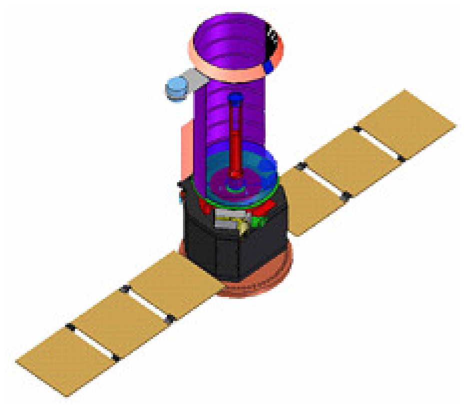
QUICKLOOK FINAL REPORT

Version 1.19



By Tactical Science Solutions, Inc.
in support of the Tactical Satellite-3 design effort

May 30, 2007

Group:	Tactical Science Solutions, Inc.	
Authors:	David Alexander	dalexander@cybermetric.com
	Soroush (Kevin) Sadeghian	kevin.sadeghian@saic.com
	Thomas Saltysiak	saltysiak@hotmail.com
	Siroos Sekhavat	siroos.sekhavat@ngc.com
File Name:	Quicklook Final Rerport v1.18.doc	
Status:	Final	
Number of Pages:	35	
Created Date:	April 20, 2007	
Saved Date:	Wednesday, May 30, 2007	

Version Modification Overview

Version	Date	Comments
1.00	04/20/07	Initial outline
1.01	04/22/07	Added overview and purpose section
1.02	04/23/07	Added learning curve evaluation section
1.05	04/25/07	Consolidated all sections
1.06	04/26/07	Section updated; grammar corrected
1.07	05/01/07	Reworked Sections 3 and 4.3 based on Dr. Laskey's and Heather's recommendations
1.11	05/02/07	Consolidated all reworked sections
1.14	05/02/07	Following new sections added: Executive summary; design diagrams for SysML; evaluation section; and definition table formatted to match acronym table.)
1.15	05/02/07	Grammar corrected
1.16	05/02/07	Introduction added; grammar corrected
1.17	05/03/07	Conclusion added; grammar corrected
1.17	05/03/07	Small changes; grammar corrected
1.18	05/03/07	Final document
1.19	05/30/07	Acknowledgement and references section added

EXECUTIVE SUMMARY

Introduction

The Tactical Science Solution (TSS) Team at George Mason University has been asked by The Aerospace Corporation to evaluate the Systems Modeling Language (SysML) as a new and emerging general-purpose modeling language that supports Model-Based Systems Engineering. In order to conduct this evaluation, the TSS Team decided to design a small satellite system using SysML as the modeling language. The analysis portion of the project will provide the stakeholders with feedback on the usability, capabilities, scope and limitations of SysML.

Project Overview

In order to evaluate SysML's effectiveness, the TSS conceived a design project based on the Tactical Satellite-3 (TacSat-3) System. The process for evaluating SysML's effectiveness involved analyzing and modeling the TacSat-3 System in SysML and using this detailed design model as an instrument to evaluate SysML's usefulness. In addition, an executable model was created based on this design model, which allowed a behavior analysis and a trade study of design alternatives. The TSS Team collected engineering effort, training, and lessons learned data throughout the project. This information was used to evaluate the ease with which organizations could adopt Model-Based Systems Engineering using SysML. The TSS design efforts, as well as the modeling process of the TacSat-3 System, have been documented in detail for this project.

Results

The TSS evaluation of the new Systems Modeling Language (SysML) indicates that SysML adequately addresses systems engineers' needs. It serves the purpose of providing systems engineers the constructs required for an effective systems analysis and design based on Model-Based Systems Engineering methods. SysML is also effective in specifying requirements, behavior, requirements allocation and traceability, including placing constraints on the system. SysML re-uses some of the notations from UML 2.1 and provides additional extensions to satisfy the modeling language requirements. Unlike the traditional systems engineering approach, SysML allows systems engineers to build realistic models and simultaneously validate systems behavior without having to rely on costly prototypes.

SysML extends model-based engineering by providing systems engineers with the necessary constructs to capture requirements within the design model, which can then be used to validate the design artifacts. This capability provides a bi-directional traceability between deign artifacts and the requirements. In additional, SysML has a reduced set of formal constraints and semantics, which take less time to learn when compared to the Unified Modeling Language (UML). A SysML model can be easily translated into an executable model to perform behavioral analyses and trade studies. Furthermore, UML introduces software bias into the semantics, and lacks the necessary notations to satisfy systems engineers' needs, which SysML fulfills.

It should be acknowledged that the TSS Team greatly benefited from IBM Rational products and EmbeddedPlus Toolkits in the evaluation of SysML. Although employing

SysML to modeling and design is still under study, this project proves the usability and value of SysML in systems engineering analysis and design efforts. In summary, the TSS team has found SysML to be a powerful and effective modeling language for systems engineers, and can bring systems engineering and software engineering efforts closer together than ever before.

TABLE OF CONTENTS

1 OVERVIEW

Model-Based Systems Engineering (MBSE) is an emerging and promising approach to Systems Engineering. It is replacing traditional Systems Engineering development techniques such as document-based requirements and specifications, which can be incomplete, ambiguous, and easily misunderstood. Model-Based Systems Engineering is different from the traditional document-centric Systems Engineering in that MBSE delivers insight into the requirements, analysis and systems design phases, while providing better impact analysis, change control, and management. The Systems Modeling Language (SysML) is a language that is designed to support Model-Based Systems Engineering, and several tool packages in the market, such as IBM's Rational Systems Developer, are emerging to provide SysML support.

The Aerospace Corporation, a federally funded research and development center, would like to determine whether their customers, National Security Space (NSS) programs, should begin using the Systems Modeling Language (SysML) and where it is, or is not, appropriate to use. In order to meet this challenge, Aerospace has approached the George Mason University's (GMU) Systems Engineering and Operations Research (SEOR) Department for assistance.

Many of Aerospace's customers in National Security Space need help with architecture and modeling of both current and future systems. These customers often have a difficult time analyzing system-of-systems not only due to inherent complexity but the challenge of selecting the correct set of tools and methodologies from many currently available in the market. Aerospace would like to provide guidance to the customers who are considering using SysML by understanding the following questions:

- Are we sure that SysML, both the specification and tool implementation, is appropriate for use on NSS programs?
- What are the SysML specification limitations and SysML language capabilities?
- What is the learning curve involved with learning to effectively and efficiently use SysML?

In order to answer the above questions, the Tactical Science Solutions (TSS) team designed a system in the space systems domain using SysML, and used the resulting system design as a vehicle to evaluate the effectiveness of SysML and the supporting design tools. The TSS Team also used this design to conduct behavioral analyses and trade studies on different design alternatives. Finally, this design became the means for the TSS Team to learn and evaluate SysML and the IBM Rational Systems Developer tool, and to understand the capabilities and limitations of the two.

2 PURPOSE

The Aerospace Corporation asked the TSS Team to evaluate SysML, and to identify and document the capabilities and limitations associated with using this language. The purpose of this evaluation is to determine whether SysML is mature enough to become the standard modeling language in the work force as well as the academic field. To achieve this goal, the TSS Team designed a system in the space systems domain using SysML to evaluate the effectiveness of SysML and the design tools that support it. In order to satisfy other customer requirements and concerns, the TSS Team also conducted behavioral analysis on the designed system using the EmbeddedPlus SysML and Simulation toolkits.

The TSS Team documented the findings, conducted a trade study of the design alternatives as part of the preliminary design analysis, assessed the learning curve involved with learning SysML, and reported these results to the stakeholders in the form of a report, and several formal and informal presentations.

3 BACKGROUND

This section provides the background information needed to understand this project, which has been named Quicklook. In order to explore the effectiveness of SysML, a design model had to be developed. Choosing the appropriate system on which to base the design model was also essential for the successful and timely completion of the project. The background of the chosen satellite system as well as the background to understand the project is presented in this section.

3.1 Systems Modeling Language

The Systems Modeling Language (SysML) is a recently standardized modeling language agreed upon by the Object Management Group (OMG). SysML is based on the popular Unified Modeling Language (UML), which has been used for many years by software engineers as a language for model-based system design. Being developed in the software domain, UML does adequately address all the needs of systems engineers. SysML was conceived as an extension to UML to serve the need for systems engineering for both hardware and software systems. The OMG defines SysML in the following way:

> The OMG Systems Modeling Language (OMG SysML™) is a general-purpose modeling language for systems engineering applications. SysML supports the specification, analysis, design, verification and validation of a broad range of systems and system-of-systems. These systems may include hardware, software, information, processes, personnel, and facilities.[1]

SysML promises huge potential benefits for systems engineering, including:

- Ease of communication due to a common language
- Effective model-based systems engineering through SysML supported software tools
- A unified data repository that is consistent through all system views and diagrams
- The use of Object-Oriented design principles to reduce design and redesign effort through reuse of common design entities

[1] Object Management Group, Inc. (OMG). *OMG SysML Specification. Page 23.* http://www.sysml.org/docs/specs/OMGSysML-FAS-06-05-04.pdf

- Consistency between the SysML model and the resulting executable model

Through use of the Quicklook design project and executable model generation, the TSS Team can determine how well SysML actually lives up to these claims.

3.2 Design Project Selection

The Aerospace Corporation requested that the SysML evaluation in this project be based on a National Security Space (NSS) domain system, however, no specific system was provided or designated. The TSS Team developed a set of criteria for potential systems for this design project:

- The system must be in the National Security Space domain
- Enough information must be openly available to develop a Concept of Operations (CONOPS) and requirements
- The system must be of manageable complexity
- The system should have some functions in addition to a communications relay (such as imagery capture)

The TSS Team conducted research and chose the Tactical Satellite-3 (TacSat-3) System because it met the selected criteria for this design project. In addition, the TacSat-3 Program is of interest because it is an important part of a paradigm shift in NSS toward leaner, more responsive programs.

3.3 TacSat-3 Satellite Concept of Operation

The TacSat-3 System provides operational and tactical level warfighters[2] with responsive intelligence support. The low cost, weight, and modularity allows the system to be deployed from the time of request to operational phase within 7 days or less. Its advanced hyperspectral imagery components provide resolution of targets previously unobtainable. Finally, the TacSat-3's internal imagery processing and ground communications capabilities push relevant information directly to the warfighters who request it.

The ability to deploy and make the TacSat-3 operational in a short time-frame provides military commanders with the flexibility to respond to rapid changes in the global environment. The satellite's modular payload capability will allow for various communications and sensory packages. Once the Joint Forces Command (JFC) initiates a mission, the satellite payload can be configured and integrated with the launch vehicle within two days. The system will support launch vehicle processes to allow the launch within four days. Once at orbital altitude, the TacSat-3 is deployed, and will rapidly initialize within twenty-four hours and be ready for full operations for at least twelve months.

[2] Warfighter is a Department of Defense term synonymous with the User in a tactical military environment

During the operational phase, the TacSat-3 will provide hyperspectral tactical imagery to the warfighter in a timely manner. The satellite will support single-pass intelligence gathering missions. The TacSat-3 system will receive a collection task, which will encompass collecting imagery, processing the data, and transmitting the information to the warfighter within ten minutes. Warfighters will send target data to the satellite using standard Common Data Link (CDL) communications protocol. The TacSat-3 will receive, prioritize and process requests based on the mission parameters. Selected target imagery data will be acquired, locally stored, processed, and then downlinked to the user via CDL within ten minutes.

