A Generic Modeling Process to Support Functional Fault Model Development

William A. Maul[1]
Vantage Partners, LLC, Brook Park, Ohio, 44142, USA

Joseph A Hemminger[2]
Zin Technologies, Inc., Cleveland, Ohio, 44130, USA

Rebecca Oostdyk[3]
SGT, Inc., Kennedy Space Center, FL 32899, USA

and

Rachael A. Bis[4]
N&R Engineering, Cleveland, OH, 44130, USA

Functional fault models (FFMs) are qualitative representations of a system's failure space that are used to provide a diagnostic of the modeled system. An FFM simulates the failure effect propagation paths within a system between failure modes and observation points. These models contain a significant amount of information about the system including the design, operation and off nominal behavior. The development and verification of the models can be costly in both time and resources. In addition, models depicting similar components can be distinct, both in appearance and function, when created individually, because there are numerous ways of representing the failure space within each component. Generic application of FFMs has the advantages of software code reuse: reduction of time and resources in both development and verification, and a standard set of component models from which future system models can be generated with common appearance and diagnostic performance. This paper outlines the motivation to develop a generic modeling process for FFMs at the component level and the effort to implement that process through modeling conventions and a software tool. The implementation of this generic modeling process within a fault isolation demonstration for NASA's Advanced Ground System Maintenance (AGSM) Integrated Health Management (IHM) project is presented and the impact discussed.

Nomenclature

AGSM	=	Advanced Ground System Maintenance
ConOps	=	Concept of Operations
ETA	=	Extended Testability Analysis
FFM	=	Functional Fault Model
FMECA	=	Failure Modes, Effects & Criticality Analysis
GCM	=	Generic Component Model
GEMINI	=	GENeric Model INstantIator
GUI	=	Graphical User Interface
ICM	=	Instantiated Component Model
IDU	=	IHM Demonstration for UPSS

[1] Aerospace Engineer, GRC-LCC, 3000 Aerospace Parkway/VPL-3, Brook Park, OH 44142, Member.
[2] Systems Engineer, GRC-LCC, 3000 Aerospace Parkway/VPL-3, Brook Park, OH 44142, Senior Member.
[3] Engineer, KSC-NE, ESC-25. Kennedy Space Center, FL 32899.
[4] Controls Engineer, GRC-LCC, 21000 Brookpark Road/77-1, Cleveland, OH 44135.

IHM = Integrated Health Management
LO2 = Liquid Oxygen
S&MA = Safety & Mission Assurance
TEAMS = Testability Engineering And Maintenance System
UPSS = Universal Propellant Servicing System
VERA = VERification Analysis
VV&A = Verification, Validation and Accreditation

I. Introduction

FUNCTIONAL fault models (FFMs) are failure space representations of systems that define failure effect propagation paths within the system structure. These models can be used to provide insight into the ability of the system to meet requirements for fault detection and isolation. In the early design stage, this insight can be used to verify requirements and could result in design changes that would be more costly to implement if left until later in the design process. The FFMs can also be maintained and advanced through the system development process and ultimately provide a real-time diagnostic assessment capability. The versatility of these models makes them a valuable system engineering capability.

The development of FFMs is still very much an art form. Each individual modeler can represent the same component in a number of different ways. The FFMs addressed in this paper are developed using a commercial product called Testability Engineering And Maintenance System (TEAMS) from QualTech Systems Inc., which has been utilized in a number of NASA projects.[1-7] Several efforts have been made during these development projects to standardize the modeling conventions and practices[8,9] and to develop support tools[10-12] with the objective of reducing the time required for model development, verification, validation, and accreditation and decreasing the issues encountered during integration of independently developed FFMs.

Even with this drive to establish a core set of conventions and practices, and a suite of support tools, manual FFM development can be costly to a project both in time and resources. The individual creation of each component model results in potential inconsistency in the way common components are represented across the system. Such inconsistency can be confusing to domain experts who may review the model resulting in detrimental cost and schedule impacts to the verification and accreditation processes. In addition, the inconsistency can lead to an imbalanced diagnosis across a larger integrated system where similar components could have varying degrees of modeled fidelity and detail. This often results in confusion from the overall system diagnostic assessment. For example, similar components in different subsystems within the same FFM could model the same failure mode in a completely different ways or identical failure modes with different names, both requiring additional processing to align them.

Previous diagnostic modeling efforts have identified the advantages of establishing a library of component models that could be used as building blocks in constructing future system models.[3,8] To reduce the time and resources required for FFM development and verification and ensure more commonality in the representation across the entire system model, the concept of generic component modeling was developed. This concept attempts to take advantage of generic representations of common components across the modeled system. Each generic model is intended to represent a specific type of component, complete with failure mode information, elements required to transition the failure information propagated to and through the component, and any observation or test points, which in general represent the sensors in the system, available within the component. These generic components can be verified through unit testing and review, and will be the basis for a generic component library that would be used during future component model instantiation. The process of model instantiation defined here is a programming term meaning the process of producing specific representation of some object by replacing variables, in this case textual placeholders within the FFM, with values specific for the given component.

This paper will provide an overview of the FFM development process with the incorporation of the generic modeling concept, and provide details about the generic modeling conventions developed and the supporting instantiation tool, GENeric Model INstantIator (GEMINI). The paper will also present specific examples of Generic Component Models (GCMs) that expose the capabilities of the GEMINI software and highlight significance of the GCM modeling conventions. Finally, the paper will summarize the implementation of this concept under NASA's Advanced Ground System Maintenance (AGSM) Integrated Health Management (IHM) project.

II. Background, Motivation and Development

A. Overview of FFM Development Process

Development of FFMs begins with gathering and studying source data that describes the configuration, operation, and failure information about the system being modeled. Configuration information is extracted from schematics or drawings, parts lists (including components and instrumentation), parts descriptive data (such as parts catalogs), and system interface documents. Operations information (such as sub-system and component operating timelines) is typically found in ConOps (Concept of Operations) documents and draft operating procedures. Failure mode and reliability data are found in Failure Modes, Effects & Criticality Analysis (FMECA) and Reliability Analysis documents created by Safety & Mission Assurance (S&MA) personnel. If these documents are not available for the system to be modeled, data can be extracted from historical examples and reliability databases. Although missing data does not preclude the development of the FFM, it could impact the model's usefulness. Gaps in, and additional explanation of, source material is further extracted from interviews with domain experts.

Once data has been gathered and an understanding of the system developed, components for which failures would impact system-level operation are identified and associated data assembled for inclusion in the FFM. The FFM developer (drawing on his/her development experience and referencing applicable modeling conventions) uses the TEAMS-Designer development environment to create/insert: (1) the skeleton of the system-level FFM; (2) details in the sub-system, component, and sub-component hardware elements of the model, including failure modes and nominal transition of system properties through the respective element; (3) test points (typically associated with actual instrumentation elements) and associated test information. TEAMS-Designer analysis and reporting options, as well as a number of additional support tools, are used to verify the integrity of the model and its diagnostic assessments during each system operating mode.

