Toolbox was created on top of the SysML standard toolboxes



Ducommun Miltec

JACOBS ESSSA Group

GFT Stereotype List

GFT Stereotype	SysML Metaclass	Description
<<Goal_Nom:Achievemnt>>	Requirement	Nominal Goals used within the GFT
<<Goal_Nom:Maintenance>>	Requirement	
<<Goal_Nom:Precention>>	Requirement	
<<Goal_OffNom:Abort>>	Requirement	Off Nominal Goals used within the GFT
<<Goal_OffNom:C&W>>	Requirement	
<<Goal_OffNom:RM>>	Requirement	
<<Goal_OffNom:Safing>>	Requirement	
<<Input>>	ObjectFlow	Object Flows used to show how and what State Variables/Vectors are used by Nominal and Off Nominal Functions
<<Output>>	ObjectFlow	
<<Aggregation:Goal>>	Aggregation	Used to connect Off Nominal Goals to Nominal Goals
<<Decomposition:Goal>>	Composition	Used to connect Nominal Goals to Nominal Goals
<<Elaboration:Goal>>	Composition	
<<Decomposition:State_Vector>>	Composition	Used to connect State Vectors to State Vectors consistently with Goal Diagrams
<<Elaboration:State_Vector>>	Composition	
<<Class:State_Vector>>	Class	Used to create “Types” of State Variables/Vectors to be specifically classified elsewhere in the model
<<Class:State_Variable>>	Class	
<<Object:State_Vector>>	Object	Used to create specific State Variables/Vectors out of State Variable/Vector “Types”
<<Object:State_Variable>>	Object	
<<Function>>	Activity	Nominal and Off Nominal Functions used within the GFT
<<Function:Abort_Trigger>>	Activity	
<<Function:C&W_Trigger>>	Activity	
<<Function:RM>>	Activity	
<<Function:Safing>>	Activity	
<<Realizes>>	Realization	Used to link Functions to Goals
<<Inherit>>	Generalization	Used to pass State Variables to Vectors and lower level State Vectors to higher level State Vectors
<<Defines_the_Range_for>>	Dependency	Connects State Variables/Vectors to Goals
<<Defines_Elaboration_of>>	Dependency	Connects Function Packages to Goals in the Goal Diagram