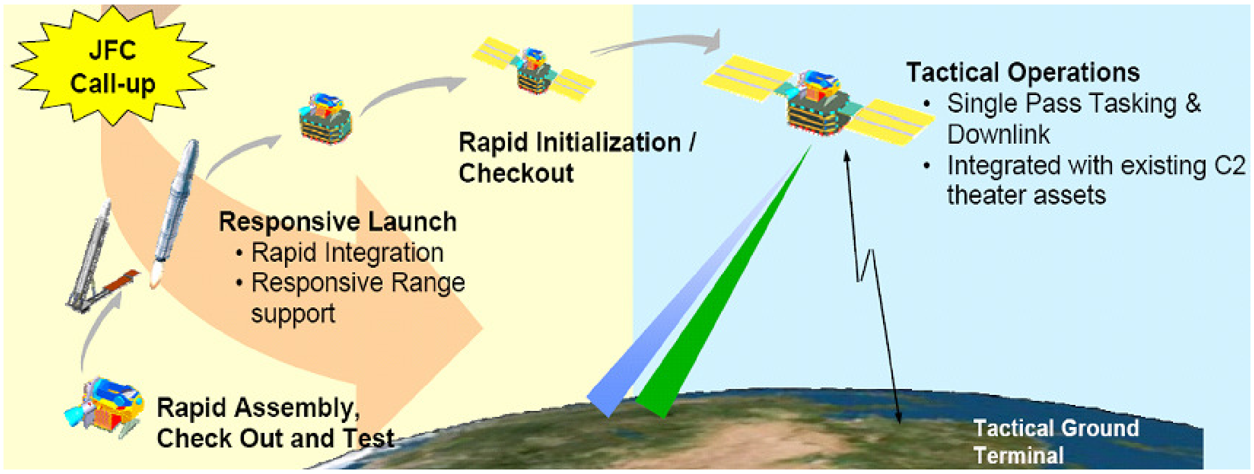

JFC Call-up
Rapid Initialization / Checkout
Tactical Operations
• Single Pass Tasking & Downlink
• Integrated with existing C2 theater assets
Responsive Launch
• Rapid Integration
• Responsive Range support
Rapid Assembly, Check Out and Test
Tactical Ground Terminal

Figure 3-1. TacSat-3 Operational Concept Graphic

*Picture from Air Force Research Laboratory Presentation: TacSat-3: Requirements Development for Responsive Space Missions

4 METHODOLOGY

The methods used to accomplish project Quicklook are described in this section. The major areas covered are project scope, evaluation criteria, design, and trade study. The processes used to accomplish the tasks are detailed below. Once the project methods are clearly defined, the results are presented in Section 5 of this document.

4.1 Project Scope

The scope for the Quicklook project applies to the requirements analysis and design phases of the TacSat-3 system. Within the design phase, the TSS Team had to ensure that the scope of the project would allow for a successful and meaningful trade study, while simple enough to allow successful project completion within the time and resource constraints. For this reason, a small satellite system was chosen (TacSat-3) and with the guidance from The Aerospace Corporation, the satellite deployment phase was scoped out of the design project. By adequately scoping the Quicklook design efforts, the TSS Team ensured that it satisfied all customer requirements and concerns through early requirements analysis, and scheduled status reports with design reviews.

When scoping the training analysis of the Quicklook project, the TSS Team decided that the training analysis will be limited to the study of the training and engineering hours.

This analysis would in turn provide an assessment of the learning curve involved with learning SysML.

4.2 Evaluation Plan

This section describes the evaluation methodology used to assess the effectiveness of SysML as a modeling language. Additionally, it outlines how the TSS Team used data from individual level-of-efforts to study the learning curve involved with this new modeling language.

4.2.1 Evaluation Objective

The primary objective of this evaluation is to facilitate answering end-user concerns about the effectiveness of SysML. TSS realizes that when a change is introduced into a process within an organization, people within that organization will have to deal with a learning curve. This learning curve is defined here as a drop in productivity as people adjust to new systems, organizations and/or processes; in the case of the TSS Team, the learning curve involves a new systems engineering modeling language.

Introducing SysML was predicted by the TSS Team to have a critical learning curve. TSS's goal for conducting this evaluation is to assess the time it takes for systems engineers who are familiar with system modeling, UML, or Object-Oriented Design (OOD) to model a system using SysML.

4.2.2 SysML Effectiveness Evaluation

In order to evaluate the effectiveness of SysML, the TSS Team analyzed and designed the Tactical Satellite-3 system, an advanced micro satellite system that is planned to be launched in October of 2007. Due to sensitivity of its operation, a greater part of information on Tactical Satellite-3 (TacSat-3) system is still classified and yet to be publicized. Nevertheless, the available literature provided sufficient basis for the Quicklook design project evaluation of SysML.

The TacSat-3 system is dependent on a modular, plug and play capability to carryout its operation while satisfying its short turnaround time requirement. This characteristic of the TacSat-3, as well as the fact that it is a fairly small system, made it an ideal candidate for an exploratory design project.

The Quicklook design project focuses on the operational phase of the TacSat-3 system, while emphasizing on the satellite's main operational capability: 1. Hyperspectral Imagery Operations and 2. Communication Operations.

Based on the background of the individuals within the TSS Team and their level of experience using UML and other systems engineering design methodologies, such as structured analysis, individual team members were encouraged to assume responsibility for designing different sections of the system. An initial hierarchical design approach made it easy to delegate tasks.

The TSS Team followed an iterative evaluation methodology as depicted in Figure 4-1. This process involved:

- Studying and learning the SysML language by means of research and review of available literature. The TSS Team also received professional training, which is discussed in the next section.
- Designing the TacSat-3 system iteratively, using the IBM Rational software and EmbeddedPlus Toolkit, while documenting the modeling software capabilities and limitations at each design iteration.
- Documenting SysML capabilities and limitations based on design activities at each pass at the design model.
- Creating an executable model using SysML as a facilitator to conduct a trade study of design alternatives for the TacSat-3 and documenting related findings.
- Augmenting and monitoring individual training hours collected on a weekly basis, and evaluating the learning curve involved with using SysML based on overall training and engineering hours.

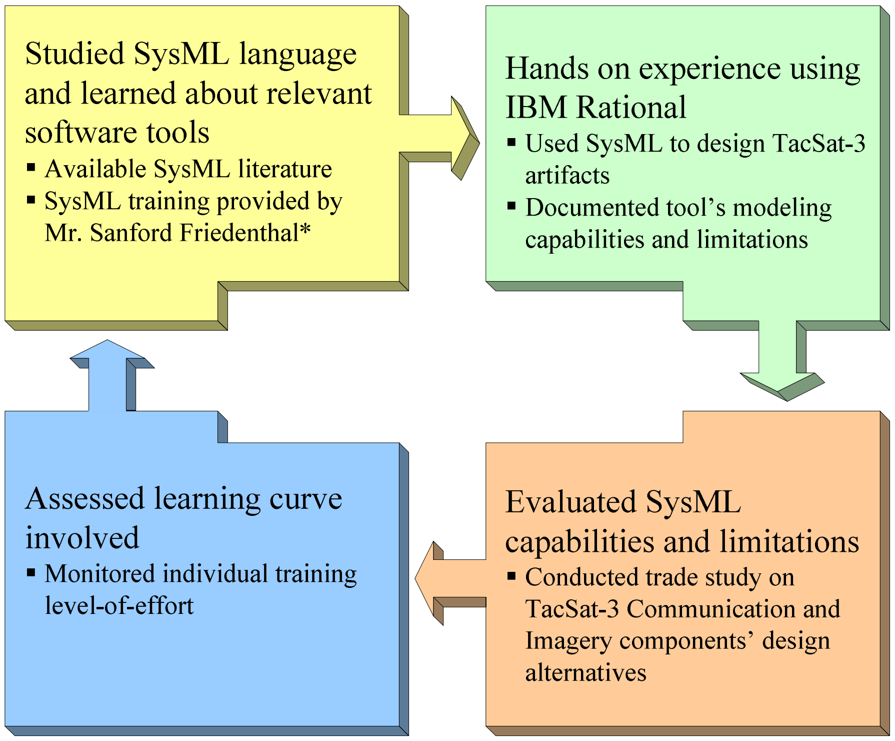

Studied SysML language and learned about relevant software tools
▪ Available SysML literature
▪ SysML training provided by Mr. Sanford Friedenthal*
Hands on experience using IBM Rational
▪ Used SysML to design TacSat-3 artifacts
▪ Documented tool's modeling capabilities and limitations
Evaluated SysML capabilities and limitations
▪ Conducted trade study on TacSat-3 Communication and Imagery components' design alternatives
Assessed learning curve involved
▪ Monitored individual training level-of-effort

Figure 4-1. Evaluation Methodology

4.2.2.1 Instructor Lead Training

Further development of the TacSat-3 design was made possible when team members received an eight-hour training session. The training session was courtesy of Mr. Sanford Friedenthal, Deputy of Corporate Systems Engineering, at the Lockheed Martin Corporation. Mr. Friedenthal is the liaison between INCOSE and the Object Management Group (OMG), and chairs the industry standards effort through the OMG to develop the UML profile for System Engineering. The training session entailed covering the SysML basics, as well as answering students' questions. In addition, the TSS Team received a 3 hours follow-on training session with Mr. Friedenthal. The focus of this session was the importance of model organization and the significance of package diagrams. See section 4.3.4.1.1, which describes packaging, for more details.

4.2.3 SysML Learning Curve

In order for the TSS Team to conduct a study of the team's training and learning efforts, performance criterion had to be specified. Several quantitative factors were identified and collected for the duration of this project. The following data provided the basis for an objective evaluation of training efforts of team members:

- Number of hours spent on SysML literature review
- Number of hours spent on the software tools' literature review (including help files and available documents)
- Instructor-lead training hours
- Total number of hours spent on designing the TacSat-3 system.

Each team member was required to submit his training hours on a weekly basis. Non-training hours were documented as engineering efforts. Training related timesheets were augmented with the previously recorded data. Training data was collected from January 22, 2007 to April 19, 2007.

The TSS Team closely monitored the level of efforts and progress of individual team members throughout the length of this project. It is important to note that the team members have different educational backgrounds and work experiences. The study has cleverly leveraged team members' experiences to address the objective of this study. Refer to section 5.4 (SysML Learning Curve) for more details.

4.3 Design

The design of the TacSat-3 Satellite System was the main vehicle used to evaluate the effectiveness of SysML and the design tools that support it. This section details the approach used by the TSS Team for system design. It is important to note that views of the TacSat-3 design are not included in this document for brevity purposes. The entire design model is attached in a separate document labeled: "Project Quicklook: TacSat-3 System Design Model for the Analysis Phase" (located in the Quicklook Project Notebook tab labeled "Section 2 System Design"). More details on each portion of the system design process is given below.