As the FFM development process has evolved over time and multiple applications, FFM developers have created additional guidance documents, sub-processes, and tools in order to create efficiencies in the overall process. Especially for large FFMs, where sub-FFMs are created by multiple developers, the consistent utilization of guidance documents is critical to efficient integration into the top-level FFM. The additional support tools reduce the time necessary to verify, validate, and accredit the final system FFM. The generic modeling process described in the following sections is one of the sub-processes that can significantly reduce the time and resources required to develop a system-level FFM and its associated components.

B. Generic Modeling Concept and Goals

The reduction of build time and resources, as well as modeling consistency, are just some of the incentives for developing an FFM development approach that builds models using generic component libraries rather than by unique individual components. Historically, FFMs have been built from scratch with each model element being developed independently based on available design documentation. Because there can be more than one way to model the same system component's failure space, it is often the case that common model elements can be represented quite differently, even within the same FFM.

Typically an FFM represents the functional failure propagation paths through the hierarchical hardware breakdown of a system. At lower levels, these models represent components that provide a specific required functionality, but are not stand-alone. Examples of components can include valves, circuit boards, sensor transducers, filters and tanks. Within the FFM, the general functionality of common components and their failure mode definition should be identical regardless of the application.

For example, a filter's nominal function is to remove impurities or contamination from the flow path. Increase in fluid contamination from degradation (e.g., extreme seal wear) of an upstream component could cause flow blockage at the filter. The filter itself could have failure modes of blockage, loss of filter capability or external leakage. This level of representation would be common regardless of application or working fluid. Therefore, capturing this functional and failure information with a generic component for each filter element in the system FFM would ensure consistency and reduce time for overall model development.

The generic modeling concept is simple: establish a baseline set of component-level FFMs that can be used to build larger system-level models. This baseline set of component models can be vetted and tested to ensure a level of verification that should increase confidence and decrease accreditation costs of the final system FFM. To facilitate the instantiation of the generic component models, a special set of modeling conventions were developed along with a software tool called GEMINI that walks the user through the process. The conventions and software tool will be discussed in more detail in subsequent sections.

The GCMs are built manually. However, instead of specific design information at key locations in the model, special text fields are entered as placeholders. These placeholders will be replaced in the instantiated component

models with available design information. In addition, each GCM contains external modules and test-points that are used for GCM unit test verification. External modules contain failure modes that test the connectivity of the internal paths in the GCM under various configurations and proper nominal transition of effect passing through the GCM. External test-points enable the establishment of specific tests that determine the exiting content of each GCM propagation path. These external testing elements are not needed in the final instantiated component model (ICM), so they are removed during the instantiation process. This concept of building the GCM as a complete stand-alone FFM facilitates unit test verification at the generic component level, greatly reducing the overall verification costs of the final system-level FFM by eliminating unit testing of each individual ICM.

Each ICM created from the same GCM will have the same generic attributes, visual appearance and functionality across the entire system-level FFM. This result will simplify the review and accreditation processes by the domain experts. Once the ICMs are generated and reviewed, they can be integrated into a system-level FFM module that represents the component decomposition of the system. The entire proposed process is shown in Fig. 1 below. Common components are identified and stand-alone GCMs are developed, tested and maintained in a library. Individual GCMs can be instantiated to represent a specific component in the modeled system. Meanwhile a system-level FFM is developed with components represented by place-holder modeling elements. During the integration, each place-holder element is replaced with the associated ICM resulting in a final system-level FFM where the individual components have been verified and are consistent across the model.

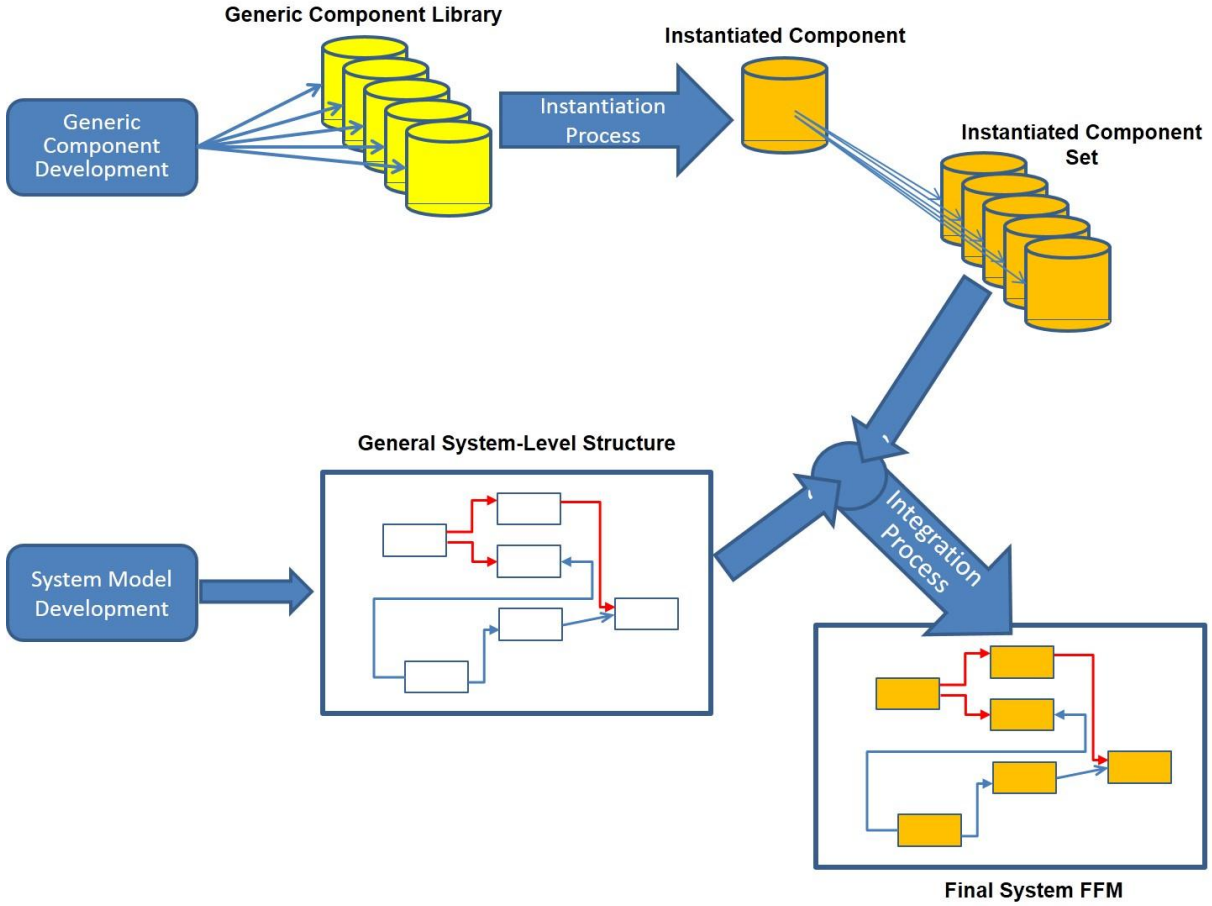

Figure 1. FFM development process incorporating the GCM concept.

