

# Satellites to Supply Chains, Energy to Finance — SLIM for Model-Based Systems Engineering

## *Part 1: Motivation and Concept of SLIM*

Copyright © 2011 InterCAX LLC and Georgia Institute of Technology . Published and used by INCOSE with permission.

Manas Bajaj<sup>1\*</sup>, Dirk Zwemer<sup>1</sup>, Russell Peak<sup>2</sup>, Alex Phung<sup>1</sup>, Andrew Scott<sup>1</sup>, Miyako Wilson<sup>2</sup>

1. InterCAX LLC  
75 5<sup>th</sup> Street, Suite 213,  
Atlanta GA 30308 USA  
[www.intercax.com](http://www.intercax.com)

2. Georgia Institute of Technology  
Model-Based Systems Engineering Center,  
MARC, 813 Ferst Drive, Atlanta GA 30318 USA  
[www.mbse.gatech.edu](http://www.mbse.gatech.edu)

### Abstract

Development of complex systems is a collaborative effort spanning disciplines, teams, processes, software tools, and modeling formalisms. Increasing system complexity, reduction in available resources, globalized and competitive supply chains, and volatile market forces necessitate that a unified model-based systems engineering environment replace ad-hoc, document-centric and point-to-point environments in organizations developing complex systems.

To address this challenge, we envision SLIM—a collaborative, model-based systems engineering workspace for realizing next-generation complex systems. SLIM uses SysML to represent the front-end conceptual abstraction of a system that can “co-evolve” with the underlying fine-grained connections to models in discipline-specific tools and standards. With SLIM, system engineers can drive automated requirements verification, system simulations, trade studies and optimization, risk analyses, design reviews, system verification and validation, and other key systems engineering tasks from the earliest stages of development directly from the SysML-based system model. SLIM provides analysis tools that are independent of any systems engineering methodology, and integration tools that connect SysML with a wide variety of COTS and in-house design and simulation tools.

We are presenting SLIM and its applications in two papers. In Part 1 (this paper), we present the motivation and challenges that led to SLIM. We describe the conceptual architecture (section 1) and use cases (section 2) of SLIM followed by tools available for production and evaluation usage (section 3). In Part 2 paper—*SLIM Applications*—we present the applications of SLIM tools to a variety of domains, both in traditional as well as non-traditional domains of systems engineering. Representative examples from space, energy, infrastructure, manufacturing and supply chain, military operations, and bank systems are presented.

---

\* Corresponding author: Manas Bajaj, [manas.bajaj@intercax.com](mailto:manas.bajaj@intercax.com), phone: +1-404-592-6897. Preferred citation: Bajaj, M., Zwemer, D., Peak, R., Phung, A., Scott, A. and Wilson, M. (2011). *Satellites to Supply Chains, Energy to Finance — SLIM for Model-Based Systems Engineering, Part 1: Motivation and Concept of SLIM*. 21st Annual INCOSE International Symposium, Denver, CO, June 20-23, 2011.

# 1 Introduction to SLIM (Systems Lifecycle Management)

SLIM is an integrated software platform for systems lifecycle management. It is envisioned to provide capabilities that combine the strengths of model-based systems engineering and product lifecycle management (PLM). Though the scope of SLIM is from systems design to delivery and sustainment, this paper primarily focuses on systems design and analysis aspects of SLIM and related tools available to date. In this section, we first present the motivation for SLIM in terms of gaps in current state-of-the-art commercial tools for design and analysis of complex systems. Following this, we present a conceptual architecture of SLIM and describe key end-user capabilities that SLIM provides.

## 1.1 Motivation

Figure 1 illustrates two types of fundamental gaps in currently available tools for systems design and analysis. These gaps are described below in the context of system lifecycle phases, and are generally applicable to both traditional and non-traditional systems engineering domains. As an example, Figure 1 shows the lifecycle phases for typical NASA systems (NASA 2007).

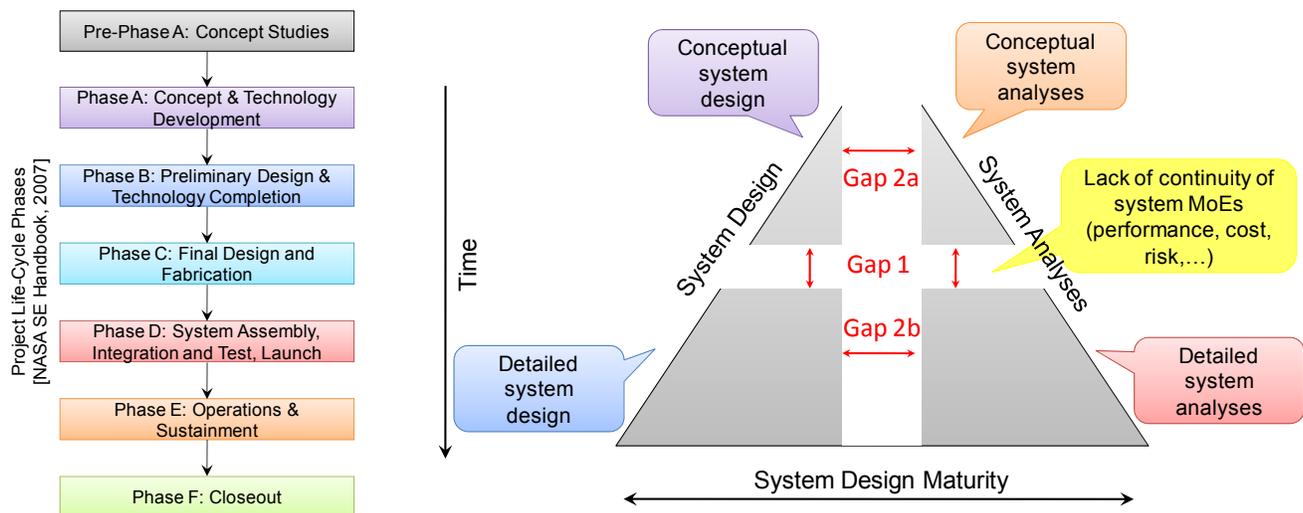


Figure 1: Gaps in current state-of-the-art tools for design and analysis of complex systems

**Gap 1** represents lack of model-based continuity of system engineering activities from the early phases (proposals and conceptual design) to detailed phases (detailed design, development and delivery). Gap 1 exists because the tools used for systems modeling and analyses are different in each phase. In several domains, conceptual design phases largely depend on the use of diagramming tools and spreadsheets. Tools used in a design phase capture only those aspects of the system that are relevant for that phase, and hence there is no continuity of system definition, requirements, performance parameters, and their evaluation results from one phase to another. Results of design exploration, uncertainty in design parameters, assessment of systemic risk, and other cross-phase parameters and studies are typically buried in mammoth and difficult-to-maintain spreadsheets, documents, and proprietary domain-specific tools thereby making it difficult to track and communicate them effectively during phase transitions.

Information forwarding for core aspects of the system, such as the requirements and objectives, structure and behavior, performance and risk parameters, cost, power and weight budgets, and its development process (tasks, resources and milestones) need to be maintained continuously through the system design and development phases independent of the diverse COTS tools used for representing and managing these aspects. In this light, some of the questions posed by systems engineering and project management teams are:

- How does one ensure that models defined in tools during early phases and later phases are representations of the same system?
- How does one ensure that the requirements and performance parameters evaluated in tools during early phases and later phases concern the same system or even the same project?
- How does one relate the analysis and evaluation results obtained during early phases and those obtained during later phases?
- How does one propagate trade study results, uncertainties in system parameters, and systemic risk assessment from the preliminary design phases to detailed design phases, and eventually to system deployment and sustainment?

**Gap 2** represents disconnects between the concerns in a given phase, such as disconnects between design and analysis/simulation models in a design phase. Gap 2 manifests in heterogeneous model transformations beyond design and analysis, such as between requirements and structure, logical structure and physical structure, and structure and behavior. Typically, individual modeling and simulation tools represent only specific aspects of a system, such as requirements, structure, and behavior, and do not provide a holistic model of the entire system. As a result, systems engineering activities can at best be realized using point-to-point data flows, custom model transformations, and workflow automation software to keep all aspects in sync. There is an urgent need in systems engineering for using a single coherent system model that can federate domain-specific design and analysis models in a configuration managed environment.

## **1.2 Conceptual architecture**

SLIM is a software environment for integrated model-based systems engineering. As shown in Figure 2, it uses SysML as the front-end for multi-disciplinary teams to collaboratively develop a unified, coherent representation of the system from the earliest stages of development. The system model (in SysML) can ‘co-evolve’ with the associated domain-specific models, such as Computer-Aided Design and Engineering (CAD/CAE) models. Relationships between the system model and the domain-specific models can range from qualitative dependency relations to quantitative acausal parametric relations which are executable on-demand for seamless model traceability and interoperability. The SysML-based system model is a conceptual abstraction of the system that has sufficient details for orchestrating all systems engineering activities, ranging from automated requirements verification and system trades to risk analysis and system V&V. This unified system model is *not* a data store but a map of the system which can be used to federate domain-specific models of different aspects of the system.