4.3.1 Design Tools

Successful model-driven design requires both an effective modeling language and tools that implement the language well. Therefore, to evaluate SysML in terms of model-based engineering, it was necessary to use software tools. As mentioned in earlier sections, the software tools used were IBM Rational Systems Developer and the EmbeddedPlus SysML and Simulation toolkits. The specific software tools used in this project were proposed by the project sponsor and are detailed below.

4.3.1.1 IBM Rational Systems Developer

The project sponsor requested IBM's Rational System Tools be used for this project. The TSS Team chose Rational Systems Developer V7.0 because it was less complex than other products in the Rational Tool Suite and was compatible with the EmbeddedPlus SysML Plug-in. Rational Systems Developer is primarily a UML software design tool. The SysML Plug-in extends the tool capabilities to cover SysML as discussed in the following section.

4.3.1.2 Embedded Plus SysML and Simulation Toolkit Plug-ins

The EmbeddedPlus SysML and Simulation Toolkit are add-in products that provide productivity tools, simulation tools, and language support for SysML to IBM's Rational modeling platform. The SysML Toolkit takes the UML capabilities of Rational System Developer and extends them to cover SysML. The Simulation Toolkit allows automatic executable model generation from the SysML model into Java or C++ code.

4.3.2 System Definition

Available information from the Air Force Research Laboratory on the TacSat-3 system was used to develop a concept of operations and originating requirements for the system design. Over several iterations, the scope of the design was narrowed to include only the space vehicle with a focus on the imagery capture and dissemination portion of the orbital operational phase of the system. The level of abstraction of the design was chosen based on the goal of conducting a trade study of design alternatives. The context from the system included the space environment, the space domain, the United States Strategic Command, and International Laws. The external systems in the space domain are the warfighters requesting the imagery, the target, and the controlling ground station.

4.3.3 High-level Design

Before the TSS Team started the design process using SysML, the team members began brainstorming the initial functional and logical design using SysML. Individual ideas for the initial design were unified during a whiteboard session. Following this approach, all team members agreed on the high-level design before beginning the implementation with software tools. Also during this meeting, the team agreed upon an initial organization scheme for the design model in the software tool.

4.3.4 Design Methodology

The extensive literature and institutional knowledge covering design methodology of established systems engineer design methods, such as Structured Analysis, does not exist for the relatively new SysML. During the SysML learning and system design process, the TSS Team developed a design methodology to support the needs of the project. It quickly became apparent that proper organization of the model was vitally important when working on a complex design model in a team environment. SysML lends itself easily to a hierarchical organization, therefore the team's approach to design also followed a hierarchical approach.

4.3.4.1 Design Model Organization

The Quicklook design model contains hundreds of entities in more than forty-five SysML diagrams. The model quickly could become overwhelming without proper organization. The model is organized into layers of abstraction ranging from the highest space domain level down to the component level. Each level of abstraction is self-contained with all the information necessary to define the TacSat-3 system at that level.

4.3.4.1.1 Packaging

One of the key ideas in organizing SysML models is that of the package. The package is a diagram in SysML but it is also a larger concept of organizing all model information. The organization scheme is not standardized by the SysML specification. Packages are organized in whatever fashion best suits the design and the designers. The TacSat-3 design is packaged along functional lines at each level of abstraction. Packages common to every level include structure and behavior. Other packages such as requirements and use cases are only used where necessary to fully define that level of the design.

4.3.4.1.2 Entity Reuse Libraries

During the design effort, the TSS Team sought to use Object-Oriented principles to reduce design efforts. Entities in the design model, which are used multiple times or which are generalization or specialization of other model entities are kept in reuse packages for repeated use without redefinition. The two areas where this technique was used in the TacSat-3 design were the reused reusable component and the input/output libraries. The reused reusable component library was used to define generalizations of similar components such as the imagery processor and control processor. The details of a generic processor were defined once in the reused component library and then the specialized processors were much easier to define. The same concept was used in the input/output library where items that flow between blocks were defined. Control data, for example, was a generalization of various types of control data which share many common attributes.

4.3.4.2 Hierarchical Design Method

In the absence of an established SysML design method, the TSS Team developed a process during the project. The agreed upon organization for the design model drove the process to be hierarchical in nature. The design was fully defined at each level before proceeding to more detailed levels of abstraction. Using this hierarchical process, it was easy to scale the complexity of the design model to fit the purpose of the design.

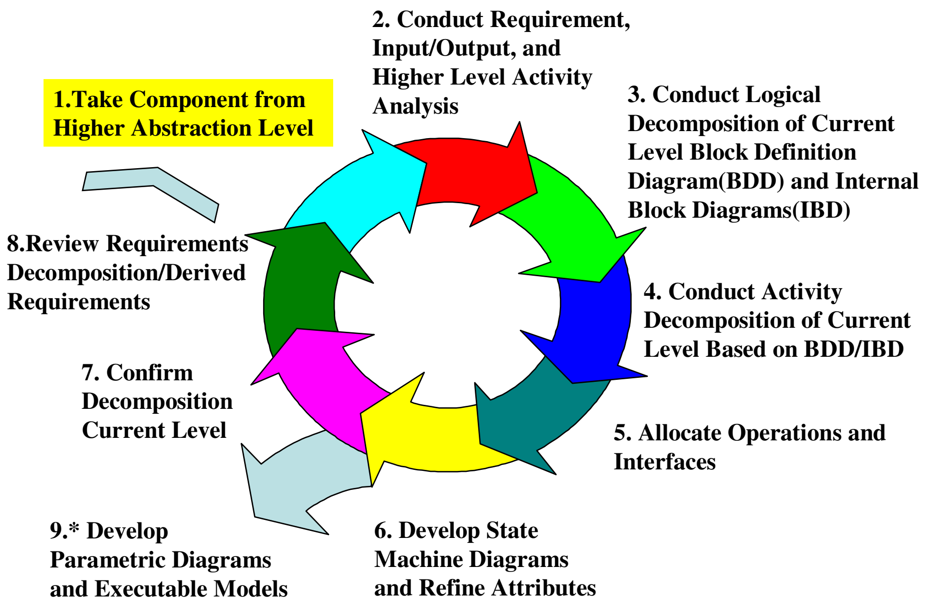

1.Take Component from Higher Abstraction Level
2. Conduct Requirement, Input/Output, and Higher Level Activity Analysis
3. Conduct Logical Decomposition of Current Level Block Definition Diagram(BDD) and Internal Block Diagrams(IBD)
4. Conduct Activity Decomposition of Current Level Based on BDD/IBD
5. Allocate Operations and Interfaces
6. Develop State Machine Diagrams and Refine Attributes
7. Confirm Decomposition Current Level
8.Review Requirements Decomposition/Derived Requirements
9.* Develop Parametric Diagrams and Executable Models

Figure 4-2. The Hierarchical Design Method

During the design phase, the TSS Team began to develop and refine the hierarchical design methodology. In the figure above this method is depicted as a circle because it can be implemented repeatedly for each level of abstraction. Each iteration takes as its input the complete design of the next higher level and produces a complete design for the next level. The steps are not discrete and iteration continuously occurs. The steps are sequential because some information must be defined in the previous step to complete later steps. The steps are explained below (See Appendix A, for definition of SysML terms):

1. Take the component from the higher level Block Definition Diagram (BDD) you wish to define further.
2. Analysis of the requirements, external input/outputs, and higher-level activity diagrams provides information needed to make a decision on logical decomposition of the new level.
3. Using the information from the higher level behavior diagrams (which gives you the activities of the block) and the Internal Block Diagram (IBD) (which gives you your external inputs/outputs), develop a logical decomposition of the current level into its components on a new BDD and IBD.
4. In accordance with the higher-level behavior diagrams, develop the current level behavior diagrams using the logical components you defined in the previous step as your swimlanes/lifelines/actors.
5. Activities of the swimlanes/lifelines become operations for the associated blocks. Information/control exchanges between lower level blocks become the interfaces and flows in the IBD.

6. Using higher-level state machine and/or activity diagrams develop state machine diagrams.
7. During this process of mapping activity diagrams onto the current level of decomposition, you will change the logical decomposition or confirm that your initial logical decomposition will meet your requirements. Once this is successfully completed, your logical design becomes your physical design.
8. Once you have defined this level of decomposition then confirm you have satisfied all requirements and document new derived requirements.
9. Parametric diagram and executable models are not developed within any specific level of abstraction. After the model is complete parametric diagrams are defined to constraint attribute the will be used to define the analytical foundation for executable models.

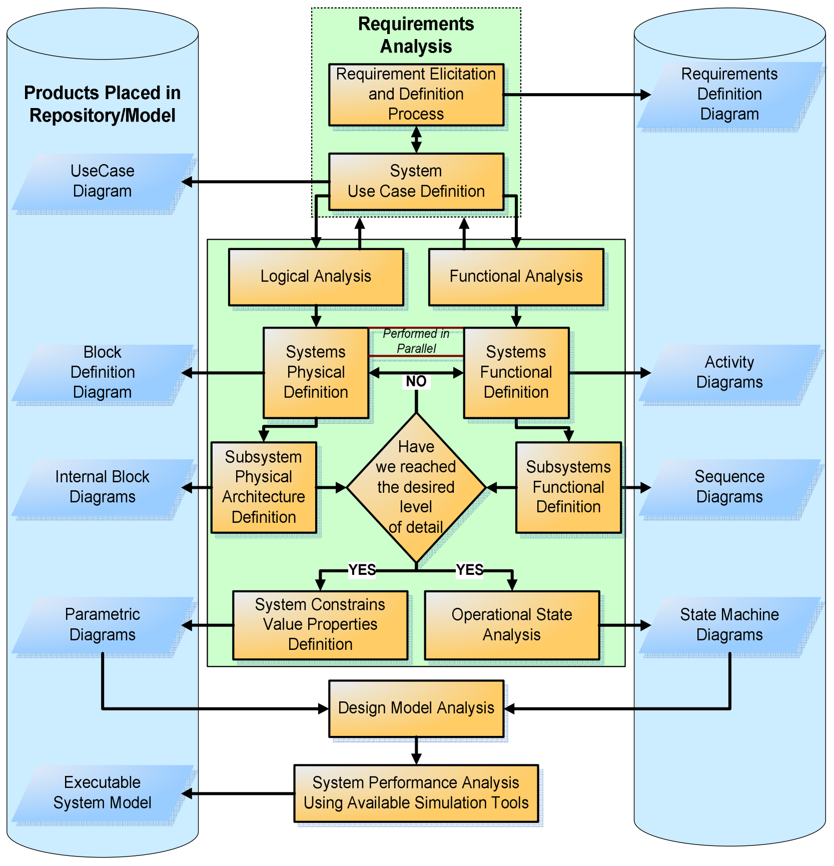

Products Placed in Repository/Model
Requirements Analysis
Requirement Elicitation and Definition Process
Requirements Definition Diagram
UseCase Diagram
System Use Case Definition
Logical Analysis
Functional Analysis
Performed in Parallel
Block Definition Diagram
Systems Physical Definition
NO
Systems Functional Definition
Activity Diagrams
Internal Block Diagrams
Subsystem Physical Architecture Definition
Have we reached the desired level of detail
Subsystems Functional Definition
Sequence Diagrams
YES
YES
Parametric Diagrams
System Constrains Value Properties Definition
Operational State Analysis
State Machine Diagrams
Design Model Analysis
Executable System Model
System Performance Analysis Using Available Simulation Tools

Figure 4-3. Design Process Overview

Figure 4-3 shows the same process from a different point of view. This view shows the entire design process with SysML diagram outputs at each stage shown on the left and right sides. The iterative process that takes place at each abstraction level as described previously is shown with the decision loop at the center of this diagram. Once each level is completed, it is determined whether this level of design detail answers the goal of the program, and if any further levels of detail will be designed (as indicated by the "NO" decision branch).