C. Generic Modeling Conventions

The GCMs have their own special conventions in addition to the currently established NASA TEAMS modeling conventions.[9] These GCM conventions establish specific fields within the TEAMS model elements' name and properties that can be easily located and updated with specific component information during the instantiation

process. The rest of the GCM properties and fields are not modified during the instantiation process as are the generic component attributes that each specific common component model will maintain.

The TEAMS models are made up of containers called modules that represent the system's structure. There are two types of modules that make up the lowest level of the TEAMS model: failure mode modules and functional mapping/blocking modules. Failure mode modules describe the failure mechanism of a component; these modules contain the functional failure effects. The mapping and blocking modules represent abstract concepts, such as the place in a system where a failure is mapped from one type to another (i.e., mapping) or where the propagation of a failure is discontinued (i.e., blocking). Mapping modules allow the user to represent the transition of a failure effect from one physical state to another. For example, a power failure may be mapped from "loss of current" to "loss of rotational velocity" within a motor component. Similarly, blocking modules prevent failure effects from propagating beyond their scope. The motor component may also have a blocking module to prevent "loss of current" from propagating through the motor to other components downstream that do not have a direct connection to the current source. The two low-level module types – failure modes and mapping/blocking – are then contained by higher level subcomponents and components that are abstractions of the actual system components, such as valves, filters, pipes, motors, transducers, etc.

A simple FFM representation of a motor component, Fig 2, contains a single failure mode module (red hatched box), a sensor module (blue hatched box) and a single mapping/blocking module (green hatched box). Other modeling elements included in the example are a switch (small white box to the right of the failure mode module) and a test-point (solid green circle). Switch elements are used to redirect the internal propagation paths within the GCM. A switch possesses internal attributes that label the switch states assigned to the switch's input or output ports, and are used by TEAMS to set the switches position and hence the module's flow-path configuration. Test-points are the observation points in the system propagation paths and often represent the system's sensors. The test-points have an internal attribute that lists the specific tests defined for that test-point while the tests themselves associate specific functional fault effects to test outcomes. The text labels in the connecting lines represent the possible failure functions as they transition through the motor module.

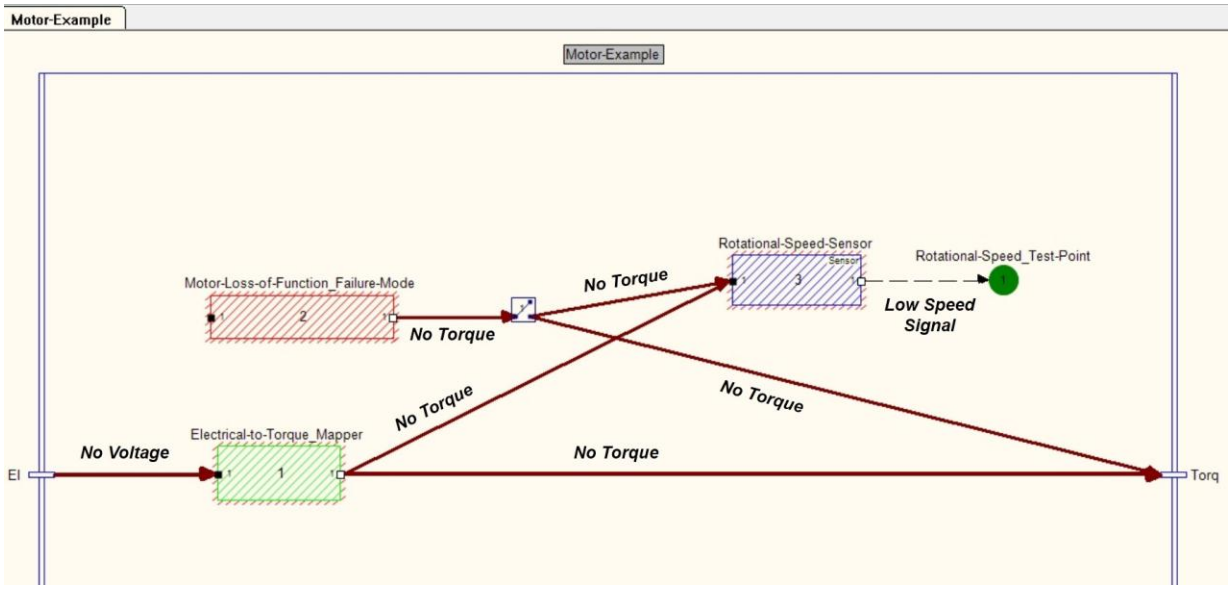

Figure 2. FFM representation of a motor.

Currently a GCM can contain two or three hierarchical levels to represent a component. Figure 3 illustrates a GCM with three hierarchical levels; the details of this example GCM are explained later in the paper. The top-level of the GCM contains the generic component being modeled along with any external testing elements that could be used to verify the connectivity and internal properties of the modeled generic component. The second level if needed could contain subcomponent modules that makeup the generic component. At the lowest level, a GCM can contain failure mode and mapper/blocker modules as well as other modeling elements such as test-points, and switches.