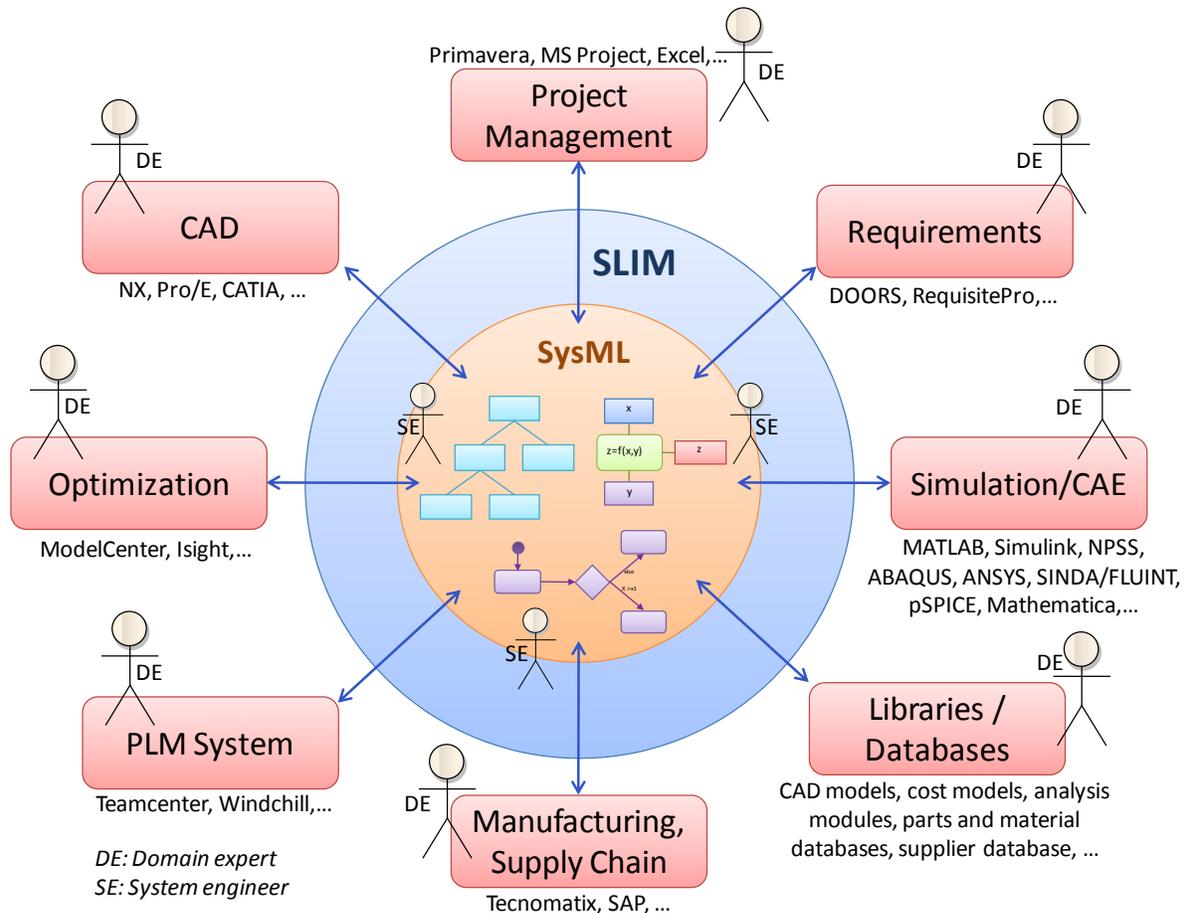


Figure 2: Conceptual Architecture of SLIM

SLIM builds on existing SysML authoring environments, such as MagicDraw (No Magic 2010), Artisan Studio (Atego 2010), Rhapsody (IBM 2010), and Enterprise Architect (IBM 2010), and PLM environments, such as Teamcenter (Siemens PLM 2010) and Windchill (PTC 2010). SLIM provides plugins to SysML authoring environment for facilitating systems engineering design, integration, and verification and validation (V&V) flows directly from the SysML model. Automated requirements verification, system trades and optimization, risk analysis, sizing, cost and performance estimation, producibility analyses, model-based design reviews, and more can be triggered from the SysML model. At its core, SLIM provides software tools that are independent of any systems engineering methodology, and are building blocks for supporting different types of design and verification flows. We envision that SLIM will provide system engineering teams the flexibility to define, test, and deploy organization-specific workflows using the building block tools.

### 1.3 SLIM as a design environment

SLIM provides a rich and seamless design environment for complex systems engineering, which includes organizations, humans, operating environment, and other aspects in addition to the system-of-interest. Figure 3 illustrates the difference between existing design environments, generalized for multiple domains, and SLIM's design environment for realizing complex

systems. Typical design environments for complex systems are driven by document-based integration or point-to-point model-based integration between models representing different aspects of the system. Lack of specialized tools for conceptual design, unlike CAD and CAE tools for detailed design, leads to usage of diagramming tools (such as PowerPoint (Microsoft 2010) and Visio (Microsoft 2010)) with spreadsheets for numerical analysis. Workflow automation tools provide a means to transfer data from one tool to another at best. A significant amount of time and resources are spent by system engineering teams in the administration of islands of automation and document-based interfaces, and producing reports for system design and project reviews.

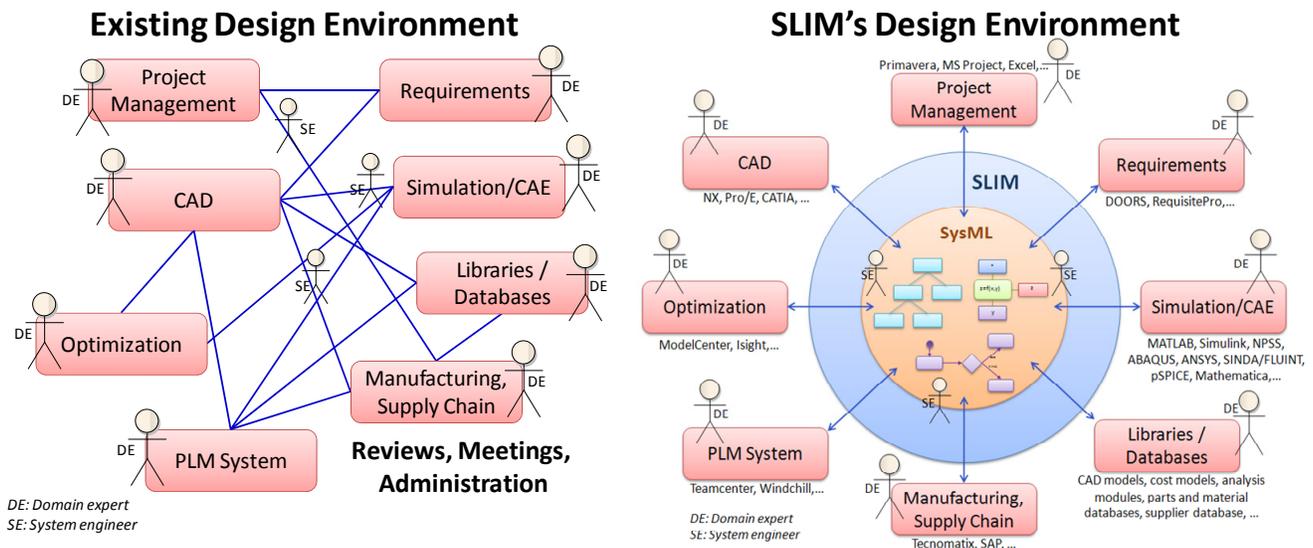


Figure 3: Comparison of existing design environment and SLIM's design environment for complex systems

In contrast, SLIM's design environment is driven by a SysML-based system model that is the centerpiece of the design process from the beginning. SLIM provides tools that aid system engineering activities directly from the SysML-based system model. SysML structure and behavior constructs (such as blocks and activities) replace drawing tools, while parametrics replaces spreadsheets. Automated unit checking, requirements verification, and trades can now be performed from the SysML model. Changes in the design architecture automatically reflect in the analysis models unlike manual updates from drawing tools to spreadsheets.

Since no single modeling language can represent all aspects of a system throughout its lifecycle, SLIM provides tools that integrate SysML with a wide array of in-house and COTS design and simulation tools, and documents and spreadsheets for data presentation and reporting. The level of integration can range from qualitative dependency relationships for traceability, interface specifications for model generation and reconfiguration, and/or quantitative relationships for model synchronization. Coupled with enterprise PLM systems, SLIM enables change and configuration management, conflict resolution, and model-based communication between stakeholders.

SLIM provides libraries of reusable models—SysML constructs connected to native design and simulation models—that make it easy to assemble and perform optimization and trades on

the system model. Examples of libraries for aircraft systems include design objects (wing, fuselages, tails, and airfoils), analysis models (cost, sizing), system behavior (mission profiles), materials, and manufacturing constructs. Examples of libraries for supply chains include libraries of: (a) suppliers, (b) customers, (c) parts and assemblies, (d) costing models, and (e) supplier allocation and optimization models.

For complex systems, SLIM provides a powerful alternative to “PowerPoint and Spreadsheet engineering” and point-to-point integration of analysis codes (spreadsheets, MATLAB/Simulink, C++, Fortran, Java) disintegrated from the system design definition.

## **1.4 Preliminary design and evaluation of advanced concepts**

SLIM directly answers the following challenge—*the conceptual design process has the highest impact on the product lifecycle but the least availability of design and decision support tools*. SysML, the front-end of SLIM, is a general-purpose systems modeling language and an open standard that is not confined to any specific design phase or methodology. SLIM serves as a decision support environment for systems-of-systems. The system-of-interest can be co-developed with the related operating environment, organization, humans, manufacturing facilities, government regulations, and market behavior. With a unified SysML model as the centerpiece, the impact of changes in requirements or any other element can be studied on both qualitative and quantitative levels.

SLIM provides analysis tools (section 3) for executing SysML constructs representing different aspects of the system, such as SysML parametric models for cost, performance, reliability, and SysML activity models for mission profiles. As a result, systems engineers can perform system trades, risk analyses, and automated requirements verification on system alternatives on a continuous basis. SLIM’s integration tools (section 3) will allow system engineers to connect the SysML-based system model to design and analysis models in COTS tools—concept sketches in CAD tools and orbit planning models in STK (AGI 2010)—for driving system analyses during conceptual phases.

A key factor in the incorporation of new designs, materials, and manufacturing technologies into new systems is the ability to model the entire end-to-end system lifecycle. SysML can incorporate issues of producibility, maintainability and disposal into the system model from the earliest stages of conceptual design. While SysML provides rich and extensible constructs for modeling “systems-of-systems”, SLIM enables qualitative and quantitative “what-if” analyses and trades for both unconventional and conventional systems built on combinations of new and existing technology.

## **1.5 Transition from conceptual design to detailed design**

SLIM provides plug-n-play and automated model re-configuration capabilities to swap low-fidelity analysis models with high-fidelity ones, while still maintaining the conceptual relationships (in SysML) between the models. The suite of integration tools in SLIM (section 3) allows system engineers to wrap externally-defined design and simulation models and codes as SysML constructs and replace (or combine) low-fidelity by high-fidelity analyses. Bi-directional

relationships between the SysML-based system model and domain-specific models in COTS tools, as shown in Figure 2, represent both qualitative and quantitative relationships that enable a wide range of model interoperability services—traceability, transformation, reconfiguration, synchronization, and conflict resolution. Design, behavior, and requirements-related parameters in the SysML model can link to spreadsheets during the conceptual design phases or CAD and CAE models during the detailed design phases (Figure 5).

## 1.6 Detailed design and analysis

SLIM builds on several years of design-analysis integration research (Peak, Paredis et al. 2005; Peak, Burkhart et al. 2007; Peak, Burkhart et al. 2007; Bajaj 2008) that provides patterns and methods to automatically compose SysML-based simulation templates—parametric models relating design models to analysis models—from a library of analytical building blocks for variable topology systems. These patterns make it possible to abstract analysis model formulation from solution, thereby allowing designers and analysts to rapidly (re)formulate analysis models and simulation templates from high-level specifications and also explore different solution strategies (e.g. FEM, FDM, Meshless) and solvers (e.g. ANSYS, ABAQUS) for an analysis problem. Trades and sensitivity analyses can be automatically triggered from the SysML-based simulation templates that also provide specifications for design space exploration and multi-disciplinary design optimization using tools such as ModelCenter (Phoenix Integration 2010) and Isight (Dassault Systèmes SIMULIA 2010).

SLIM’s approach for relating requirements, design, and analysis does not require each engineer to learn the others’ tools or compromise his or her own models. Figure 4 shows the general concept. Requirements, design, and analysis concerns can be modeled for every component in the system model. Component-level requirements, derived from decomposing system-level requirements, are pushed from the system model to specialists (such as designers and analysts) while design parameters and analysis results flow back into the system model. Parametric relationships can exist between concerns (design, analysis, requirements) for a given component, and across the system hierarchy (such as for weight and cost roll-ups).