4.4 Executable Model

The Object-Oriented nature of SysML facilitates the creation of an executable model based on the system design. As a result, SysML tools can easily convert a SysML design to an executable model. The EmbeddedPlus Simulation toolkit for the Rational Systems Developer tool allowed the TSS Team to create an executable model within the Quicklook SysML framework. This was accomplished by adding pieces of custom Java code to the Block Operations, State Operations, and State Transitions of the Quicklook SysML design. The Simulation toolkit then automatically generated the code required to run the executable model based on the SysML objects and the custom Java code. The diagram in Figure 4-4 illustrates how the executable model is created within the SysML framework.

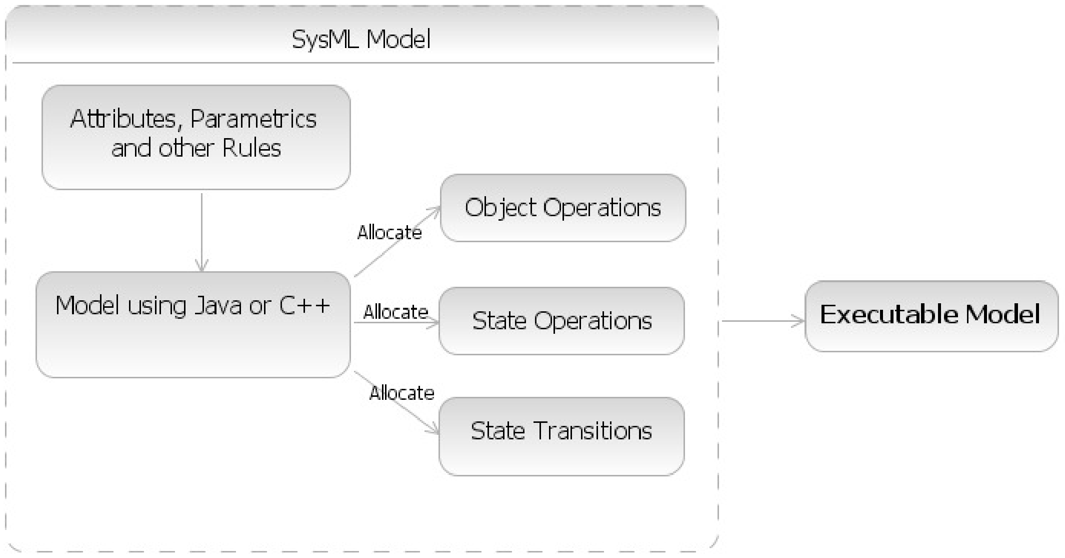


Figure 4-4. Creating the Executable Model

The following output methods were used to capture the executable model results:

- Text in the console – The easiest way to output the executable model results is by outputting text in the Rational System Developer's console. This feature can be implemented directly within the SysML design so the user only needs to run the model to see the output.
- Sequence diagrams – The sequence diagrams generated from the executable model can be compared to the sequence diagrams created during the design to ensure the system is behaving as expected.

4.5 Trade-study of Design Alternatives

In order to perform a trade study of design alternatives, the following methodology was derived.

- The design was scoped to focus on the imagery capture and processing, as well as the communication components.
- The trade study was used to explore variations of morphological components to find the design alternatives that best met the system requirements.
- The morphological box components were chosen to represent realistic design choices; components that are more expensive have a higher performance level.
- An executable model was created to allow the trade study to be conducted based on information in the design model without having to create a separate analytical model.

In order to evaluate the effectiveness of performing trade studies of design alternatives using SysML, the TSS Team proposed to create an executable model of the TacSat-3 SysML design. The executable model of TacSat-3 system was developed using the EmbeddedPlus Simulation Toolkit plug-in for the Rational Systems Developer tool. The resulting executable model was used to analyze the communications and image processing components of the TacSat-3 system.

5 RESULTS

This section provides the results of the TSS evaluation of SysML's modeling capabilities, and the learning curve associated with using SysML and related software tools to design a system comparable in size with the TacSat-3 design efforts.

5.1 SysML Evaluation Overview

SysML was specifically designed to address systems engineers' needs for a domain specific modeling language that reduces the document centric approach to design and modeling of complex systems and system of systems. SysML addresses the needs of systems engineers in a similar manner as UML satisfies the modeling needs of software engineers and software intensive projects.

SysML is a profile, or a subset of UML with some additional semantics to tackle the age-old requirements-to-design traceability. It also constrains design artifacts through the use of the parametric diagrams for behavioral analysis.

As a profile of UML, SysML is much smaller in size and requires less time for the user to become familiar with its semantics. Because of its small size and explicit semantics, and the similarities to UML, systems and software engineers alike can become skilled in using this modeling language in a short period of time. The TSS Team spent approximately 300 hours learning the language and the related IBM Rational and EmbeddedPlus Toolkit products. This translates to approximately five hours per week, per group member. This is not to say that SysML is simplistic, but rather easy to learn if the user has had exposure to Model-Based Systems Engineering and modeling using UML.

In addition to it being easy to learn, SysML is very easy to tailor to specific design and modeling needs. Due to its “block” components’ unique method of abstraction, this language can be used to model various systems, even those specific to their particular domain (See 5.1.2 SysML Versatility).

5.1.1 Object-Oriented Modeling with SysML

SysML, like UML, allows engineers to use Object-Oriented concepts to model their systems. Modeling with SysML using OO concepts can reduce modeling efforts through reusability of blocks and eliminating redundancy in the design model. Using OO also makes the design model independent of the development language (e.g. C++, JAVA).

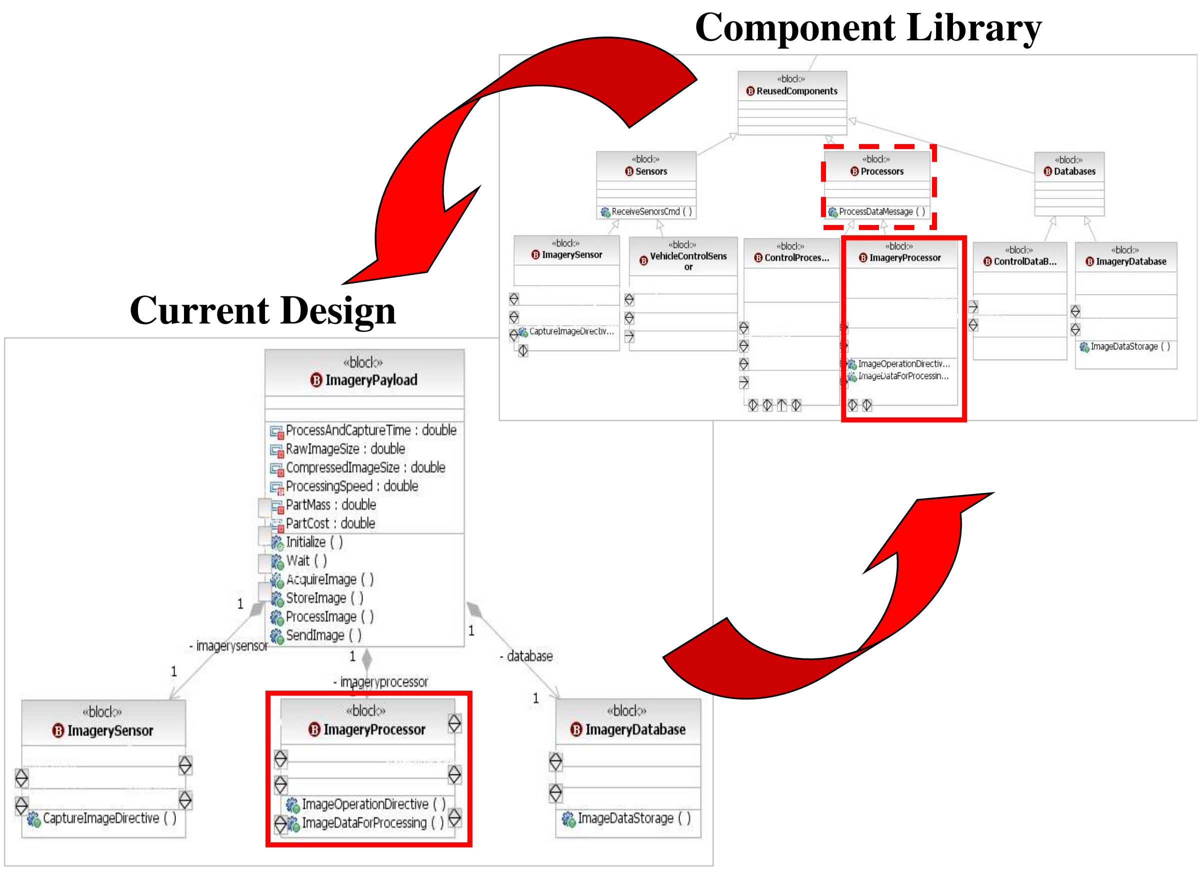


Figure 5-1. Use of Object-Oriented Reuse Libraries

A specific example of an Object-Oriented modeling technique, used in the TacSat-3 design model, is shown in Figure 5-1. During the logical decomposition of the imagery payload block, it was determined that a Processor block would be needed to process imagery and command data. The design team also realized that a processor would also be necessary in the vehicle block. A generalized Processor block was defined in the component library. To further define the processor, the Processor block was decomposed into specialized processor blocks. Using the generalization notation, both Imagery and Control processors inherited the attributes and operations of the general processor block. Now each processor can be used for their particular application (i.e. Imagery and Vehicle). In this way, less design effort was required for the specification of the processors.

5.1.2 SysML Versatility

Each artifact or diagram in SysML has a clear purpose. Structure, Behavior and Requirement diagrams can completely define a system. Since none of these diagrams are domain specific, SysML, unlike UML, can be used to design a variety of systems, at various levels of detail. Even domain specific systems, such as satellite systems, communications system or mechanical systems, can be modeled using SysML.

Not only SysML can support various systems, it also allows engineers to design their systems to the right level of abstraction. For example, a futuristic satellite system might only need a top-level design definition, however, a mechanical system may need every part accounted for before handing off the model to the manufacturers.