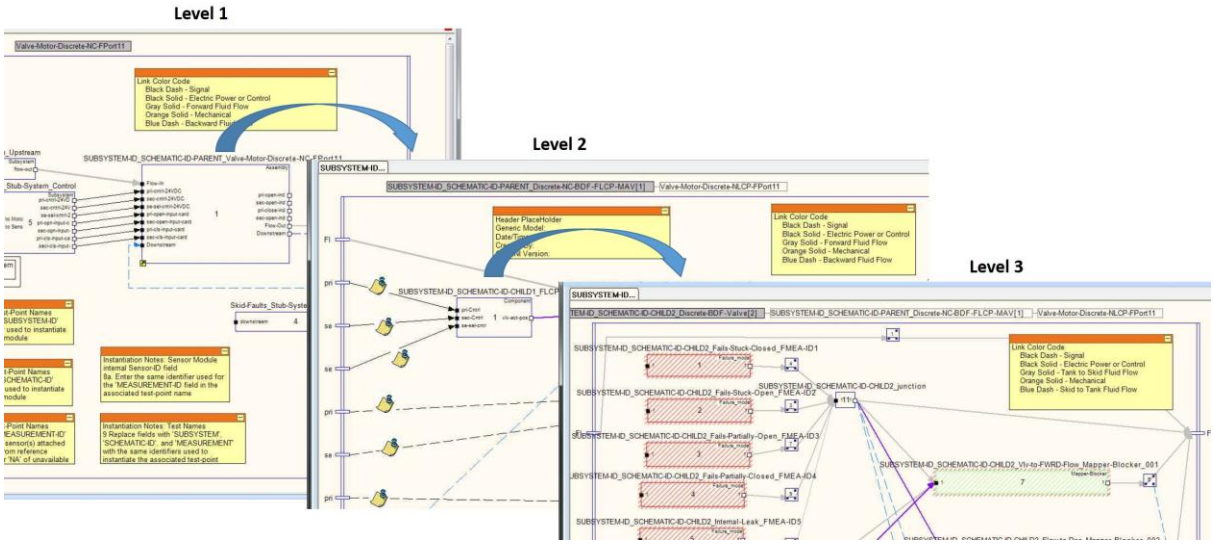

Figure 3. An example GCM with three hierarchical levels.

Table 1 presents a summary of most GCM conventions currently developed. The first column indicates the TEAMS modeling element within the GCM, and the second column indicates that element's property being defined by the convention. The third column details the format of the convention. As indicated in the NASA TEAMS modeling conventions,9 modeling element names and certain assigned attributes are text strings separated by underscore characters, '_', to define certain fields. Bold font in the convention format indicates that the field must contain that exact text string and that the string will be updated or replaced during the instantiation process. Fields in italics font and contained within angle brackets (i.e., < >) are text strings that can be replaced by the user during the instantiation, but they don't have to be. Often these refer to general component descriptions that the user may want to clarify for the final instantiated model. Text in square brackets (i.e., [ ]) indicate fields that are not available for updating during instantiation. These fields should be left untouched to ensure consistency within the final FFM.

Table 1. Summary of Generic Conventions

TEAMS Model Element	Element Property	GCM Format Convention
2nd Order GCM Module	Name	**SUBSYSTEM-ID_SCHEMATIC-ID**_*<GCM-Description>*
3rd Order GCM Module	Parent Name	**SUBSYSTEM-ID_SCHEMATIC-ID-PARENT**_*<GCM-Description>*
	Child Name	**SUBSYSTEM-ID_SCHEMATIC-ID-CHILD{n}**_*<Sub-Component-Description>*
Failure Mode Module	Name	**SUBSYSTEM-ID_SCHEMATIC-ID**_[Failure-Mode-Description]_**FMEA-ID{i}**
	Attached Function	**(SUBSYSTEM-ID)**_[Failure-Mode-Description]
Mapper-Blocker Module	Name	**SUBSYSTEM-ID_SCHEMATIC-ID**_[Mapper-Description]_Mapper-Blocker
	Mapped Function	**(SUBSYSTEM-ID)**_[Failure-Mode-Description]
Test Point	Name	**SUBSYSTEM-ID_SCHEMATIC-ID**_[Test-Point-Description]_**MEASUREMENT-ID**
	Attached Test	**SUBSYSTEM-ID_SCHEMATIC-ID**_[Test Type] _[Test-Description] _**MEASUREMENT-ID**
Switch	Standard State	**SSID_SCHID-CH{n}**_[Switch-State]
	Bidirectional State	**Forward** or **Reverse**

D. Instantiation Tool – GEMINI

To support and further improve the instantiation process, a stand-alone JAVA software package called the GENeric Model INstantIator (GEMINI) was developed. GEMINI provides a graphical user interface (GUI) (see Fig. 4) that guides the user through the instantiation process. As part of that process, it restricts user inputs so that resulting ICMs contain design-specific information and adhere to the NASA-developed modeling conventions for FFMs. GEMINI relies on the GCM adhering to both NASA TEAMS modeling conventions as well as the generic modeling conventions described in the previous section.