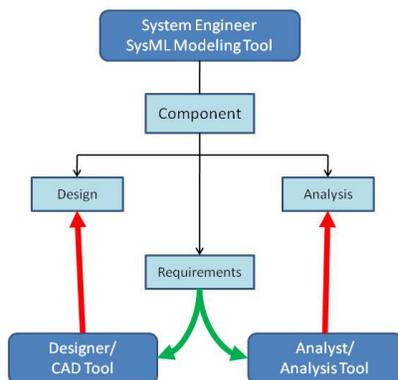


Figure 4: Interaction between systems engineer, designer, and analyst.

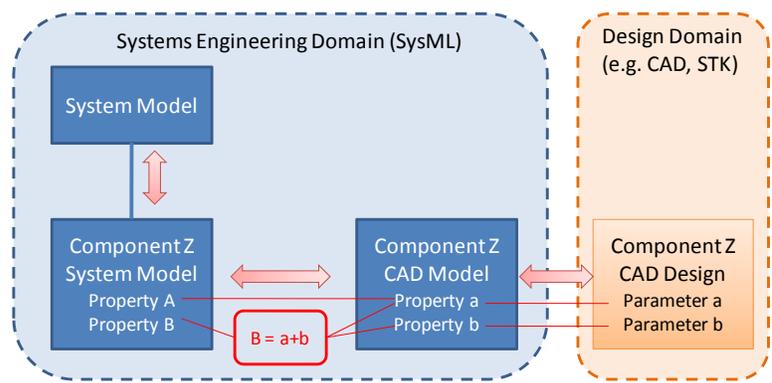


Figure 5: Later-stage systems engineering – design - analysis integration.

SLIM provides multiple types of model-connectivity patterns that can be used to connect the SysML-based system model to externally-defined design and analysis models. System engineers,

designers, and analysts can select the connectivity mode best suited for the purpose. Both file-based exchange (STEP, XML) and live model-based interaction (via native API) are supported. Figure 5 above illustrates one of these patterns used for relating detailed design models to the SysML model. The CAD design for Component Z resident in the CAD tool is mirrored automatically by SLIM in the SysML modeling environment. The CAD model parameters, however, may not correspond exactly to the system model properties needed to evaluate other aspects of the system. For example, a CAD model may calculate true wing area, while systems calculations require the wing reference area. SysML parametrics is used to connect the CAD model parameters and the system model parameters at a fine-grained level. Execution of parametric relationships enables bi-direction information flow.

## 2 SLIM Use Cases

The goal of SLIM is to facilitate model-based design, analysis, review, change management, procurement, and sustainment of complex systems. Figure 6 is a SysML use case diagram that illustrates some of the key systems engineering use cases that SLIM will address. The use cases are described in this section in an anti-clockwise order of appearance in the figure, starting with the use case *Facilitate model-based design and analysis*.

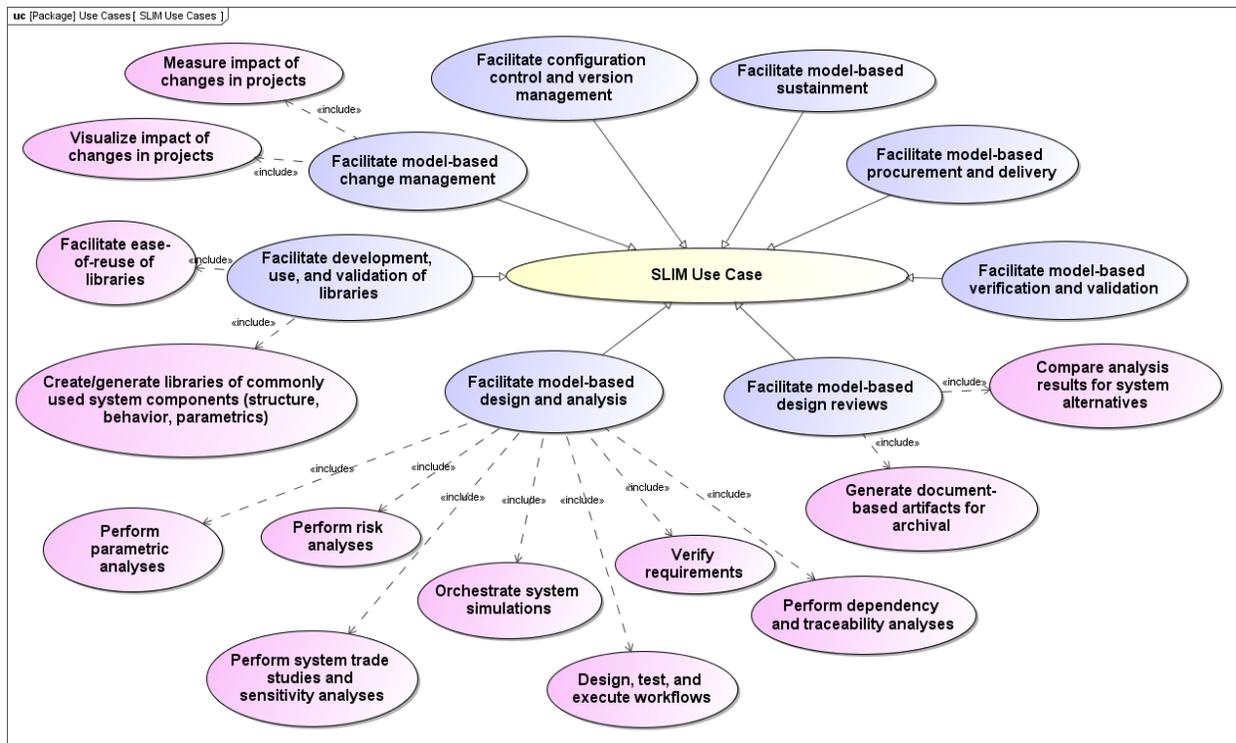


Figure 6: Key systems engineering use cases of SLIM

SLIM tools that are currently available or in-development are primarily focused on **model-based design and analysis** of complex systems. This includes tools for parametric analysis, orchestrating system simulations, performing trade studies and risk analyses, and automated verification of requirements.

SLIM will facilitate **model-based design reviews**. Design reviews in existing design environments involve manual creation of an extensive set of documents that have to be carefully scanned by review teams. In SLIM's design environment, reviewers will be able to invoke all system alternatives that were analyzed, the types of analyses performed, the analysis results, and design decisions and rationale directly from the SysML-based abstraction of the system. Document-based artifacts can be automatically generated from the SysML model for archival and future reference.

A unified system model can represent not only the design and analysis aspects of a system but also test procedures necessary for verification and validation (V&V) and their relationships to system and sub-system requirements. SLIM can facilitate **model-based V&V** directly from the SysML model. Test activities can be scheduled and automatically executed each time a new version of the system model is checked into a repository. This capability is similar to unit and functional testing and project automation commonly used in software development projects, but for large systems engineering projects.

SLIM can facilitate **model-based procurement and delivery** of complex systems. The SysML-based system model is not only useful for facilitating the development of complex systems but also as a contractual artifact binding the relationship between an organization and its customers, suppliers, and collaborators. Customers can issue request-for-proposals (RFPs) using requirements, use cases, and functional specifications defined using SysML. Proposals and deliverables in the form of SysML-based design models can then be automatically verified and analyzed against specifications. After a system has been delivered and deployed, the SysML-based system model serves as a blueprint of system structure, behavior, and target performance rating, much like design drawings and access plans for aircrafts and ships. System upgrade, maintenance, repair, and other **sustainment**-related plans can then be mapped and generated from the system model.

Design environments for collaborative development of complex system need to provide configuration control and version management capabilities. SLIM will leverage project management and **configuration control** capabilities native to commercial-strength PLM tools such as Teamcenter.

With the existence of a system model, SLIM can facilitate **model-based change management**. Changes in customer requirements, available resources, system design and behavior are common in large systems engineering projects. With SLIM, system engineers can analyze the impact of changes via visualization and parametric analyses from the SysML model. Such analysis will be crucial to planning, controlling, and affecting changes in the system design process.

SLIM will provide tools for creating **libraries** of reusable system constructs, such as those based on structural, parametrics, and behavior concepts. Examples for aircraft systems include libraries of aircraft wings and fuselages, libraries of aircraft maneuvers and mission profiles, and libraries of cost, weight, and sizing analysis models. Elements in the library can be defined natively in SysML, or can be SysML elements wrapping externally-defined library elements such as CAD and STK models.

### 3 SLIM Components

In this section, we present SLIM tools that are available for production and evaluation usage. Figure 7 is a SysML block definition diagram that shows the conceptual decomposition of the SLIM framework. SLIM tools are accessible and driven from SysML authoring environments, and will access externally-defined models in PLM environments in the future. The main components of the SLIM framework are listed below, followed by a detailed description of tools and their availability.

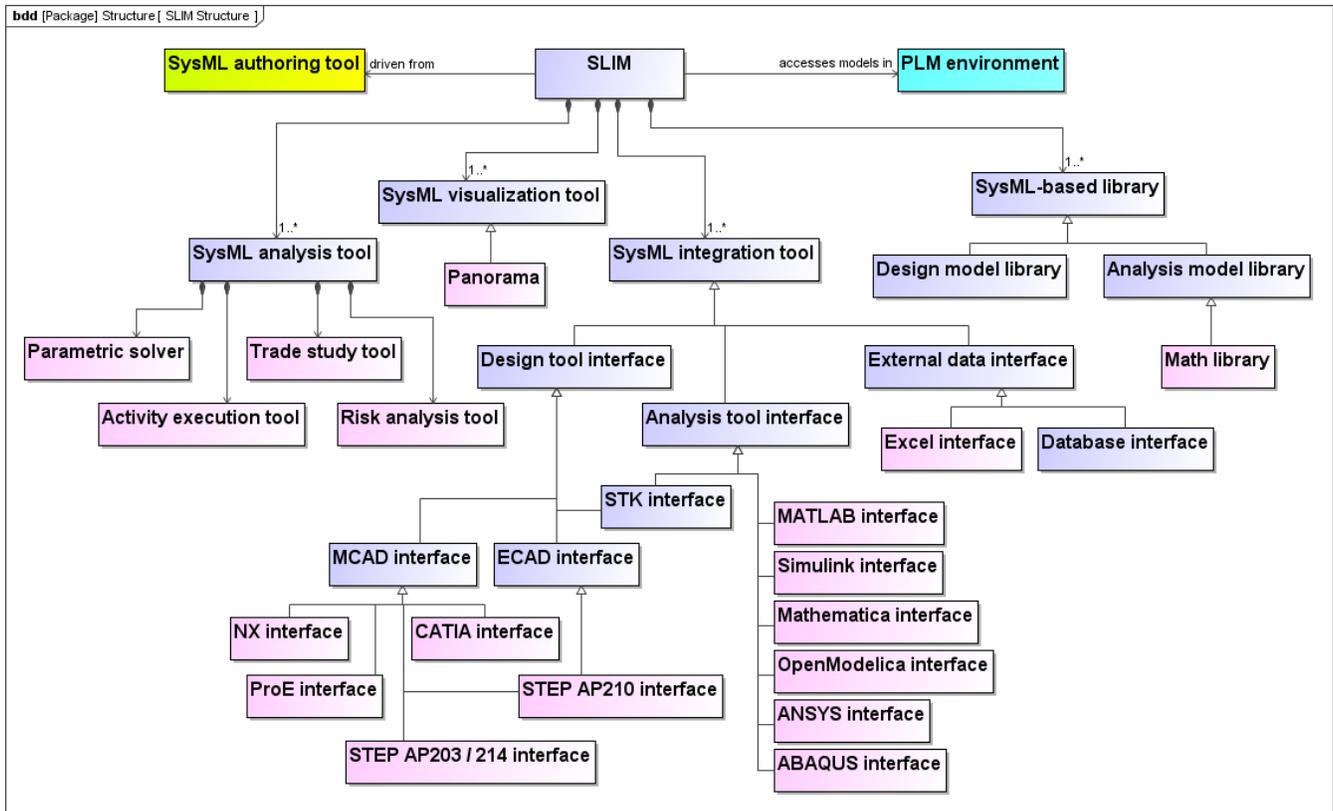


Figure 7: Conceptual decomposition of the SLIM framework

- *SysML analysis tools* provide capabilities to trigger system-level analyses and automated requirements verification from a SysML model. This includes (1) Parametric solver for executing parametric relationships, (2) Trade study tool for performing trade studies using parametric models, (3) Risk analysis tool for performing Monte Carlo simulations based on parametric and activity models, and (4) Activity execution tool for executing SysML activity models.
- *SysML visualization tools* provide capabilities to visualize SysML models, in addition to the nine (9) types of SysML diagrams, for enhanced model traceability. Flattened and hierarchical graph- and table-based views can better highlight cross-cutting relationships.