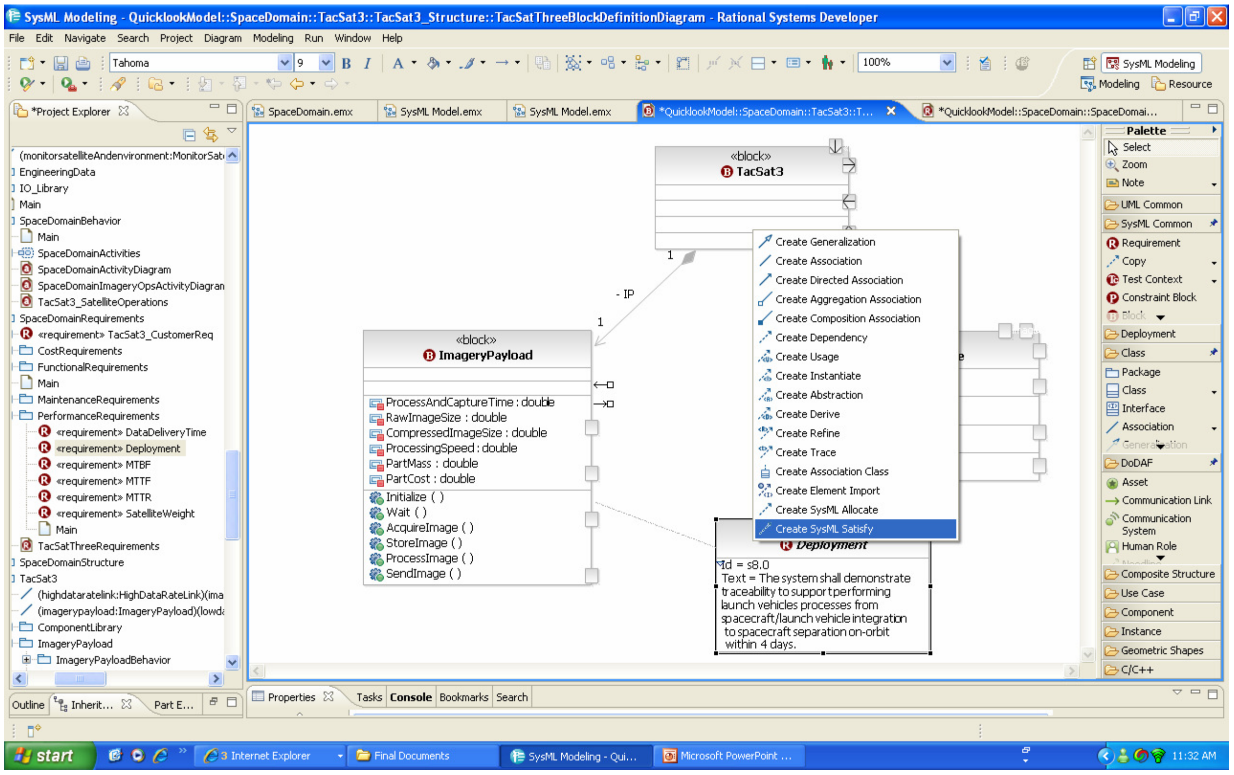
5.1.3 Verification and Validation

Conventional methods of systems engineering involve vigorous requirements engineering and proper documentation of systems requirements. However, requirement intensive engineering is bound to introduce ambiguity into the system design through incomplete, unclear, and misworded requirements.

SysML introduces two additional diagrams to address issues with traceability and systems verification and validation process: Requirements diagrams help capture the requirements and Parametric diagrams help constrain the design model. In addition, SysML semantics allow for bi-directional traceability between requirements and the model. SysML enables the system engineer to both textually and graphically capture the requirements, while enabling the linkage of these requirements to other model elements. This further elaborates the design through tracing requirements to test scenarios or implementation procedures.

5.1.3.1 Requirements Diagrams

Using Requirements diagrams to represent systems requirements, designers can see the upstream and downstream impact of requirements on any level of abstraction in the design process. A system’s requirements are graphically represented in a flexible, adaptable format that works with the user’s individual process. Changes to requirements or system model changes are less difficult to deal with because the requirements are traced to the system model. SysML provides the notation necessary to trace requirements and establish the relationship of each requirement to the design model. (i.e. *trace*, *satisfy*, and *verify*). Requirements traceability to the design model allows for the automation of verification through testable models bound by mathematical equations through the use of parametric diagrams.



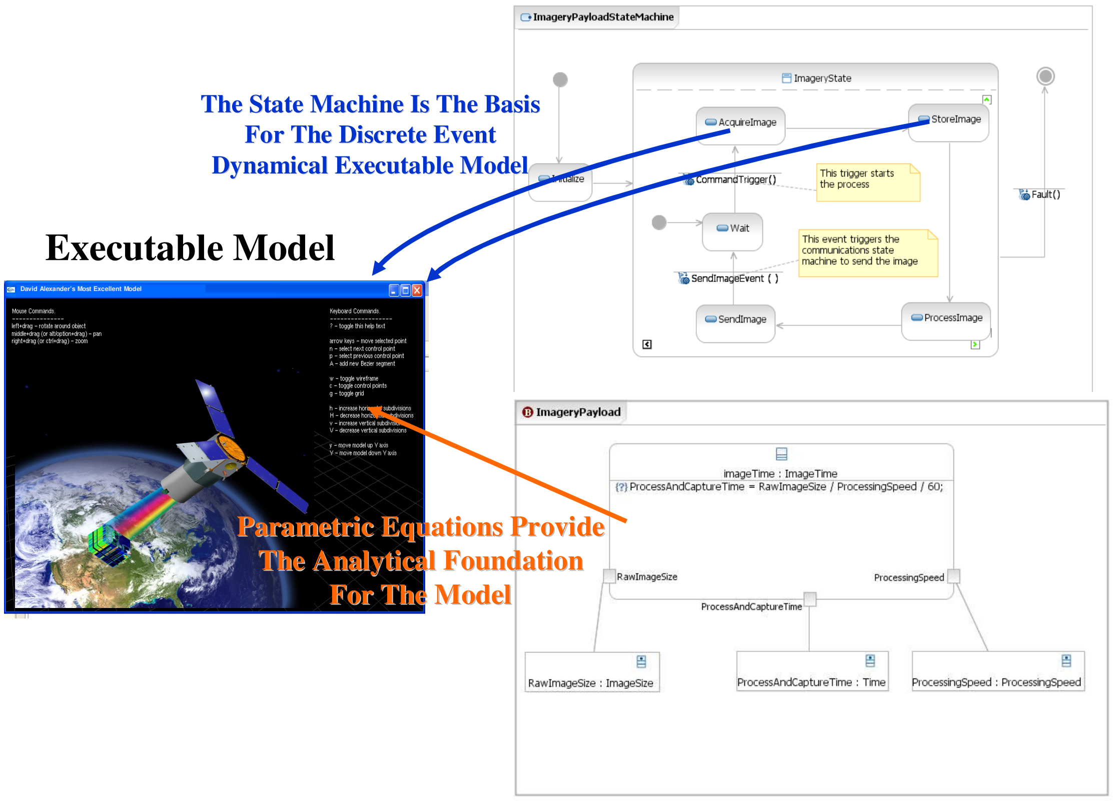
Id = s8.0
Text = The system shall demonstrate traceability to support performing launch vehicles processes from spacecraft/launch vehicle integration to spacecraft separation on-orbit within 4 days.

Figure 5-2. Allocation of a Requirement to a Block

An example of how a requirement was defined to be satisfied by a specific block in the TacSat-3 design model is shown in Figure 5-2. Once all requirements were related to the components of the design model, it became easy to perform an analysis of how well the design satisfied the requirements. Queries could be used to determine the requirement(s) satisfied by a block and conversely, all the blocks traced to any requirement. From this analysis, it was easy to determine which requirements were not met and how design or requirement changes would affect each other. The ability to conduct this analysis within the design model without having to constantly reference a requirements document is a significant advantage of model-based systems engineering, which SysML has enhanced by providing the necessary notation for Requirements Diagramming.

5.1.3.2 Parametric Diagrams

Parametric diagrams provide a method of introducing mathematical equations and constraints to bind the design model. Combination of traceable requirements and executable parametric models replaces the need for document intensive requirements engineering. SysML delivers a method of using executable specifications to verify system design through proper use of its requirements and parametric notation. This allows systems engineers to build testable system models, which allow for validating system behavior against the requirements. In addition, early introduction of executable models into the design leaves less room for errors to be encountered down the road, which translates to saving time and cost on new system design, development and testing.


The State Machine Is The Basis For The Discrete Event Dynamical Executable Model
Executable Model
David Alexander's Most Excellent Model
Parametric Equations Provide The Analytical Foundation For The Model
ImageryPayloadStateMachine
ImageryState
Initialize
AcquireImage
StoreImage
CommandTrigger()
This trigger starts the process
Fault()
Wait
This event triggers the communications state machine to send the image
SendImageEvent ( )
SendImage
ProcessImage
ImageryPayload
imageTime : ImageTime
{?} ProcessAndCaptureTime = RawImageSize / ProcessingSpeed / 60;
RawImageSize
ProcessingSpeed
ProcessAndCaptureTime
RawImageSize : ImageSize
ProcessAndCaptureTime : Time
ProcessingSpeed : ProcessingSpeed

```
Mouse Commands.
----------------
left+drag - rotate around object
middle+drag (or alt/option+drag) - pan
right+drag (or ctrl+drag) - zoom

Keyboard Commands.
------------------
? - toggle this help text

arrow keys - move selected point
n - select next control point
p - select previous control point
A - add new Bezier segment

w - toggle wireframe
c - toggle control points
g - toggle grid

h - increase horizontal subdivisions
H - decrease horizontal subdivisions
v - increase vertical subdivisions
V - decrease vertical subdivisions

y - move model up Y axis
Y - move model down Y axis
```

Figure 5-3. State Machine and Parametric Diagrams Relations to the Executable Model

The relationships between the state machine diagram, parametric diagram, and the executable model of the TacSat-3 design are shown in Figure 5-3. An important aspect of model-based engineering is the linkage between the definitions of system behavior in the design model and the resulting behavior realized in the executable model.