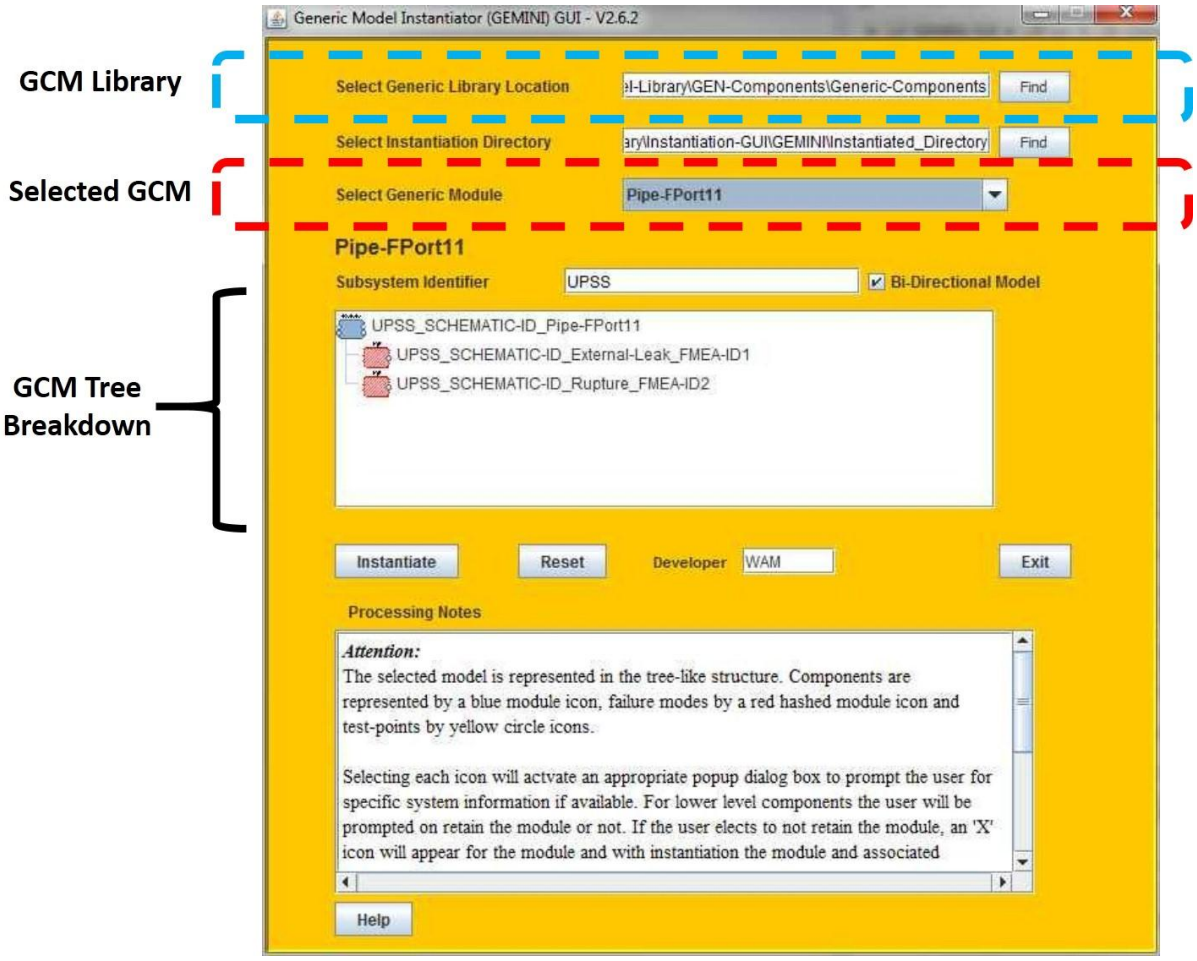

Figure 4. An example of the GEMINI GUI after a GCM has been selected from the designated library

As illustrated in the figure, the user can import a GCM from a specified library of GCMs (light blue dashed area). The user can select the model from the library using a pulldown menu shown in the red-dashed area in the figure. The program determines elements within the GCM that need to be instantiated and populates a tree breakdown of the GCM as indicated in the figure. Selecting the individual elements of the tree drives the instantiation session and prompts the user for the appropriate input for that selected element. The GCM can be further modified to remove any sub-components, failure modes, or test-points that are not needed in the actual ICM. Upon completion of the user inputs, the program generates an ICM with all the updated element names and internal fields and generates a temporary model structure that enables the user to load the ICM into a TEAMS model for review. Once the ICMs have been reviewed, they can be inserted manually into the overall system model.

During a typical session, the user can walk through a GCM updating all the pertinent data as prompted by GEMINI and finally click the 'Instantiate' button to generate the specific ICM. Then the user has the option of returning to the edited fields to generate a new ICM of the same component type as the previous ICM, updating only the data needed to identify the new ICM from the previous one, or selecting the 'Reset' button to reset all the placeholders back to the GCM values for the currently loaded component, or he/she can select a different GCM from the library. This functionality allows the user to generate multiple ICMs that vary slightly in specific data, in a single session for rapid ICM development.

The development of GEMINI is the central piece of this generic model convention process. The GCMs have many common placeholders located throughout the model in element names and internal properties that must be updated simultaneously and consistently. This program was designed specifically to address this requirement and it does this while at the same time enforcing applicable modeling conventions. GEMINI reduces the model development time by eliminating the manual editing of the GCMs needed previously to instantiate the GCMs, and at the same time eliminates certain verification testing of the final ICMs through the model convention enforcement.

III. Application

A. Implementation of GCM Concept under AGSM IHM Project

The FFM-modeling process – illustrated in Fig. 5 and described here – was utilized during the development of a fault isolation module of the Universal Propellant Servicing System (UPSS). The fault isolation module was one IHM component of the AGSM's IHM Demonstration for the UPSS (IDU). The resulting FFM represented the cryogenic and pneumatic components that are part of the UPSS Liquid Oxygen (LO2) feed system. After initial review of source data, 97 hardware elements were identified and summary data were captured in a hierarchically-organized component-to-model ("Comp's-to-Model") workbook which included the indicated source technical data (step 1a) and formatted according to the indicated guidance documents (step 1b). In step 2a, 25 sample component FFMs that were either derived from previous FFM projects or developed from scratch were collected in a GCM library. Thirteen (13) of the GCMs were deemed to be applicable to the IDU application and underwent further development and unit verification testing and review.

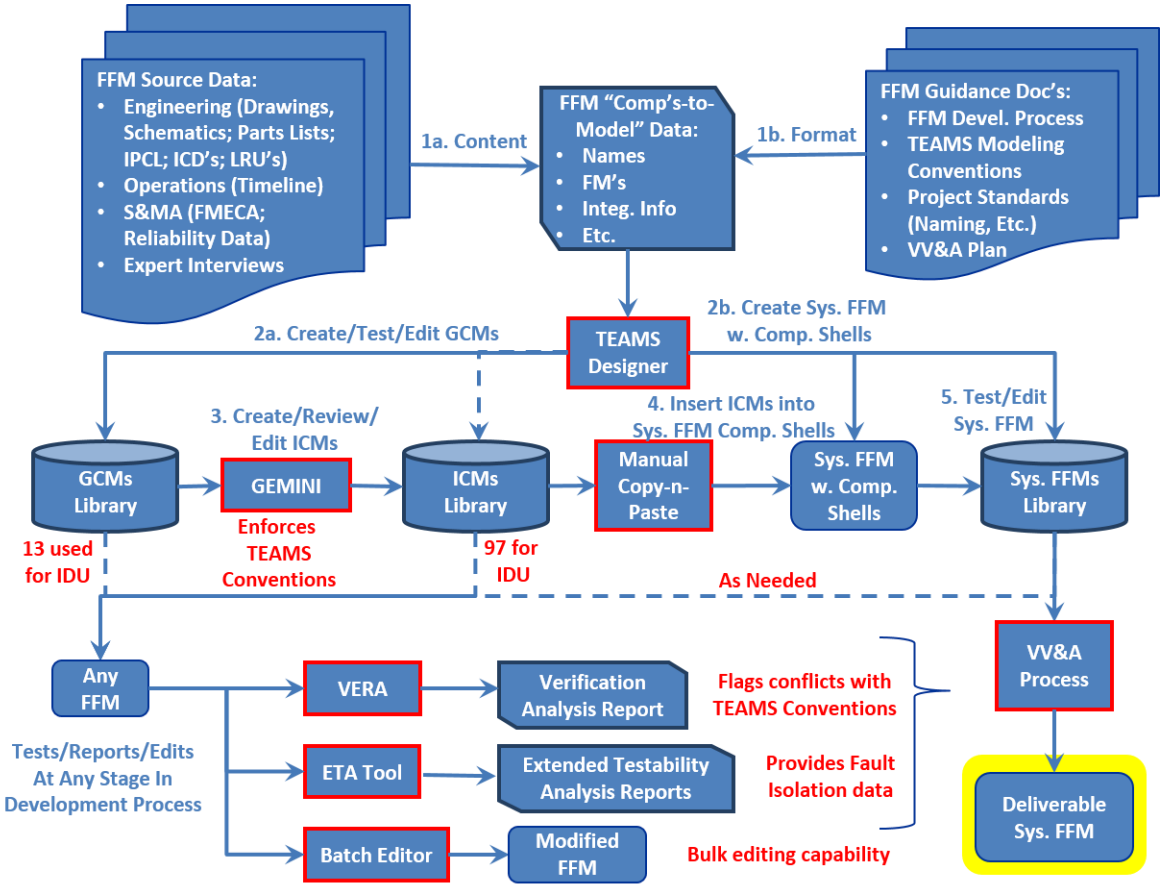

Figure 5. Overall IDU FFM Generic Modeling Process

In parallel with the development of the GCMs, a system-level model containing empty component shell elements was developed (step 2b). The pre-instantiation review of the system FFM included: (1) verifying the component shells so that the type and order of the ports in the FFM elements were consistent with those of their comparable GCMs; and (2) verifying proper propagation path connections between the component shells.