- *SysML integration tools* provide capabilities to integrate the SysML-based system model with models and data based on other open standards (such as STEP) or defined in COTS and in-house tools.
- *SysML-based libraries* provide building blocks for creating and executing the SysML-based system model. Library elements could be defined natively in the SysML environment, such as library of math functions for parametric analyses and trade studies, or involve SysML elements wrapping CAD, CAE, and STK models in PLM environments.

In sections 1-2, we presented the vision and use cases of SLIM. In this section, we present SLIM tools and capabilities that are: (1) available commercially for production usage, (2) available as beta for early adopters, and (3) are under development. We invite users to participate in this effort by using existing tools, providing feedback, and contributing to the development of new tools under the SLIM framework. Two categories of SysML-based tools are presented here. These are: (1) SysML analysis tools, and (2) SysML integration tools.

## **3.1 SysML analysis tools**

### **3.1.1 Parametric solver**

The Parametric solver tool executes SysML parametric models using math solvers such as Mathematica (Wolfram Research 2010), OpenModelica (OSMC 2010), and MATLAB (The MathWorks 2010). Parametrics is one of the four pillars of SysML and provides modeling constructs to define acausal, fine-grained, mathematical relationships between SysML block properties. Parametrics is applicable to a broad range of systems engineering use cases. For example, parametric relations could represent the following scenarios:

- Design constraints, such as sum of the weights of all parts should be equal or less than a pre-defined value.
- Simulation templates that relate design and analysis models, such as the calculation of cost variables, key performance parameters (KPPs), risk metrics, and other measures of effectiveness (MoEs) from the design model.
- Mathematical relationships between different design views and analysis views, such as relationships between properties of logical and physical system structure, between properties of hardware and software components, or between SysML blocks representing physical system components and the corresponding CAD models.
- Property-based requirements—refinement of text-based requirements—represented as mathematical constraints that can then be automatically verified for design alternatives.

Due to its broad applicability, SLIM's Parametric solver is a powerful systems engineering tool. It can be used for verifying or solving design constraints; computing system MoEs, KPPs, and risk metrics; synchronizing diverse views of the system; automatically verify requirements; and connecting, synchronizing, and controlling externally-defined models (CAD/CAE/Simulink) with SysML models.

Figure 8 shows a parametric model (in MagicDraw) of the FireSat system model being developed by the INCOSE SSWG Challenge Team (INCOSE). This parametric model relates performance metrics (ground coverage and scan resolution) and annual operational cost of the FireSat to the design and operational variables, such as orbit altitude, sensor resolution, and angle covered by the optical instruments. The performance metrics and the cost variable are highlighted in red in the parametric diagram. The parametric model also shows constraints used for verifying requirements. The `ResolutionReq` constraint block is used for the `rR1` constraint property (bottom right corner of the parametric diagram) and is a mathematical representation of a text-based requirement concerning FireSat scan resolution. It compares the scan resolution to the required value (30m) and computes the `ScanResVerify` property that represents the result of requirement verification (1 for pass and 0 for fail).

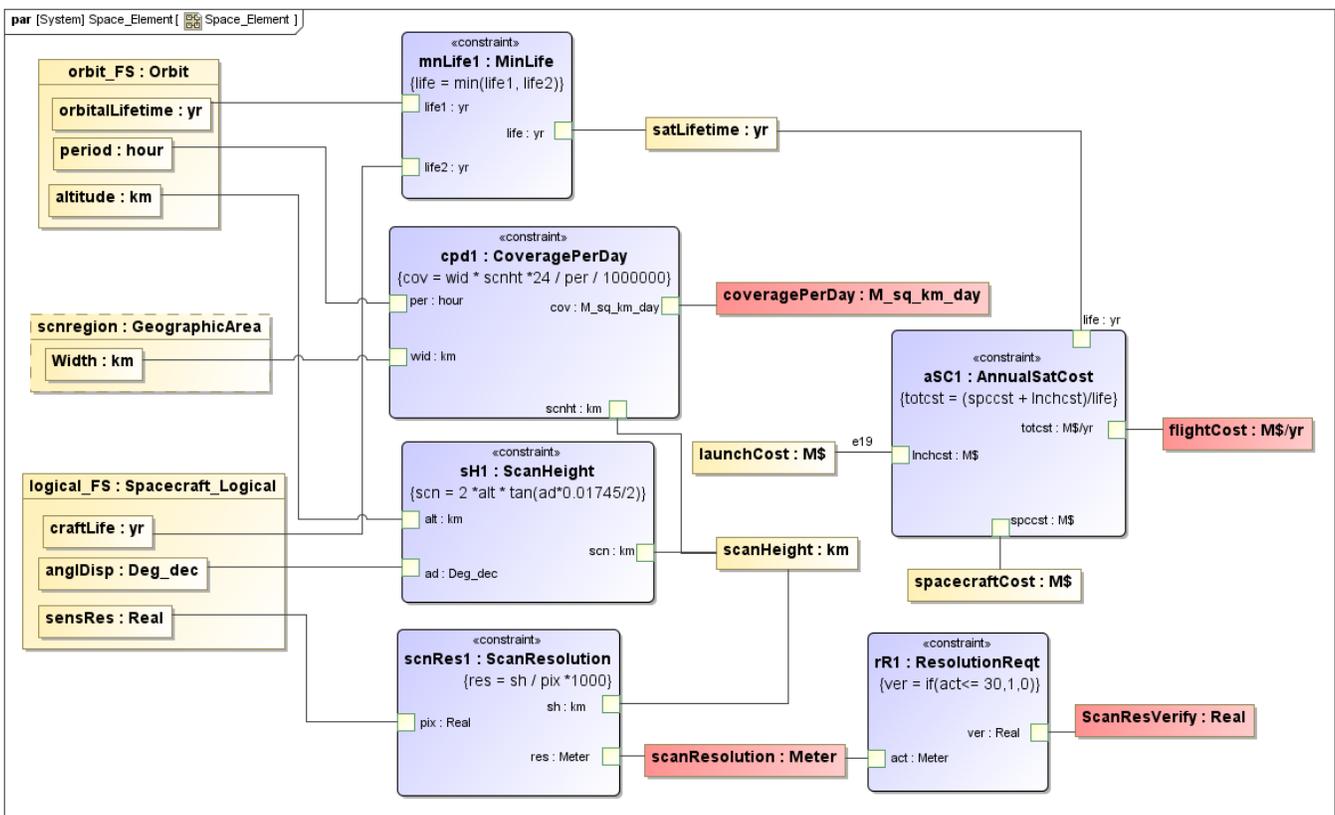


Figure 8: SysML parametric model for computing coverage, scan resolution, and flight cost of the FireSat, and for verifying resolution requirement

Figure 9 shows the trade study results generated by ParaMagic<sup>®</sup> (InterCAX - ParaMagic 2010)—SLIM’s Parametric solver for MagicDraw. Each row represents a trade study scenario (given values of design and operational variables) for 2 FireSats. In this case, ParaMagic<sup>®</sup> read input values for each scenario from the spreadsheet; computed target variables (value properties `flightCost`, `coveragePerDay`, and `scanResolution` in Figure 8); verified them against requirements; and exported results back to the spreadsheet. The columns `Cost req.`, `Coverage req.`, and `Res. req.` correspond to the results of verifying cost, coverage, and resolution (for each FireSat) requirements—1 for pass and 0 for fail. The results indicate that when the satellites

operate at lower orbits, the annual costs exceed the operating budget and the coverage is not sufficient to meet the requirements. On the other hand, when the satellites operate at higher orbits, the resolution requirement fails. Rows highlighted in green are scenarios for which all requirements were successfully met. Sensitivity plots and other graphs can also be generated by the Parametric solver.

Satellite 1		Satellite 2						Satellite 1		Satellite 2	
Altitude	Ang. Aperture	Altitude	Ang. Aperture	Cost/yr.	Cost req.	Coverage/day	Coverage req.	Tgt. Resolution	Res. req.	Tgt. Resolution	Res. req.
km	deg	km	deg	M\$/yr	1-pass, 0-fail	M sq km/day	1-pass, 0-fail	meters	1-pass, 0-fail	meters	1-pass, 0-fail
300	3	300	3	77.23	0	2.50	0	15.71	1	15.71	1
325	3	325	3	51.61	0	2.70	0	17.02	1	17.02	1
350	3	350	3	36.28	0	2.89	0	18.33	1	18.33	1
375	3	375	3	26.65	0	3.08	1	19.64	1	19.64	1
400	3	400	3	20.36	0	3.26	1	20.94	1	20.94	1
425	3	425	3	19.67	1	3.45	1	22.25	1	22.25	1
450	3	450	3	19.67	1	3.63	1	23.56	1	23.56	1
475	3	475	3	19.67	1	3.81	1	24.87	1	24.87	1
500	3	500	3	19.67	1	3.99	1	26.18	1	26.18	1
525	3	525	3	19.67	1	4.17	1	27.49	1	27.49	1
550	3	550	3	19.67	1	4.34	1	28.80	1	28.80	1
575	3	575	3	19.67	1	4.52	1	30.11	0	30.11	0
600	3	600	3	19.67	1	4.69	1	31.42	0	31.42	0

Figure 9: Trade study and requirement verification results generated and reported by ParaMagic®

Figure 10 shows a parametric diagram in Artisan Studio where the SimulinkHomeHeating constraint block wraps a MATLAB script M-file used for executing a Simulink model. ParaSolver (InterCAX - ParaSolver 2011) —SLIM’s Parametric solver for Artisan Studio— executes all relationships in the parametric model, including the wrapped Simulink model, to compute the target properties. In a similar manner, constraint blocks can wrap user-defined Mathematica functions. Figure 10 also shows the ParaSolver browser (bottom left) which presents the parametric models for a system alternative in an object-oriented spreadsheet structure. When a user selects a specific system component, the parametric relations owned by that component can be seen in the relationship table of the browser.