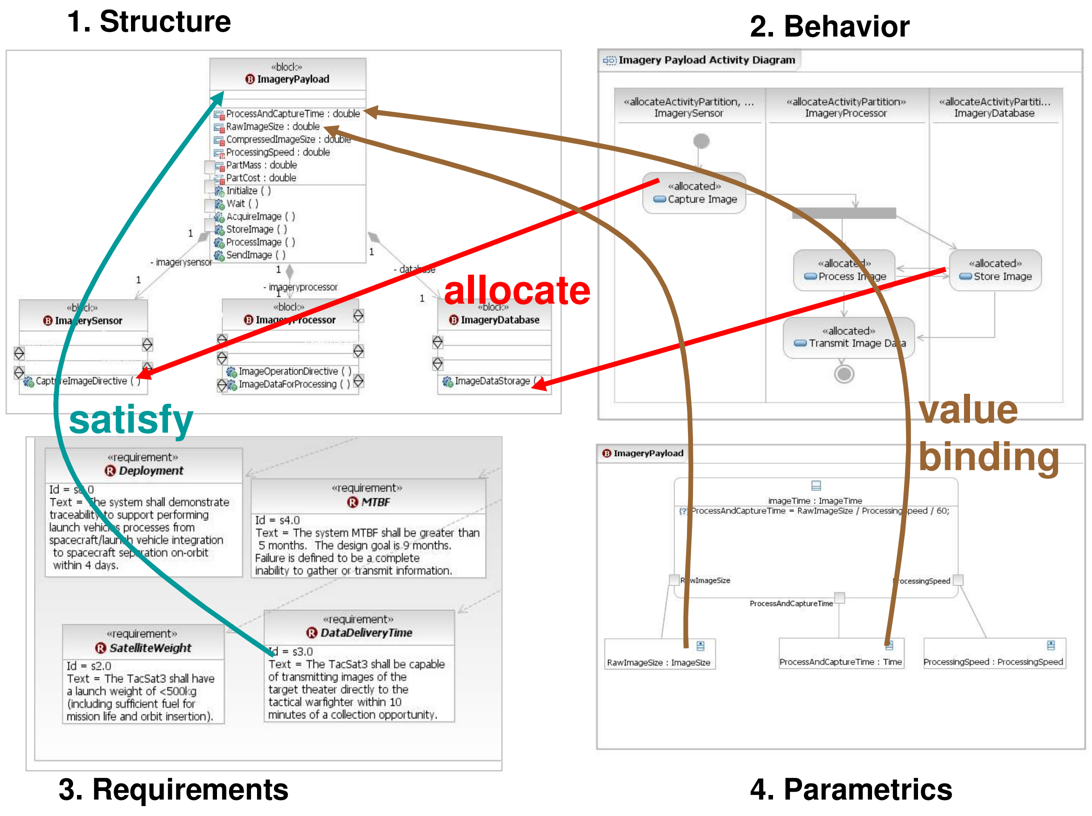
The software tools used in this project allowed traceability between the design model and the executable model. This linkage gives model-based engineering the huge advantage of having design changes reflect in the executable model. The affects of design changes can then quickly be determined without extensive executable model rework or having to rely on costly prototypes. This will reduce design cost because of a reduction in errors, modeling delays, and redesign efforts.

5.1.4 Comments and Findings

Model-based systems engineering, and modeling using SysML can benefit from advancements in enabling technologies, such as well developed software packages that fully support SysML, while integrating essential software like Requirements Management and Configuration Management. The TSS Team used Rational Systems Developer (RSD) v7 for the Quicklook project. The tool allowed the team to create a Unified Data Dictionary (UDD), which served as the design model for the TacSat-3 system.

5.1.4.1 Unified Data Dictionary

The Unified Data Dictionary is important in that it has to be unambiguous so that everyone can understand how the design pieces are categorized, and how the model is organized. A UDD combined with an advanced software tool, could potentially ease the translation of the design model into executable models. Creation of the UDD was orchestrated with Mr. Friedenthal's direction, using SysML's Package diagrams.



*The concept of this graphic is based on slide 65 of the OMG Systems Modeling Language (OMG SysML) Tutorial, 11 July 2006

Figure 5-4*. Design Concordance Facilitated by Model-based Engineering

Figure 5-4 shows the concordance between entities in the TacSat-3 design model that was facilitated by model-based systems engineering in SysML. The unified data dictionary will ensure this concordance if the semantics of SysML are followed during the system design. This is huge improvement over previous design methods where concordance between design model views could only to be ensured by tedious manual crosschecking.

5.1.4.2 Package Diagrams

Package diagrams provide a method for systems engineers to organize the design model. This becomes exponentially important when dealing with a design model that is used by a large number of users. By fragmenting the design model into a logical organization, users can select to work only on their piece. This translates to cost savings and a well-established CM plan will help the integrity of the model.

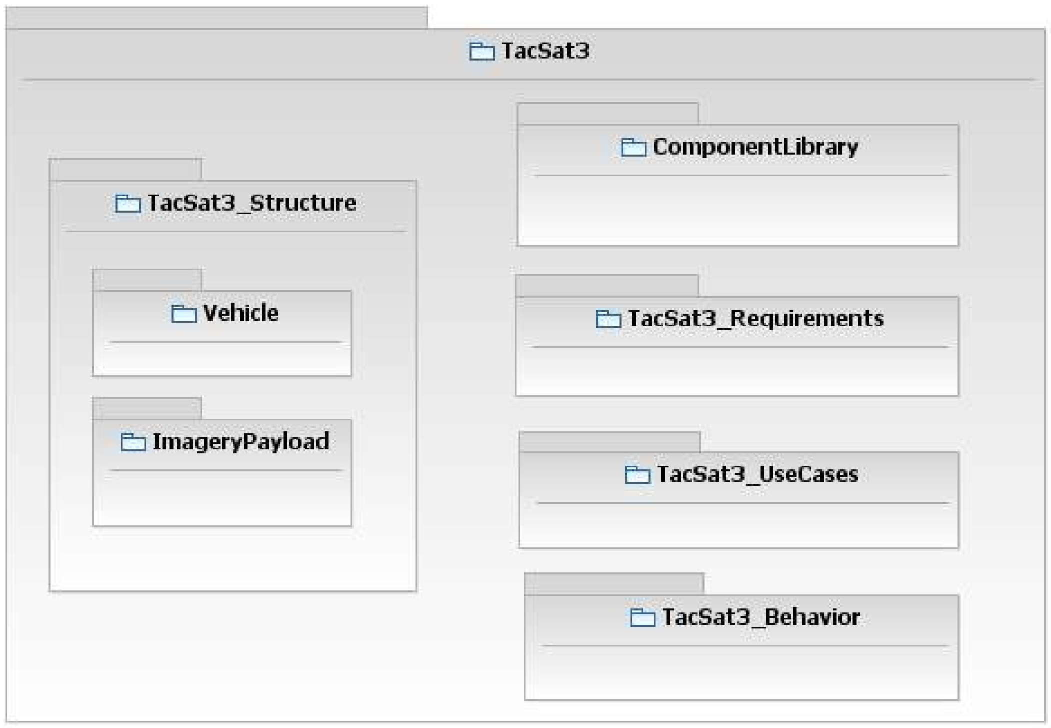


Figure 5-5. Package Diagram for the TacSat-3 Model

How package diagrams were used to create logical organizations in the design model of TacSat-3 is demonstrated in Figure 5-5. The TacSat-3 design model contains over six hundred and thirty entities and over forty-five diagrams. Effective organization was essential for team members to divide work and to communicate aspects of the design to each other. The more complexity that was introduced into the design the more organization became essential for managing the design model.

5.2 Executable Model Evaluation Overview

The power behind integrating an executable model within the SysML design is the direct traceability back to SysML elements. Since the code is automatically created from the SysML model, design and logical flaws can be traced directly back to the SysML elements that were used to generate the code. The TSS Team found that the resulting executable model is a powerful tool for system performing behavior analyses, state trace analyses, and trade studies of design alternatives, as will be illustrated in the following sections.

5.2.1 Behavior Analysis Results

The executable model allowed TSS to analyze the system behavior and determine any logical errors contained within the SysML model. By specifying various input parameters, the model could be run and the output analyzed to determine if the system is behaving as expected. If any logical errors are found, the SysML elements can be identified and corrected. Figure 5-6 shows the results of a behavior analysis from the executable model in the form of a sequence diagram.

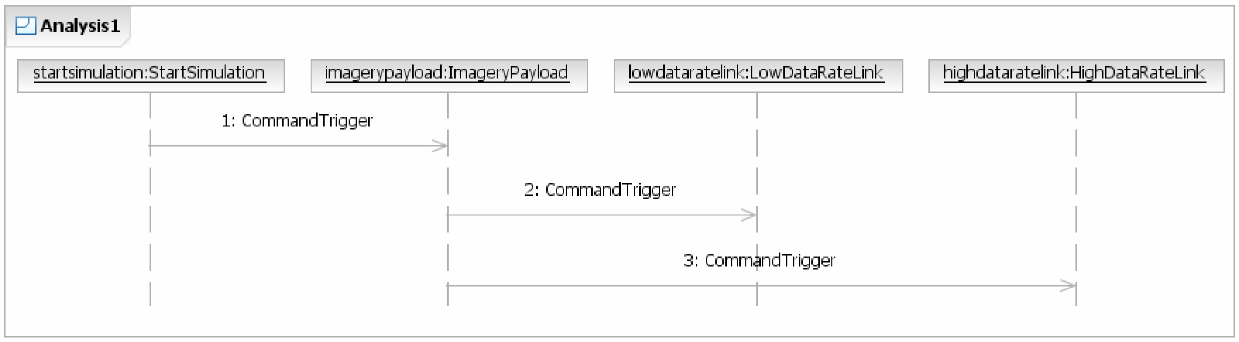
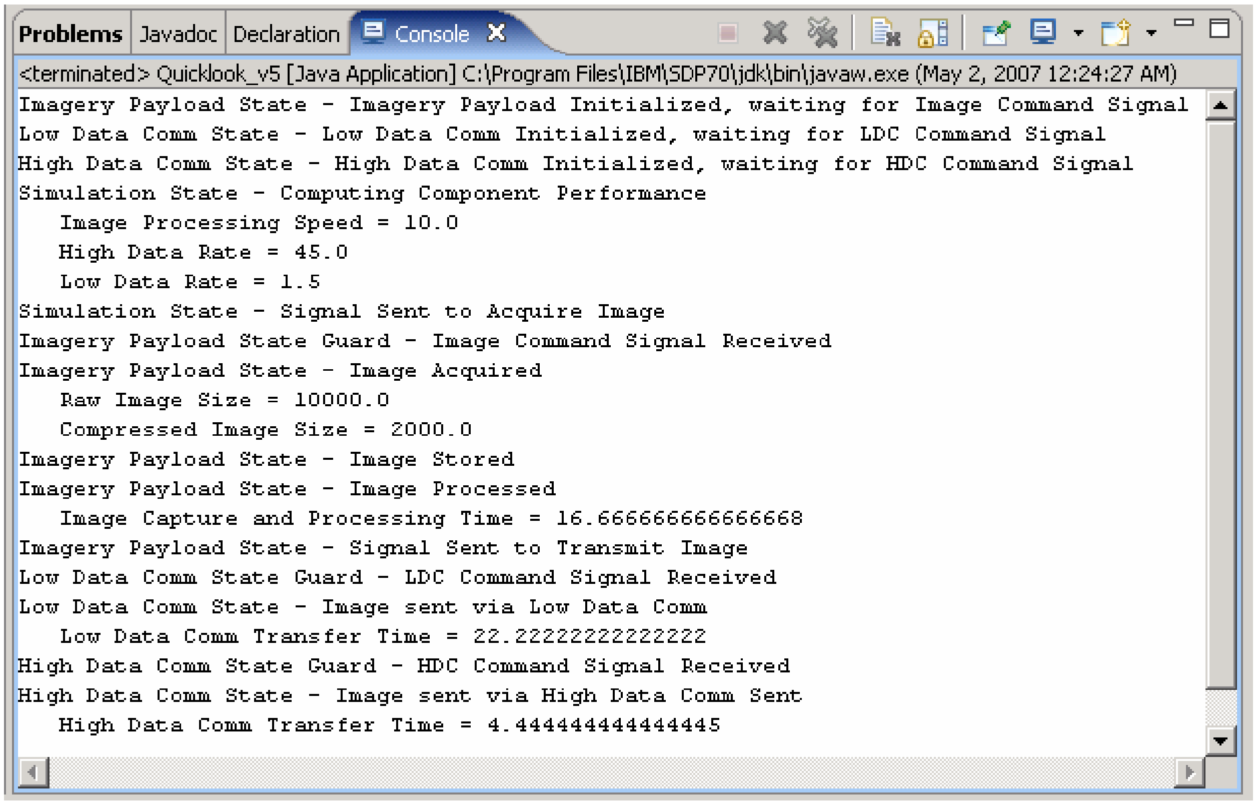

Analysis1
startsimulation:StartSimulation
imagerypayload:ImageryPayload
lowdataratelink:LowDataRateLink
highdataratelink:HighDataRateLink
1: CommandTrigger
2: CommandTrigger
3: CommandTrigger

Figure 5-6. Sequence Diagram

This sequence diagram shows the interaction between system components as the model executed. This automatically generated sequence diagram can then be compared with the equivalent Quicklook sequence diagram and associated requirements to ensure the system is behaving as expected.