When the required GCMs and system model verification efforts were complete, the GEMINI tool used the applicable 13 GCMs to create an ICM for each of the 97 hardware elements (step 3). In the GEMINI GUI, the applicable component sub-elements were selected and individual element/sub-element names, failure modes, sensor IDs, etc., documented in the "Comp's to Model" workbook were entered into the appropriate GUI fields. The resultant ICM was then deposited into the ICM library. The instantiation process was repeated for each of the 97 UPSS components represented in the FFM, first selecting the appropriate GCM representing the component type

and then replacing the placeholder fields with the component's particular data. Following completion and review of an ICM, it was integrated into its comparable shell element in the system model (step 4).

During the instantiation process, an ICM Creation log was maintained. This log is used to track: (1) the ICM ID; (2) the GCM utilized to create it; (3) status of GCM/System component ports matching; (4) ICM creator and creation/update dates; (5) peer review reviewer and date; (6) ICM-to-system FFM integrator and date. Once the ICMs had been integrated into the system FFM, system-level verification tests and reviews were carried out (step 5).

At various points in the development process, standalone support tools were used to debug and test the FFM. Example tools [10-12] included: (1) the newly-created Verification Analysis (VERA) tool which assesses the degree to which an FFM agrees with many of the published modeling conventions; (2) the Extended Testability Analysis (ETA) Tool which creates a variety of additional test reports that helps the developer assess the FFM's Fault Isolation capability; (3) the Batch Editor which was used to make bulk edits to a number of like features in the system FFM.

For the IDU application using this generic component development process, only thirteen (13) GCMs were developed and verified as components, whereas the previous methodology would have require all 97 components models to be developed and verified. While the ICMs still have to be checked for accuracy from the instantiation process, the GCM provides a framework for the component that will reduce errors due to modeling inconsistencies. This resulted in a reduction of over 80% in costs, time, and resources required for this phase of the FFM development. This cost will be further lowered if and when the GCMs are reused to build future FFM system models.

B. Examples of GCM Development

The GCM library for the AGSM IDU project was evolved from a similar previous effort to build an FFM that represented a cryogenic propellant transfer test bed. Recognizing that the AGSM IDU and previous demonstration system had many common components, the FFM development team extracted portions of the original FFM in an attempt to identify common components. Components were grouped by features and hardware specifications. For each group of components, a generic FFM representation was developed, complete with all possible failure modes and functional flows paths anticipated. Eventually, twenty-five (25) generic components were created as a result of this effort and the sophistication of these GCMs ranged from simple (e.g., a simple pass-through pipe) to complex (e.g., a variable position motorized valve). The purpose of this section is to walk through an example GCM, highlighting the key features in order for the reader to gain an appreciation of the potential for future GCM development as other target systems are identified.

Figure 6 shows the top-level view for the discrete-position, normally positioned in the last commanded position motorized valve. The red dashed box highlights the actual GCM that will be instantiated and inserted into the system model. The yellow boxes in the figure are internal note elements that the GCM developer uses to provide comments or instructions that indicate intent and modifications in the GCM. Modules and test-points outside the red-dashed box are elements that the developer can provide to exercise the GCM during its verification testing. Elements external to the red-dashed box are stripped away during the instantiation process.

Drilling into the valve GCM, Fig. 7, the model contains four child sub-components. Two of these children are identified as sensor transducers, having the blue-hatched background. This GCM also contains four test-points, two for each sensor transducer that are indicated by the blue boxed-circles. The current GCM has a primary and a secondary measurement for each transducer as is typical in ground propellant loading applications. Future GCM developments may expand this to three or more measurements depending on the application. During the instantiation process, the user can provide a unique schematic identifier for each sub-component, or use the schematic identifier of the parent if none is specified. The software tool will also insert the identifier information from the sensor transducers into the test-point(s) names and associated test(s) names linked to those transducers. Also during this process, the user may remove any of the sub-component modules or individual test-points in order to match the content of the target ICM. The created ICM also contains a note element, located at the top-left of the module that specifies instantiation information such as instantiation date, developer, and version of the GEMINI software used to create the ICM.

At the lowest level, the valve GCM contains base-level elements, failure mode modules, mapper/blocker modules and switches as shown in Figure 8. The failure mode modules (red-hatched rectangular boxes), the mapper/blocker modules (green-hatched rectangular boxes) and the switches (small white square boxes) all have internal properties that need to be updated simultaneously with the overall GCM instantiation. The figure displays an example of the internal property window for a failure mode module to show the internal fields that are updated during the instantiation process. Without an automated instantiation process through GEMINI, each one of these internal fields would need to be located and updated manually.