SLIM’s Parametric solver can be used by system engineers to orchestrate system simulations and automate analysis workflows. System engineers can create parametric models that chain externally-defined MATLAB functions or scripts, Mathematica functions, Simulink models to compute key performance parameters and other MOEs.

SLIM’s SysML Parametric solver is available for production usage for the following SysML authoring tools:

- **ParaMagic®** (InterCAX - ParaMagic 2010) for MagicDraw
- **Melody™** (InterCAX - Melody 2010) for Rational Rhapsody
- **ParaSolver™** (InterCAX - ParaSolver 2011) for Artisan Studio
- **Solvea™** (InterCAX - Solvea 2011) for Enterprise Architect

These tools are being used for system design and analysis at leading organizations in aerospace, defense, supply chain, energy, and electronics and telecommunications sectors.

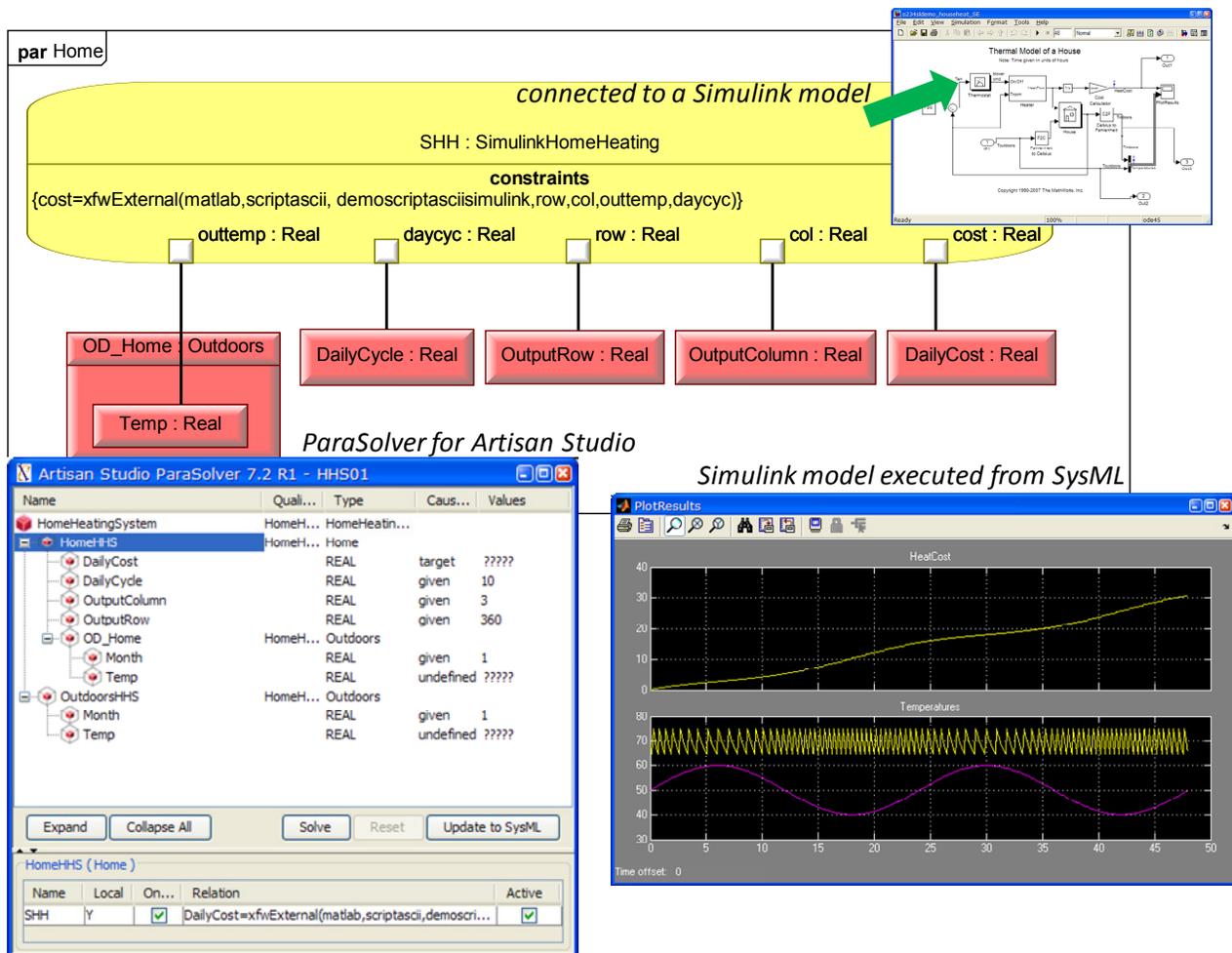


Figure 10: Parametric constraints wrapping Simulink models executed by ParaSolver (for Artisan Studio)

### 3.1.2 Trade study tool

The Trade study tool provides system engineers the capability to setup and execute trade studies directly from the SysML model. In the current version, system engineers can execute parametric models in a batch mode for a set of scenarios that have been specified explicitly or implicitly as intervals or discrete values.

Figure 13 illustrates the configuration view of the Trade study tool. The configuration view shows the SysML instance model in the left hand pane that serves as a fixed topology template for parameter variations in the trade study. Users can identify inputs, constants, and outputs in the trade study, and specify the data sources and targets for the input and output variables respectively. Inputs can be setup in an explicit manner by specifying discrete values or connecting to a cell range in an Excel spreadsheet, or in an implicit manner by specifying a range with increments. The Trade study tool automatically generates scenarios (if specified implicitly) by permuting all combinations of input variables. While running a trade study, the tool shows the specific scenario for which the parametric models are being executed and the specific solver being used. After execution, the Trade study tool exports results to the spreadsheets configured during setup. Post-processing and plotting capabilities of Excel can be used for displaying and reporting trade study results.

The Trade study tool is commercially available with each of the Parametric solver tools—**ParaMagic<sup>®</sup>**, **Melody<sup>™</sup>**, **ParaSolver<sup>™</sup>**, **Solvea<sup>™</sup>**.

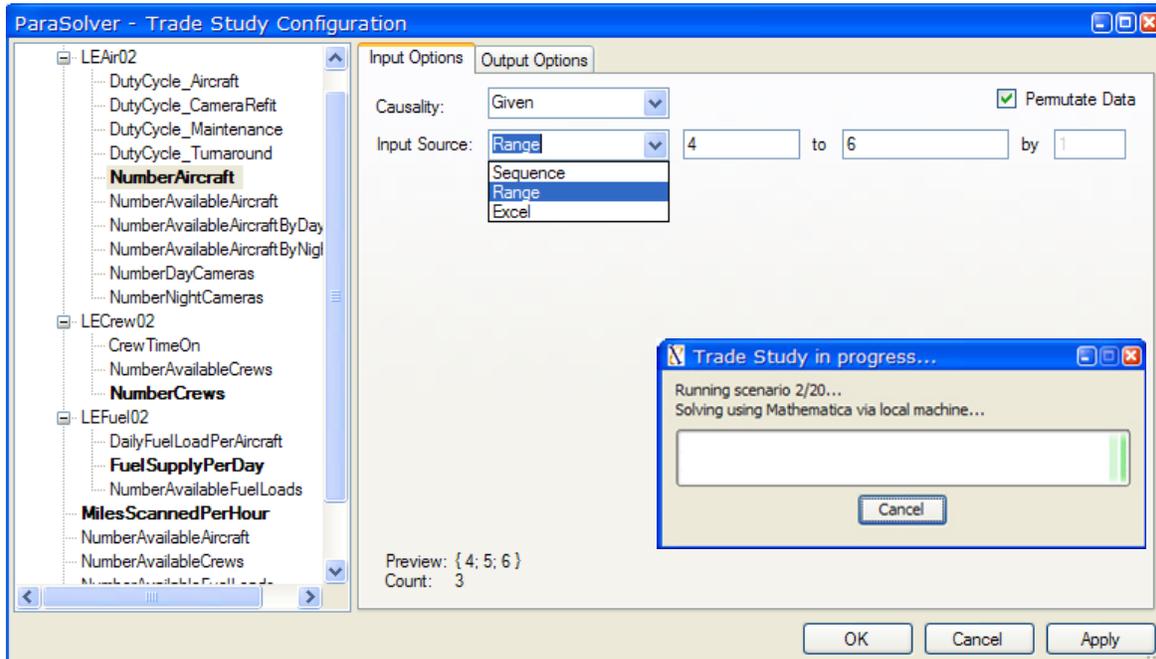


Figure 11: Trade study tool – configuration view

In the future, the Trade study tool will be enhanced to perform design exploration during system trades, potentially leveraging optimization and design exploration capabilities of ModelCenter and Isight.

### 3.2 Risk analysis tool

SLIM’s Risk analysis tool will provide system engineers the capability to compute system MoEs, KPPs, cost, and other metrics given the uncertainties in input variables—design, operation, project, and others. Given the probability distributions for input variables, the Risk analysis tool will run Monte Carlo-type simulations on SysML parametric models and/or activity models to calculate probability distributions and related statistical metrics for output variables (such as system MoEs and KPPs). Such analyses directly benefit system engineers developing proposals or designing complex systems in performing trades between performance, cost, schedule, and risk.

Figure 12 shows a SysML parametric model for a military operation where a defense system composed of aircrafts, UAVs, ground forces, and analysis cells is trying to track and destroy time-sensitive targets (TSTs). The parametric model (shown in Rational Rhapsody) is used to compute: (1) the response time of the defense system once a time-sensitive target is spotted, (2) the probability that the target is able to fire a missile before being destroyed, and (3) the probability that the target is eventually destroyed before it goes into hiding. The spreadsheet in the figure shows the input and output variables (in red) for this analysis. The output variables indicate if the missile was fired by the time-sensitive target (TST), if the TST was destroyed, and the response time of the overall defense system. A value of 1 indicates success and a value of 0

indicates failure. Input values were generated based on normal probability distributions with given mean and variance. For a defense system alternative (specified number of ground troops, aircrafts, and cells), the analysis shows 36% mean probability that a missile is fired by the target and 100% probability that the target is destroyed before it goes into hiding. The response time for the scenarios is shown using a histogram and a cumulative frequency curve. This analysis was performed using Melody™—SLIM’s Parametric solver tool for Rhapsody.

The Risk analysis tool builds on capabilities available with the Trade study tool, and is available with all Parametric solver tools—**ParaMagic®**, **Melody™**, **ParaSolver™**, and **Solvea™**.