5.2.2 State Trace Results

By running the executable model using various input parameters, TSS was able to run a state trace analysis of the system. A state trace analysis helps verify that the system functions as expected. It also helps determine if there are any potential deadlocks in the system. If deadlocks are found within the model, the offending SysML elements can be identified and additional rules added to prevent the deadlock from occurring. Figure 5-7 shows the results of a state trace analysis from the executable model.



```
<terminated> Quicklook_v5 [Java Application] C:\Program Files\IBM\SDP70\jdk\bin\javaw.exe (May 2, 2007 12:24:27 AM)
Imagery Payload State - Imagery Payload Initialized, waiting for Image Command Signal
Low Data Comm State - Low Data Comm Initialized, waiting for LDC Command Signal
High Data Comm State - High Data Comm Initialized, waiting for HDC Command Signal
Simulation State - Computing Component Performance
   Image Processing Speed = 10.0
   High Data Rate = 45.0
   Low Data Rate = 1.5
Simulation State - Signal Sent to Acquire Image
Imagery Payload State Guard - Image Command Signal Received
Imagery Payload State - Image Acquired
   Raw Image Size = 10000.0
   Compressed Image Size = 2000.0
Imagery Payload State - Image Stored
Imagery Payload State - Image Processed
   Image Capture and Processing Time = 16.666666666666668
Imagery Payload State - Signal Sent to Transmit Image
Low Data Comm State Guard - LDC Command Signal Received
Low Data Comm State - Image sent via Low Data Comm
   Low Data Comm Transfer Time = 22.2222222222222
High Data Comm State Guard - HDC Command Signal Received
High Data Comm State - Image sent via High Data Comm Sent
   High Data Comm Transfer Time = 4.444444444444445
```

Figure 5-7. State Trace Analysis

This output console shows the block states and state guards that are triggered as the model executes, as well as the state values of the associated states. This generated data can then be compared with the equivalent Quicklook state machine diagrams and associated requirements to ensure the system is functioning and performing as expected.

5.2.3 Trade Study of Design Alternatives Results

One of the major benefits of the executable model was the ability for the TSS Team to do a trade study of design alternatives. The ability to easily change the input parameters for cost and performance allowed TSS to determine the overall cost required to meet the performance requirements of system components. Using the resulting data from the state trace analysis, a table illustrating design alternatives and their resulting cost and performance results can be easily constructed. Figure 5-8 shows the results of a trade study of design alternatives from the executable model.

Image Results: Raw Image Size = 10,000 Mb Compressed Image Size = 2,000 Mb

🟨 = Exceeds Requirements

Cost and Performance Results → / ↓ Design Alternative	Image Capture and Processing Time (min)	Raw Image High Data Rate Transfer Time (min)	Compressed Image High Data Rate Transfer Time (min)	Compressed Image Low Data Rate Transfer Time (min)	Total High Data Rate Transfer Time (min)	Total Low Data Rate Transfer Time (min)	Total Mass (kg)	Total Cost (millions $)
IPS = 10 LDRB = 1.5 HDRB = 45	16.7	3.7	0.7	22.2	🟨 21.1	🟨 38.9	30.0	1.5
IPS = 50 LDRB = 3 HDRB = 137	3.3	1.2	0.2	11.1	4.8	🟨 14.4	40.0	3.0
IPS = 100 LDRB = 10 HDRB = 234	1.7	0.7	0.1	3.3	2.5	5.0	50.0	4.5

IPS = Image Processing Speed
LDRB = Low Data Rate Bandwidth
HDRB = High Data Rate Bandwidth

Figure 5-8. Trade Study of Design Alternatives

This table effective demonstrates the results of the trade study of design alternatives using data generated by the executable model. Since the executable model is integrated within the system design, various trade studies can be easily performed throughout the design process of the system.

5.2.4 Executable Model Recommendations

While SysML is a standard modeling language, there is currently no standard for creating executable models based on a SysML design. Each SysML tool may handle the executable model development differently, so it is important to understand the limitations of the tool used to create the executable model. It is also important to have plans for the executable model before the SysML design begins. For example, proper naming must be followed for the executable model to compile. Since the executable model is rule driven, some programming experience will be required. Therefore, it is important to have someone with programming experience during the design phase.

5.3 SysML Enabling Software

While it is important to evaluate the capabilities of SysML as a modeling language, it is also important to evaluate the tools used to create the SysML design. The section reviews the results of the design software used for the Quicklook project.

5.3.1 General Findings

The IBM Rational Systems Developer (RSD) with the EmbeddedPlus SysML Toolkit allowed team members to have a unified repository for the design artifacts, and considerably accelerated the ongoing design efforts. The integration process for incorporating individual design pieces into a unified model took a considerable amount of time. This is partially due to unfamiliarity with the software packages' capabilities, failure to establish a naming convention, and a lack of appreciation for upfront creation of a reusable library. Overall, this tool provided the TSS Team with an effective means for managing the Quicklook design.

5.3.2 EmbeddedPlus SysML Toolkit

The TSS Team used the EmbeddedPlus SysML Toolkit to create the Quicklook SysML design. This toolkit effectively addresses engineers' needs for modeling a system in SysML effectively. There were no limitations found based on TSS modeling efforts; the TSS Team was able to design all relevant SysML diagrams using this toolkit.

5.3.3 EmbeddedPlus Simulation Toolkit

The TSS Team used the EmbeddedPlus Simulation Toolkit to create an executable model from the Quicklook SysML model. While this toolkit effectively addresses engineers' needs for creating an executable model using an existing SysML model, the TSS Team did note some limitations:

- While UML2 ports are supported, SysML flowports are not (this limitation was documented). Since the Quicklook design uses flowports, a work around was devised by creating associations between blocks and using the Java SendEvent command to trigger state transitions.
- Parametric Constraint Blocks cause Java compile errors (this limitation was not documented). Since the Quicklook design uses parametric diagrams with constraint blocks, a work around was devised by placing the parametric diagrams in a separate model under the Quicklook project space.

During the process of creating the executable model, the TSS Team documented some possible enhancements that would make the Simulation toolkit even more powerful. The following enhancements were communicated to the EmbeddedPlus engineering team:

- Use parametric diagrams to automatically generate code for the executable model. Since parametric diagrams are a powerful tool for adding engineering analysis to the system design, incorporating parametric equations into the executable model will greatly enhance system traceability and would eliminate the need to write custom code to constrain the necessary block attributes.
- Add a feature that allows the executable model to be traced and debugged from the SysML model itself. Having the ability to trace and debug from the SysML model instead of the Java code would be a great benefit for analyzing and debugging the system.

5.4 SysML Learning Curve

The TSS Team, for the purpose of this evaluation, have been divided into two groups. Group A, are the members with more extensive UML background and group B members have had limited exposure to UML modeling. This is a realistic scenario, since not all systems engineers, or those who will be doing systems engineering as professionals, have been through a formal systems engineering program and therefore not exposed to basic UML knowledge. In addition, those with an extensive software engineering background are much more familiar with UML. Diversity of group members' experiences was used to answer stakeholders' concerns about differences in learning curve for systems engineers who are familiar with Object-Oriented design and UML modeling and those who are used to modeling functional breakdowns. For more details about the group members' individual background and work experience, please refer to Quicklook's Evaluation Plan.

5.4.1 Training Results

The TSS Team spent approximately 250 hours on training and learning the processes required to reach a level of competency in which they were able to produce acceptable results. This is the total number of hours spent on learning SysML and the Rational Systems Developer software. Various training and learning methods have been involved, which have been outlined in the Quicklook Evaluation Plan.

The total number of hours spent training for this project, provides a baseline for future teams, which plan to take on a similar size design project with or without UML background. As the results of our evaluation indicates, Group A, which are those group members with previous UML experience, took about 34% less time than Group B members, who had little or no UML knowledge.

5.4.1.1 Professional Training

TSS understands that not every group will have the opportunity to receive professional training, so the number of hours spent on professional training has been excluded from the total training hours. It is highly recommended however, for groups of any size to receive some kind of instructional training.

5.4.1.2 Assessment of Learning Curve

The learning curve for modeling in SysML has been evaluated based on the level-of-effort of individual team members, and based on their level of experience with UML.

Figure 5-9 depicts the cumulative weekly training hours for the entire team. As can be seen, the slope associate with the number of hours spent is rather small in the first four weeks of the project. The steepest slope, peaked during the period of time from 2/22/07 – 03/14/07 (area between the dotted lines). This period coincides with the team receiving professional training session, becoming familiar with the RSD software and picking the pace with design efforts.

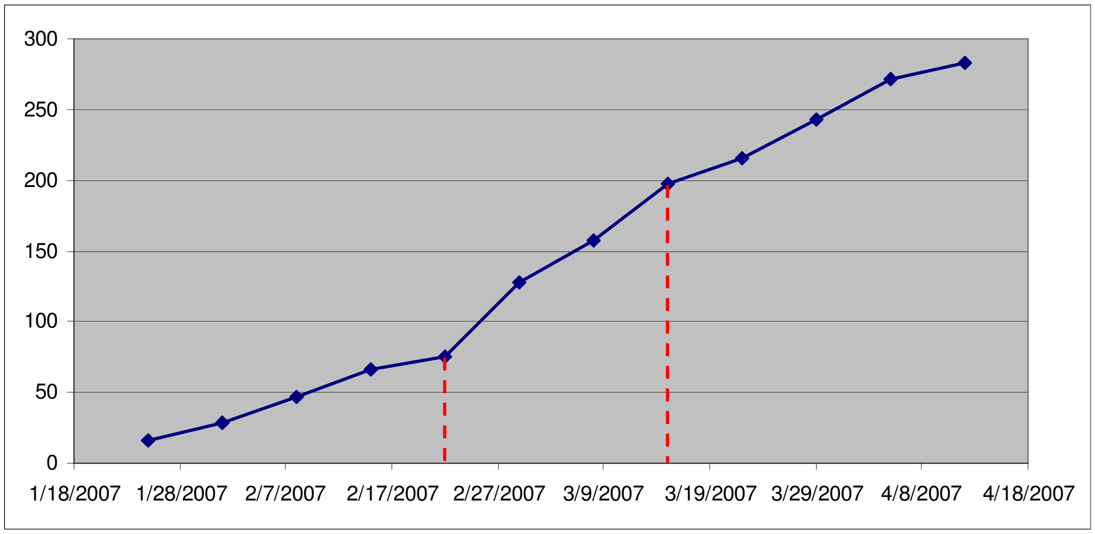


Figure 5-9. Level of Competency

The TSS Team found the comparison between the training hours spent by each group to be of interest since it could provide feedback as far as showing the difference in learning curves given previous UML knowledge and involvement. Figure 5-10 shows a comparison of the training hours spent by each team given the span of the project. As seen, group B spent more hours during the design phases of the project since lack of previous UML knowledge meant having to conduct more research and ultimately a steeper learning curve.