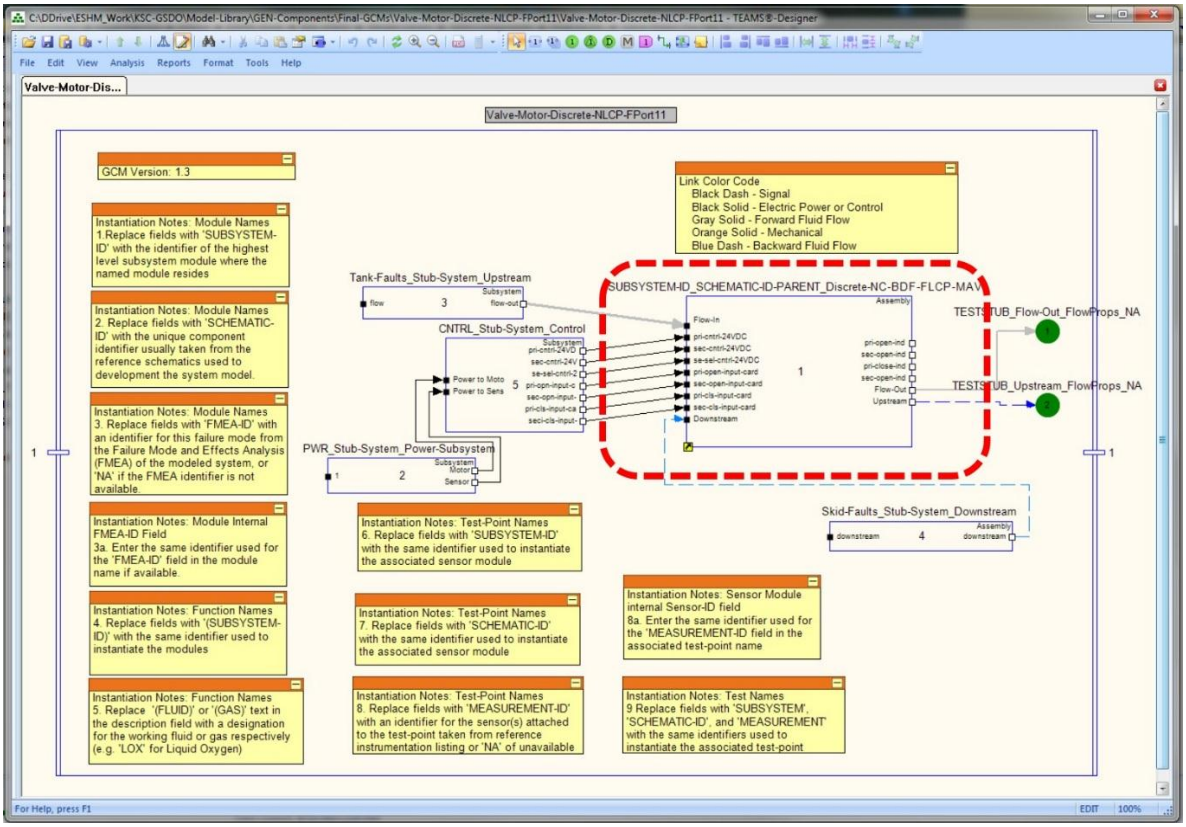

Figure 6. Top-Level view of an example valve GCM

IV. Summary

The generic modeling process that was refined under the AGSM IDU project at NASA provided significant savings in terms of development and verification of FFMs. Applying this process demonstrated greater than 80% cost savings during the component modeling phase of the FFM development. The high-level modeling process relied heavily on the development of strict conventions for naming, and failure representation as defined in the NASA "Testability Engineering and Maintenance System (TEAMS) Modeling Conventions and Practices" document. The conventions steered development of a GCM library that can be re-used and augmented for future projects, and the GEMINI tool can be employed by future FFM developers to quickly build a new FFM using the GCM library. In addition, the standardized modeling process provides the traceability necessary to perform verification, validation and accreditation of both the AGSM IDU and future FFMs. Finally, the generic modeling process ensures a uniform, standard appearance of the FFM and will result in consistent diagnostic assessments as the FFMs are used in the future for both offline and real-time diagnostic assessments.

Acknowledgements

This effort was performed under the direction the Advanced Ground Systems Maintenance (AGSM) project at the NASA Kennedy Space Center (KSC). The AGSM project is part of NASA's Ground Systems Development and Operations Program. The authors would like to acknowledge the support of Barbara Brown, Jose Perotti, Bob Ferrell, and Mark Lewis of KSC.

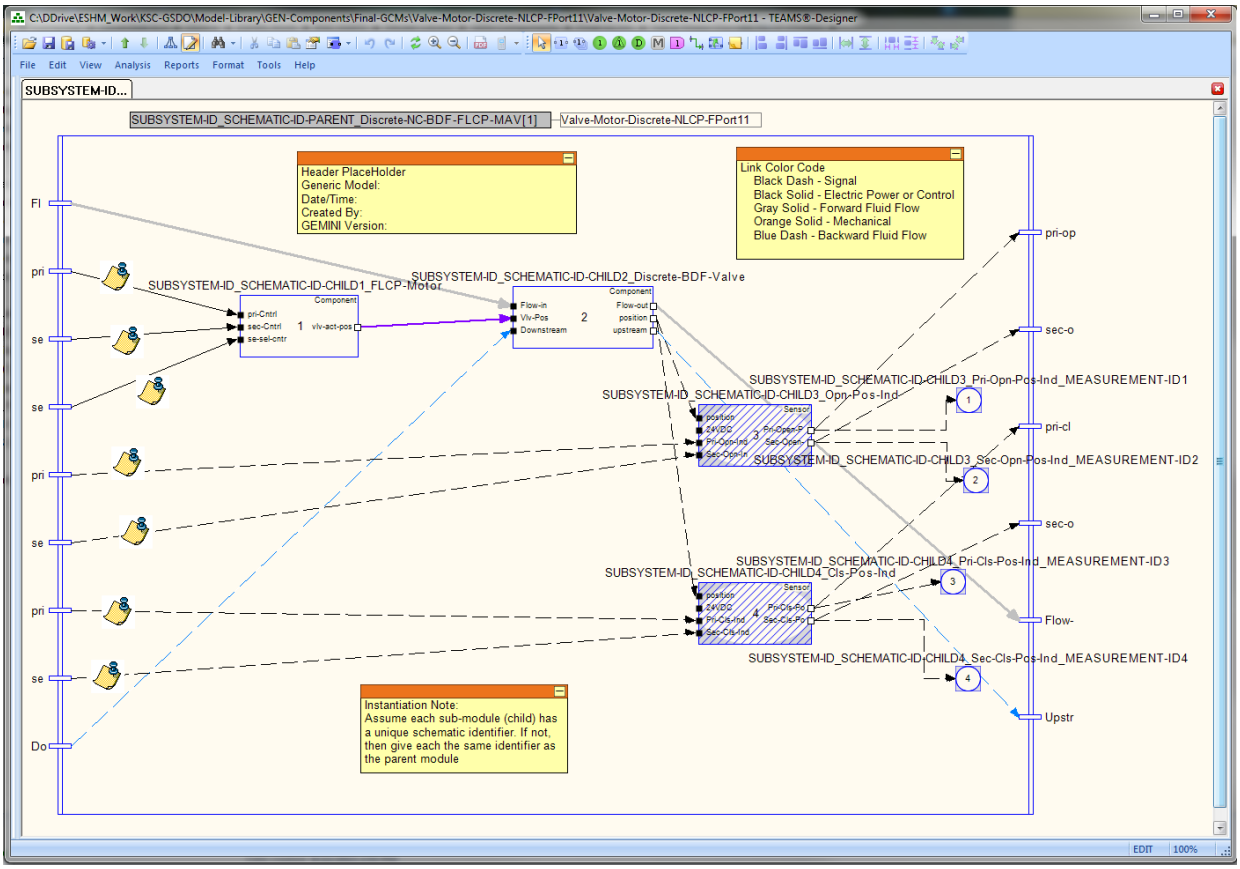

Figure 7. Mid-Level view of an example valve GCM

References

[1] Kurtoglu, T., Johnson, S., Barszcz, E., Johnson, J. and Robinson, P., "Integrating System Health Management into the Early Design of Aerospace Systems Using Functional Fault Analysis," *2008 International Conference on Prognostics and Health Management*, Denver, Colorado, October 2008.