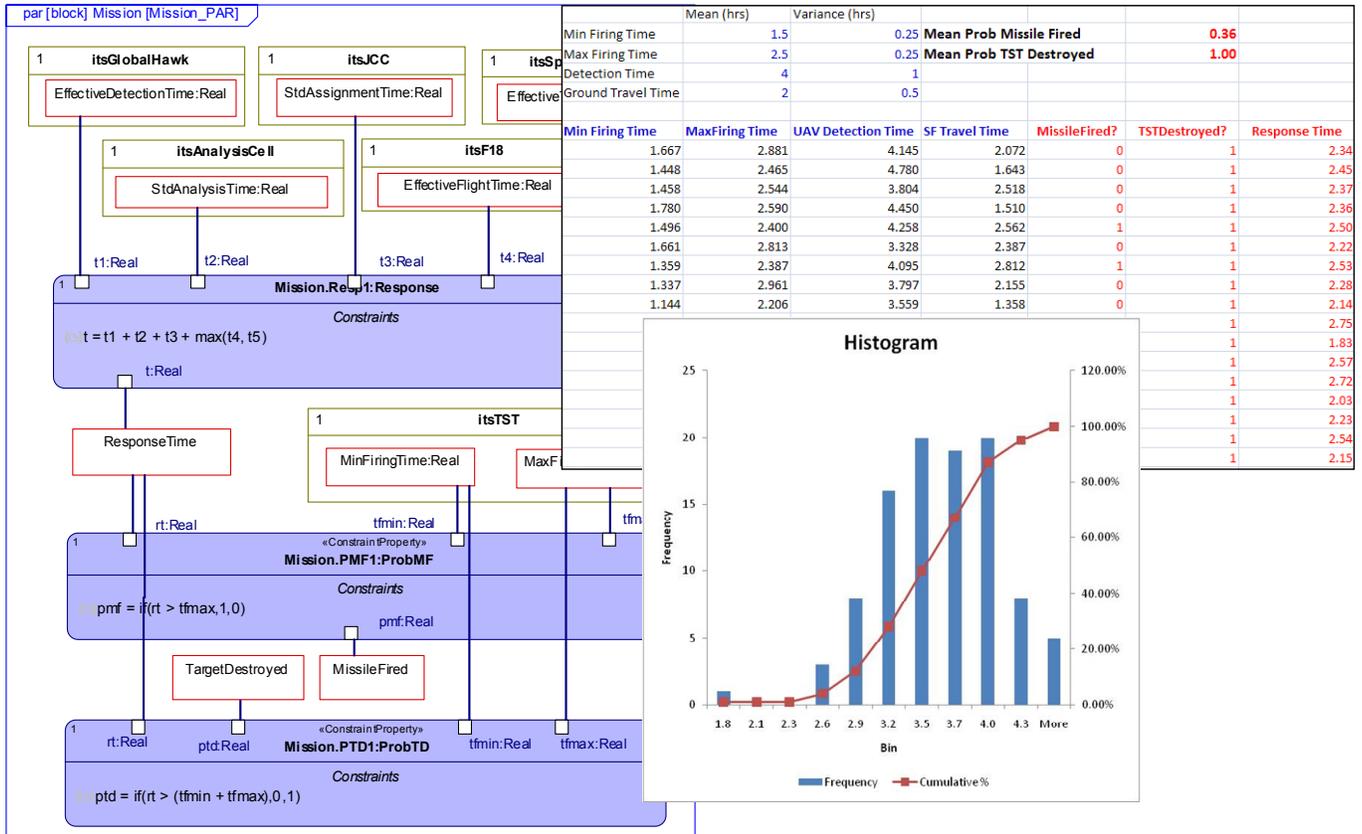


Figure 12: Computing success probabilities & response times for military operations using Melody™ for Rhapsody

### 3.3 Activity execution tool

SysML activity models are used for modeling function-based behavior of systems, such as for modeling mission profiles for multi-mission space systems, or financial and business processes. Execution of activity models provides system engineers and project managers the ability to simulate and analyze system behavior. Figure 13 illustrates a SysML activity model for driving mini-rovers (as shown in the picture). Execution of this activity model generates and executes python code that governs the movement of the rover. The example also demonstrates the use of SysML for system operations. The same SysML model used for designing the system can be used for operating it. This prototype capability is available as a plugin called MyroMagic (Georgia Tech - MyroMagic 2011) for MagicDraw.

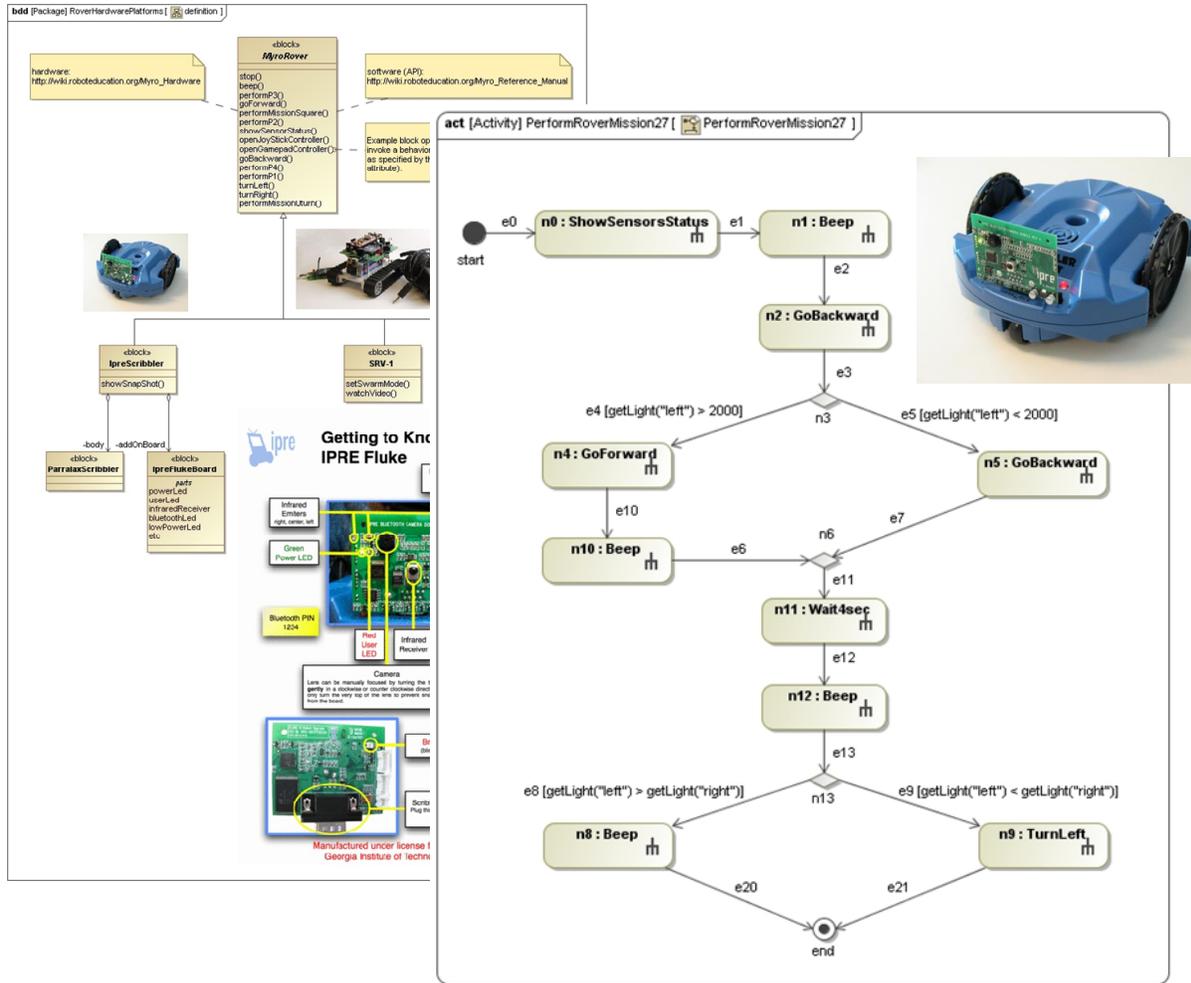


Figure 13: Activity model for driving mini-rovers using the MyroMagic plugin

### 3.4 SysML integration tools

#### 3.4.1 SysML-Excel integration

The SysML-Excel integration tool allows system engineers to connect Excel spreadsheets to SysML models. SysML is used for modeling the system architecture, behavior, and related parametrics, while Excel is used for data storage, presentation, and reporting. With the SysML-Excel integration tool, system engineers can connect system properties to cells in spreadsheets, and automatically read from or write to Excel. Figure 14 shows the Excel Setup utility for the SysML-Excel integration tool. In the Excel Setup utility, properties configured to read from Excel are shown in blue and those configured to write to Excel are shown in red. After setup, users can invoke Excel read/write operations from the SysML model. This integration capability is available with all Parametric solver tools—**ParaMagic**<sup>®</sup>, **Melody**<sup>™</sup>, **ParaSolver**<sup>™</sup>, and **Solvea**<sup>™</sup>.

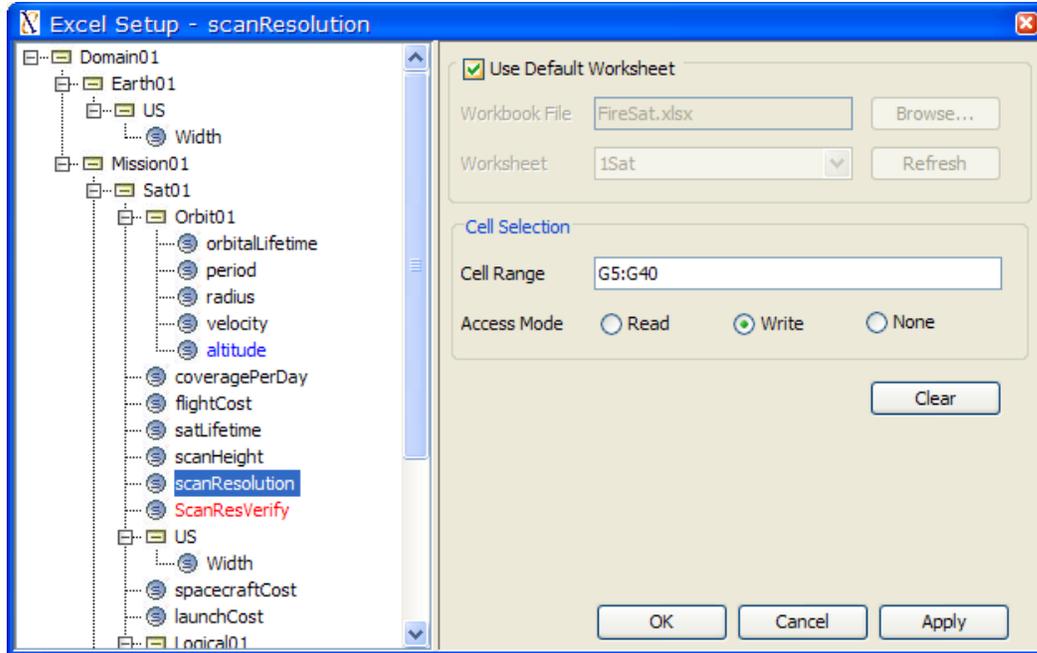


Figure 14: SysML-Excel integration tool - ParaMagic<sup>®</sup>, Melody<sup>™</sup>, ParaSolver<sup>™</sup>, Solvea<sup>™</sup>

### 3.4.2 SysML-MATLAB/Simulink integration tool

The SysML-MATLAB/Simulink integration tool allows users to wrap MATLAB M-files (functions or scripts) as SysML constraint blocks and reuse them in SysML parametric models. During execution, the Parametric solver tool invokes MATLAB/Simulink for executing constraint relationships wrapping M-files and brings results back to SysML or uses it for solving next set of constraint relationships. This capability is illustrated in Figure 10 and is available with all Parametric solver tools—ParaMagic<sup>®</sup>, Melody<sup>™</sup>, ParaSolver<sup>™</sup>, and Solvea<sup>™</sup>.