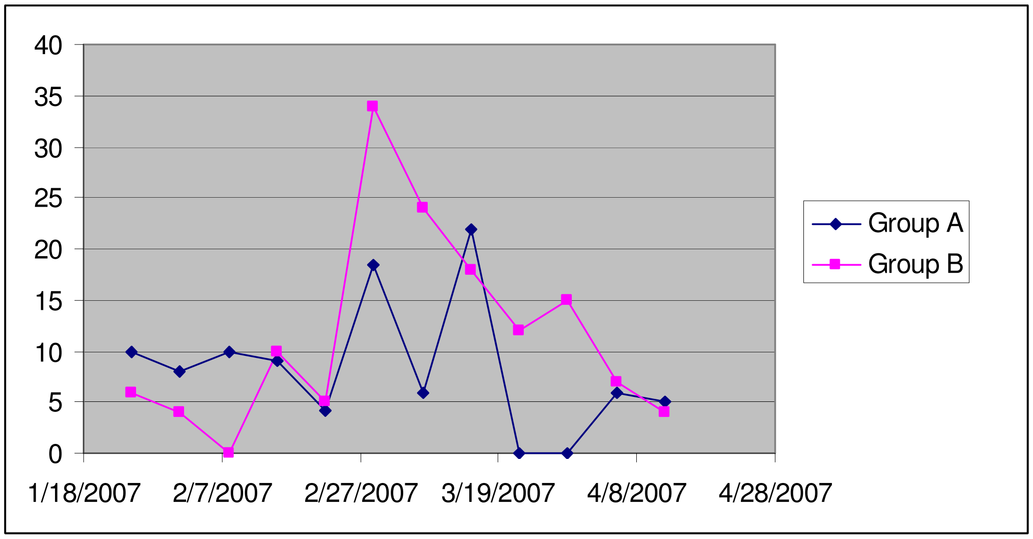


Figure 5-10. Team A and B's Training Level-of-effort

Group A spent approximately 150 hours on SysML and RDS with EmbeddedPlus Toolkit, while Group B spent approximately 100 hours.

Training hours related to executable modeling have been separated from the rest of training hours, since it requires some software programming knowledge, which not all team members possessed. In addition, training hours related to executable modeling, using EmbeddedPlus Simulation Toolkit has been kept apart from this study since it requires some software programming knowledge. The executable modeling efforts in Simulation Toolkit took 30 hours for a single TSS Team member to complete. Prior to the executable model, the individual spent 10 hours reviewing available literature on Simulation Toolkit. To learn more about this, please refer to the Quicklook Evaluation Plan.

5.5 Lessons Learned

With traditional systems engineering development, the requirements and specifications are often document-based, which can be incomplete, ambiguous, and easily misunderstood. Model-Based Systems Engineering allows system architects to build executable models using the finalized system design, along with a defined set of mathematical models and formulas, much easier than producing the traditional and ambiguous text documents. This in turn allows for the reduction or even elimination of actual prototypes prior to choosing a design alternative. The mathematical formulas within the executable model can be modified to refine the model as necessary in order to validate system behavior against the requirements. This verification and simulation process could help in determining and identifying errors earlier and in turn, reduce the amount of time spent making these corrections at a later stage.

After completing the design of the TacSat-3 system within the IBM Rational Systems Developer tool using the SysML language, the TSS Team was able to realize a valuable set of Lessons Learned. These lessons learned are crucial and essential to the stakeholders of the project in that they can provide valuable insight into the capabilities and limitations of SysML. The findings of the TSS Team were that SysML adequately addresses systems engineering needs through a model-based design that is more clearly laid out. It was determined through working with SysML that it enables improved communications across development teams, while greatly enhancing the ability to manage complex systems. Upon the completion of the system design, SysML allowed the systems engineer/architect to move directly into creating executable model with little knowledge of programming. While conducting behavior analyses on the designed TacSat-3 system, the advantages of using SysML became evident. It was observed that SysML allowed for validating system behavior against requirements. By modifying the mathematical formulas used within the executable model, the team was able to manipulate the outcome of the model and validate system behavior against the defined requirements. Thereafter, modifications realized after performing design trade study could be automatically updated throughout the model.

6 CONCLUSIONS

Project Quicklook has shown the effectiveness of SysML and supporting tools. This effort has demonstrated that Model-Based Systems Engineering (MBSE), with all its purported benefits, is becoming a reality. Most of the advertised strengths of MBSE and SysML were clearly demonstrated in the TacSat-3 design and executable model. SysML's inclusion of requirements in the design model is a powerful tool for analysis of requirement traceability, satisfaction, and flexibility. The TSS design process proved that if designers follow SysML semantics when using software tools, concordance between design model diagrams is ensured. The ability to organize or package a SysML model in any fashion an organization desires provides powerful flexibility for work groups.

Software tool support for SysML is currently sufficient and will improve as more organizations adopt the language. TSS was able to use software tools to create an executable model that was derived directly from the SysML design model. Modifications to the design could also be reflected in the executable mode. The executable model was effective in conducting a trade study of design options.

The TSS Team's training and engineering results show that a design team can learn and use SysML in a reasonable amount of time (five standard workweeks) without significant training or experience. In this way, Project Quicklook has dispelled the notion that organizations cannot use model-based systems engineer with SysML because the start-up resource cost is too high.

7 ACKNOLWDGEMENTS

The TSS team would like to thank the following people and companies for their assistance with the Quicklook project.

Faculty Advisor

Kathryn B. Laskey, PhD

Systems Engineering and Operations Research Department

George Mason University

Project Sponsor

Shana Lloyd

Heather Howard

The Aerospace Corporation

Technical Advisor

Sanford Friedenthal

Chair of the OMG Systems Engineering Domain Special Interest Group (SE DSIG)

IBM Rational Software

The TSS team would like to thank IBM for providing the GMU SEOR department with licenses for their IBM Rational Software.

EmbeddedPlus

The TSS team would like to thank EmbeddedPlus for providing the GMU SEOR department with licenses for their SysML Toolkit and Simulation Toolkit.

8 REFERENCES

Sanford Freidenthal, Alan Moore, and Rick Steiner, *OMG Systems Modeling Language (OMG SysML) Tutorial*, INCOSE, 11 July 2006

Sanford Friedenthal, and Kobryn, *Extending UML to Support a Systems Modeling Language*, INCOSE Symposium, Toulouse, France June 2004

Ian Bailey, Fatma Dandashi, Huei-Wan Ang, and Dwayne Hardy, *Using Systems Engineering Standards In an Architecture Framework*

Thomas M. Davis and Captain Stanley D. Straight, *Development of the Tactical Satellite 3 for Responsive Space Missions*, USAF, 27 April 2006

Thomas M. Davis, Dr. Tom Cooley, and Captain Stanley D. Straight, *ARTEMIS: Tactical Satellite 3 for Responsive Space Missions*, USAF, 31 May 2006

Thomas M. Davis and Captain Stanley D. Straight, *Tactical Satellite 3: Requirements Development for Responsive Space Missions*, USAF, 27 April 2006

Appendix A Systems Modeling Language Term Definitions

The following definitions are from OMG SysML Glossary April 2006.

Term	**Definition**
Activity diagram	Diagram that depicts behavior associated with activities using input/output and control flow.
Actor	An actor specifies a role played by a user or any other system that interacts with the subject. (The term “role” is used informally here and does not necessarily imply the technical definition of that term found elsewhere in this specification.)
Block [SysML]	A modular unit that describes the structure of a system or element.
Block definition diagram [SysML]	A diagram that represents the relationship between blocks and the structural and behavioral features of blocks.
Internal block diagram [SysML]	A diagram that depicts the internal structure of a block, including the interaction points to other parts of the system. It shows the configuration of parts that jointly perform the behavior of the containing block. The diagram specifies a set of instances playing parts (roles) in the context of the enclosing block (context).
Lifeline	A lifeline represents an individual participant in the Interaction.
Package diagram	A diagram that depicts how model elements are organized into packages and the dependencies among them, including package imports and package extensions.
Parametric diagram [SysML]	A diagram that represents a network of constraints on properties to support engineering analysis such as performance, reliability and mass properties analysis.
Requirement [SysML]	A capability or condition that must (or should) be satisfied.
Requirements diagram [SysML]	A diagram that represents requirements and their relationships. See *requirement*
Sequence diagram	A diagram that depicts an interaction by focusing on the sequence of messages that are exchanged, along with their corresponding event occurrences on the lifelines. Unlike a communication diagram, a sequence diagram includes time sequences but does not include object relationships. A sequence diagram can exist in a generic form (describes all possible scenarios) and in an instance form (describes one actual scenario). Sequence diagrams and communication diagrams express similar information, but show it in different

Term	**Definition**
	ways. See: *communication diagram*
State machine diagram	A diagram that depicts discrete behavior modeled through finite state-transition systems. In particular, it specifies the sequences of states that an object or an interaction goes through during its life in response to events, together with its responses and actions. See: *state machine*.
State machine	State machines can be used to express the behavior of part of a system. Behavior is modeled as a traversal of a graph of state nodes interconnected by one or more joined transition arcs that are triggered by the dispatching of series of (event) occurrences. During this traversal, the state machine executes a series of activities associated with various elements of the state machine.
Swimlane	A partition of an activity diagram – **partition** A grouping of any set of model elements based on a set of criteria.1. activity diagram: A grouping of activity nodes and edges. Partitions divide the nodes and edges to constrain and show a view of the contained nodes. Partitions can share contents. They often correspond to organizational units in a business model. They may be used to allocate characteristics or resources among the nodes of an activity.2. architecture: A set of related classifiers or packages at the same level of abstraction or across layers in a layered architecture. A partition represents a vertical slice through an architecture, whereas a layer represents a horizontal.
Use case	A use case is the specification of a set of actions performed by a system, which yields an observable result that is, typically, of value for one or more actors or other stakeholders of the system. See: *use case instances.*
Use case diagram	A diagram that shows the relationships among actors and the subject (system), and use cases.

Appendix B Acronyms

BDD	Block Definition Diagram
CDL	Common Data Link
CONOPS	Concept of Operations
DODAF	Department of Defense Architecture Framework
GUI	Graphical User Interface
IBD	Internal Block Diagram
IBM SDP	IBM Software Development Platform
JFC	Joint Forces Command
MBSE	Model-based Systems Engineering
NSPD-40	National Security Presidential Directive
NSS	National Security Space
OMG	Object Management Group Inc.
OO	Object Oriented
RSD	Rational Systems Developer
SEOR	Systems Engineering and Operations Research
SysML	Systems Modeling Language
TacSat-3	Tactical Satellite 3
TSS	Tactical Science Solutions Inc.
UDD	Unified Data Dictionary
UML	Unified Modeling Language
V&V	Verification and Validation