[2] Maul, W., Melcher, K., Chicatelli, A. and Johnson, S., "Application of Diagnostic Analysis Tools to the Ares I Thrust Vector Control System," *AIAA InfoTech Conference*, Atlanta, Georgia, April 2010.

[3] Ferrell, B., Lewis, M., Perotti, J., Oostdyk, R. and Brown, B., "Functional Fault Modeling of Cryogenic System for Real-Time Fault Detection and Isolation," *AIAA InfoTech Conference*, Atlanta, Georgia, April 2010.

[4] Ferrell, B., Lewis, M., Perotti, J., Oostdyk, R., Spirkovska, L., Hall, D. and Brown, B., "Usage of Fault Detection Isolation & Recovery in Constellation Launch Operations," *AIAA SpaceOps 2010 Conference*, Huntsville, Alabama, April 2010.

[5] Schwabacher, M., Martin, R., Waterman, R., Oostdyk, R., Ossenfort, J., and Matthews, B., "Ares I-X Ground Diagnostic Prototype," *AIAA Infotech Conference*, Atlanta, Georgia, April 2010.

[6] Spirkovska, L., Aaseng, G., Iverson, D., McCann, R., Robinson, P., Dittemore, G., Liolios, S., Baskaran, V., Johnson, J., Lee., C., Ossenfort, J., Dalal, M., Fry, C., and Garner, L., "Advanced Caution and Warning System, Final Report - 2011," NASA TM-216510, 2013.

[7] Frank, J., Aaseng, G., Dalal, K., Fry, C., Lee, C., McCann, R., Narasimhan, S., Spirkovska, L., Swanson, K., Wang, L., Molin, A., and Garner, L., "Integrating Planning, Execution and Diagnosis to Enable Autonomous Mission Operations," *2013 International Workshop on Planning & Scheduling for Space (IWPSS)*, Moffett, California, 2013.

[8] Ferrell, B., Lewis, M., Perotti, J., Oostdyk, R., and Brown, B., "Functional Fault Modeling Conventions and Practices for Real-Time Fault Isolation," *SpaceOps 2010 Conference*, Huntsville, Alabama, April 2010.

[9] K0000190582-GEN, "Testability Engineering and Maintenance System (TEAMS) Modeling Conventions and Practices," National Aeronautics and Space Administration, 2014.

[10] Maul, W., Fulton, C. and Melcher, K., "Extended Testability Analysis Tool User Guide," *AIAA InfoTech Conference*, St. Louis, Missouri, March 2011.

[11] Barszcz, E., Robinson, P., and Fulton, C., "Tools Supporting Development and Integration of TEAMS Diagnostic Models," *AIAA InfoTech Conference*, St. Louis, Missouri, March 2011.

[12] Bis, R., and Maul, W., "Verification of Functional Fault Models and the Use of Resource Efficient Verification Tools," *AIAA InfoTech Conference*, Kissimmee, Florida, January 2015.

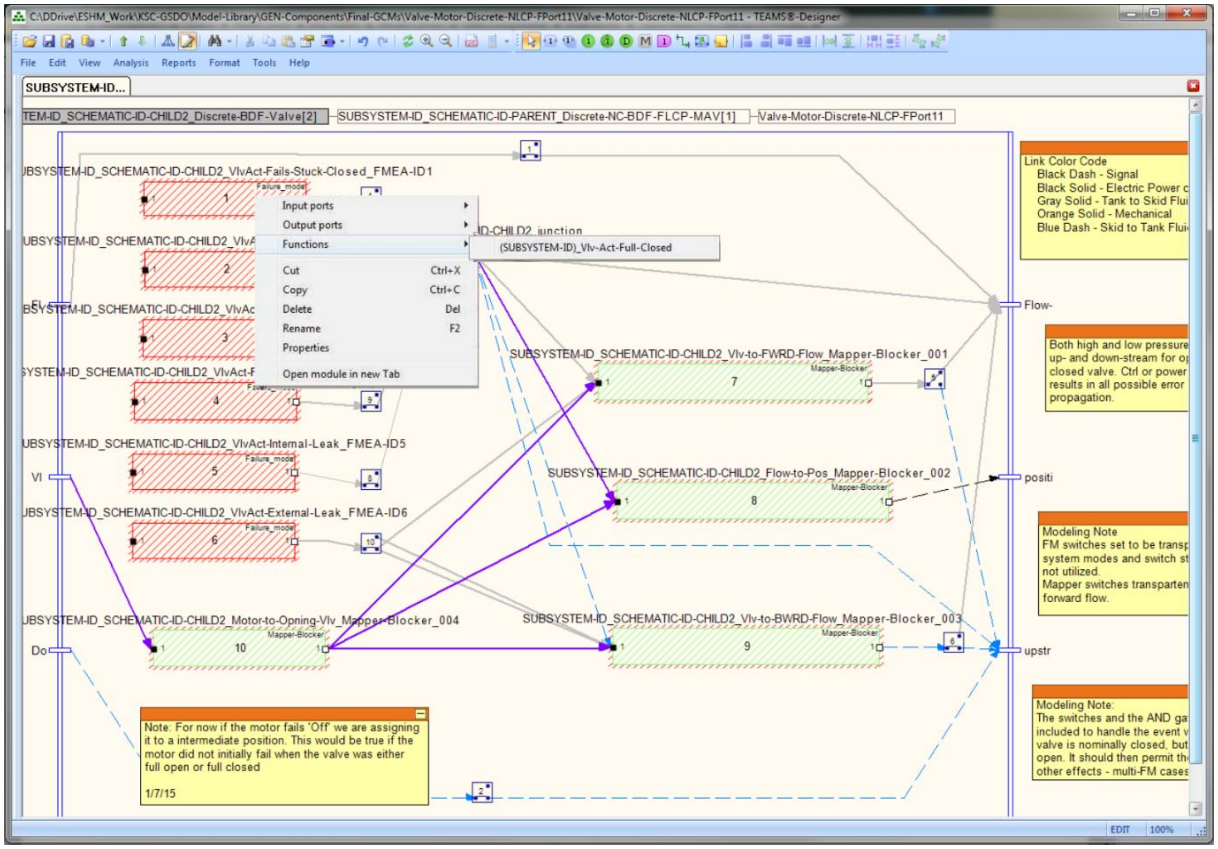

Figure 8. Lowest-Level view of an example valve GCM