### 3.4.3 SysML-Mathematica integration tool

The SysML-Mathematica integration tool allows users to wrap user-defined Mathematica functions (.m files) as SysML constraint blocks and reuse them in SysML parametric models, in the same manner as MATLAB M-files. In addition, the Parametric solver tool uses Mathematica as the core solver for executing parametric relationships. This capability is available with all Parametric solver tools—ParaMagic<sup>®</sup>, Melody<sup>™</sup>, ParaSolver<sup>™</sup>, and Solvea<sup>™</sup>.

### 3.4.4 SysML-OpenModelica integration

The Parametric solver tool can also use OpenModelica (free) as a core solver for executing parametric relationships. This capability is available with all three Parametric solver tools—ParaMagic<sup>®</sup>, Melody<sup>™</sup>, ParaSolver<sup>™</sup>, and Solvea<sup>™</sup>. In the future, SLIM will support the use of SysML-Modelica libraries based on the SysML-Modelica transformation specifications (Object Management Group 2010).

### 3.4.5 SysML-CAD (MCAD/ECAD) integration

The SysML-CAD integration tool provides system engineers the ability to connect SysML models to CAD models. This capability is necessary for: (a) integrating CAD-based concept sketches with the system model during preliminary design, (b) transitioning from conceptual to detailed design, and (c) maintaining a consistent view of the system and its components during detailed design, especially for cases where multiple CAD models (MCAD, ECAD) are being concurrently developed for a single component.

Figure 15 illustrates a generalized approach for integrating SysML and design models (e.g. CAD, STK). The integration approach is shown for a single sub-system component that needs to be connected to its detailed definition in the design tool (CAD, STK). Step 1 shows the SysML-based system model (LHS) as being developed by the system engineer, and the detailed design model (RHS) as being developed by a designer. The CAD integration tool automatically generates a view of the CAD model in the SysML modeling environment along with the mapping relationships, as shown in Step 2. Then, the system engineer and the designer can relate the properties of the CAD model view and the system model using SysML parametrics, as shown in Step 3. SysML parametrics is a key to model-based communication between domains. In this case, the parametric relationships represent knowledge that is often lost in documents, spreadsheets, and other ad-hoc modes of communication. Capturing this knowledge in the SysML model makes it traceable. Execution of parametric relationships, using the Parametric solver tool, achieves bi-directional flow of information between the system model and the CAD model, as shown in Step 4. System engineers can visualize and compute dependencies between the system model and domain-specific design models, and automatically verify system requirements on a continuous basis.

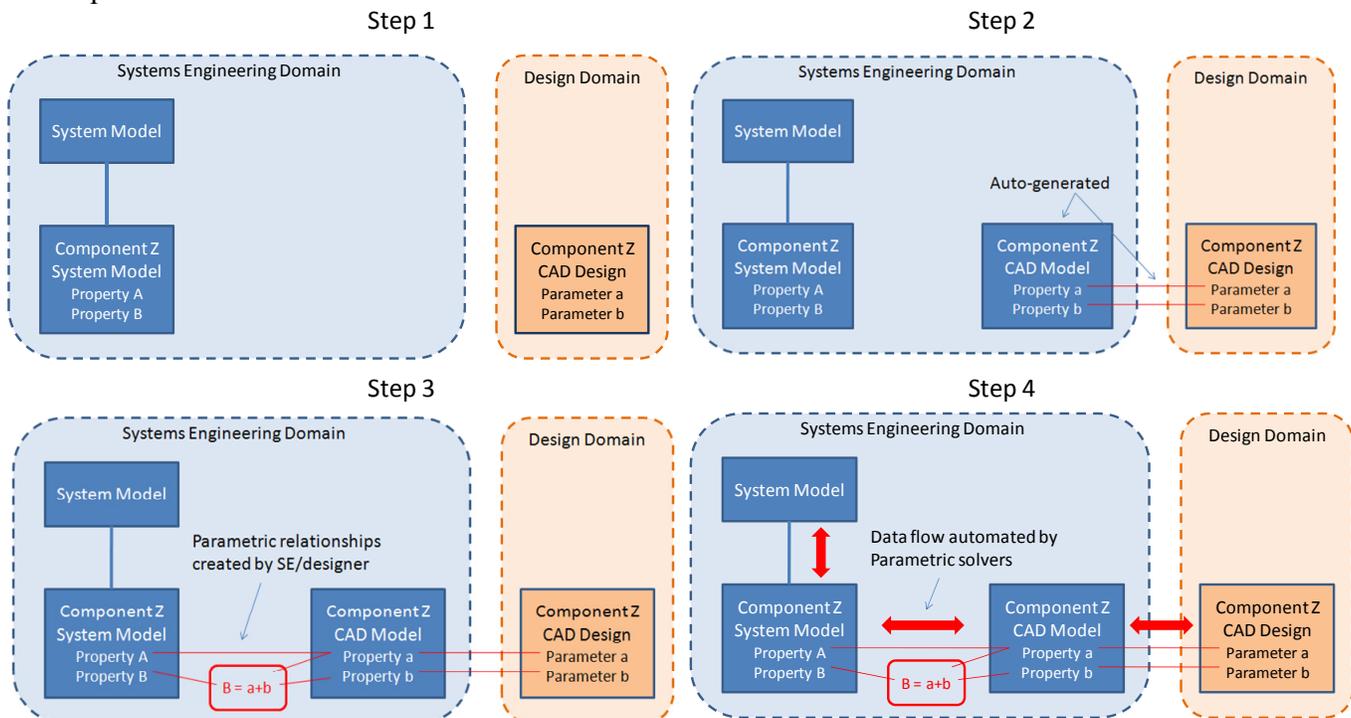


Figure 15: SysML-Design (MCAD/ECAD, STK) integration capability – end-user steps

Figure 16 and Figure 17 illustrate the SysML-CAD integration tool for a Mini-Satellite system. One of the components (BGA) of the Mini-Satellite is being designed in NX (Siemens PLM 2010) CAD tool. The system engineer wants to connect the parameters of the BGA component in the system model to the properties of its detailed CAD definition in order verify component and system-level requirements. The SysML-CAD integration tool automatically generates a SysML view of the CAD model, as shown in Figure 17 (RHS). Then, the system engineer creates parametric relationships between the BGA component in the system model (LHS in Figure 17) and the SysML view of its CAD definition (RHS in Figure 17). The parametric relationships are used to compute the bounding box and the weight of the BGA electronic component from its CAD definition. The Parametric solver tool is used to execute these and other parametric relationships to achieve information flow between the CAD model and the SysML-based system model. Changes made to the BGA CAD model in NX can automatically be reflected in the BGA component of the Mini-Satellite and vice versa. This interaction is based on a live connection between the SysML authoring environment (e.g. MagicDraw) and NX, which is established by SLIM's SysML-CAD integration tool. With this capability, a system engineer can automatically pull the latest updates from the detailed design model into the SysML model, and use this information for performing system-level analyses, verifying requirements, performing trades, and other system V&V tasks.

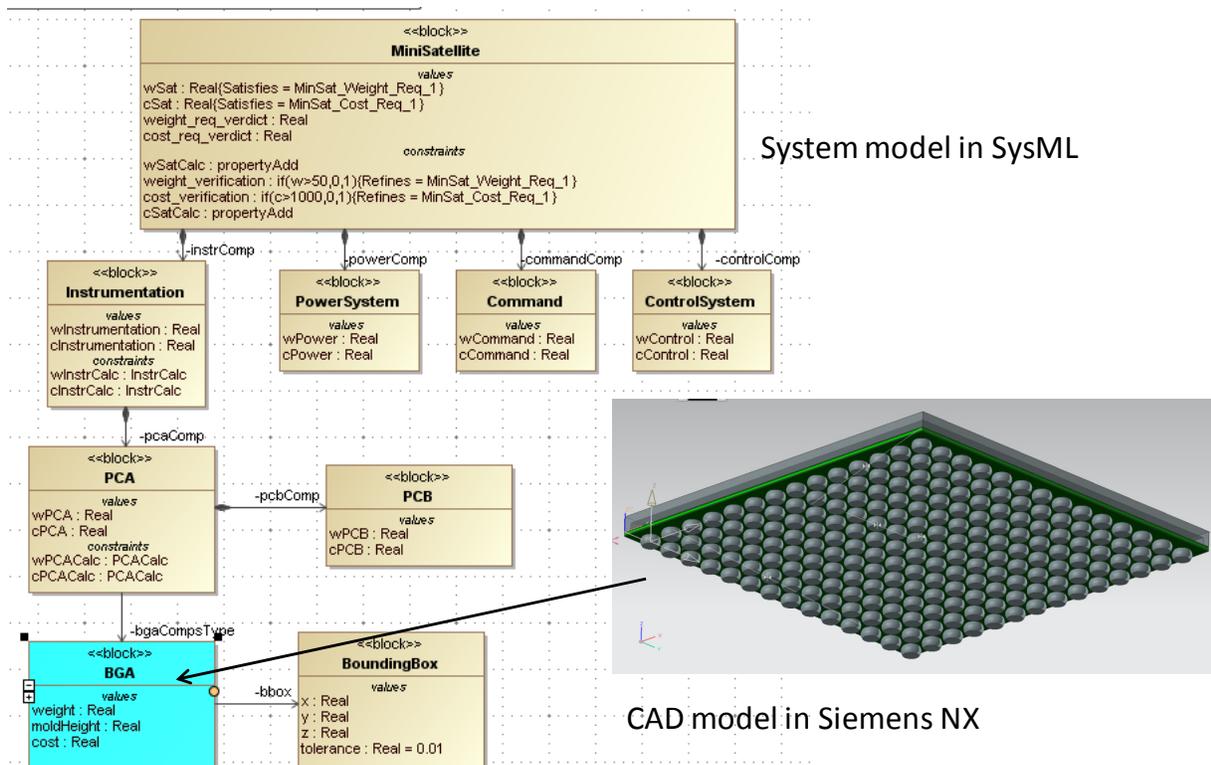


Figure 16: Mini-Satellite system model in SysML. The BGA electronic component and its CAD definition are highlighted.

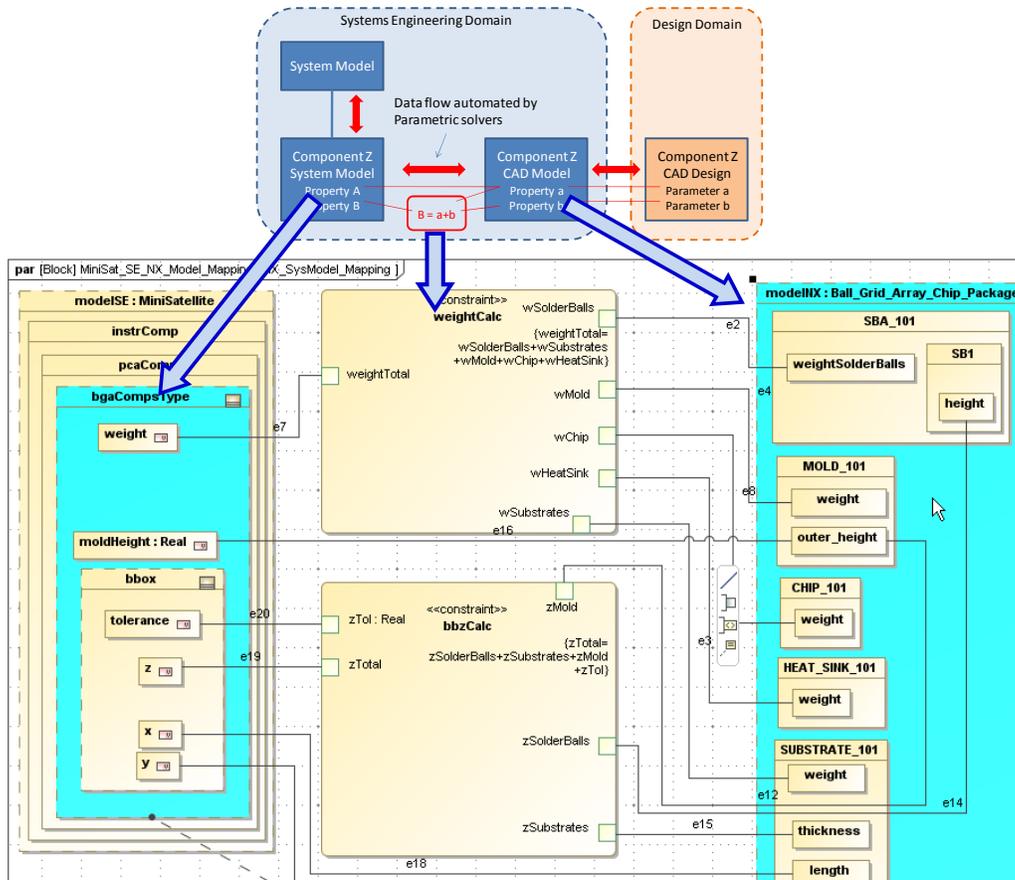


Figure 17: Parametric relationships between the BGA component in the system model and its CAD definition.

The SysML-CAD integration tool will be deployed as Shape™ plugin (InterCAX - Shape 2011) for different SysML authoring environments. Currently, the Shape™ plugin is available for MagicDraw SysML tool and NX CAD tool at a beta-level maturity.

### 3.4.6 SysML-STK integration

The SysML-STK integration tool works on the same integration pattern as shown in Figure 15. It allows system engineers developing aerospace systems to integrate the SysML-based system model with orbit planning and analysis models in STK. SysML parametrics is used to achieve fine-grained synchronization between the SysML-based system model and the STK model—a capability similar to that shown for SysML-CAD models in Figure 17 above.

Figure 18 illustrates the SysML-STK integration plugin (called Apogee™) in action for the FireSat model. Here, Apogee™ is used for sending system-level design parameters from the SysML model to the STK / AGI Component simulation environment, and retrieving STK simulation results back into the SysML environment for requirement verification and further processing. For example, the sensor descriptions of the FireSat are sent to the STK environment and the chain and coverage response results are retrieved from STK simulations. The chain response provides information on the specific time period during which the FireSat has access to specific features-of-interest on the ground. The coverage response provides information on how

much of a specific feature-of-interest (absolute and relative area) would be accessible by the FireSat in a given time interval.

Currently, the Apogee™ plugin (InterCAX - Apogee 2011) is available for MagicDraw SysML tool at a demonstrator-level maturity.

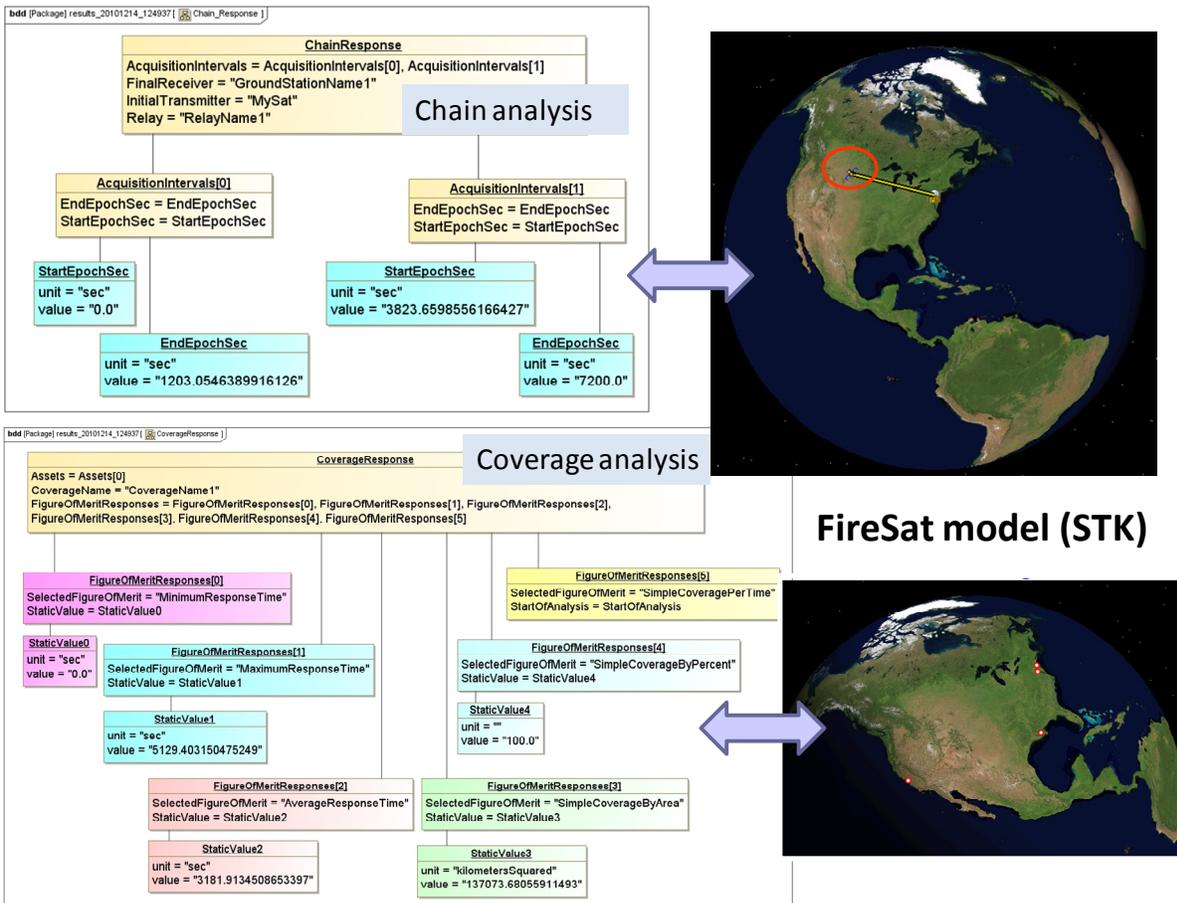


Figure 18: Apogee™ plugin for integrating SysML and STK models

### 3.5 SysML-based libraries

The Parametric solver tool comes with a library of constraint blocks that wrap commonly used math functions, such as trigonometric, logarithmic, exponential, hyperbolic, conditionals (if-else), and aggregate functions. Complex functions can be composed from simpler functions using SysML parametrics. Constraint blocks can also wrap externally-defined functions and scripts, such as MATLAB function and script M-files and Mathematica functions.

The SysML-CAD and -STK integration tools make it possible to reflect libraries of CAD parts and STK design/analysis elements in the SysML environment, which can then be used for composing system models. Libraries of system-level analysis models, such as cost models, performance models, and reliability models can be developed directly using SysML constructs, such as parametrics and activity models) or as SysML constructs wrapping externally-defined models, such as library of constraint blocks wrapping MATLAB routines. In the future, SLIM

will leverage SysML-Modelica transformation specifications (Object Management Group 2010) to make libraries of analytical models in Modelica accessible from the SysML modeling environment.

### 3.6 SysML visualization tools

#### 3.6.1 Panorama

Panorama (Georgia Tech - Panorama 2011) is a SysML-based visualization tool that flattens parametric models for a given system alternative. Figure 19 below illustrates flattened SysML parametric models for a supply chain and for an electronics recycling network.

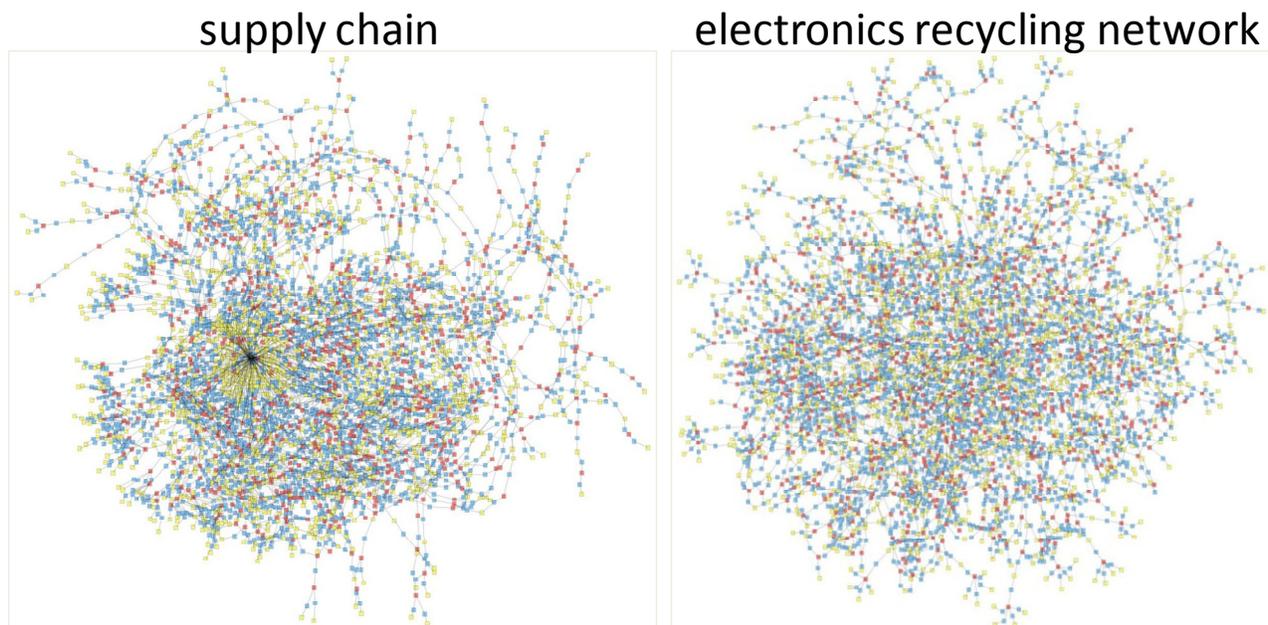


Figure 19: SysML parametric models (flattened) in Panorama for supply chain and electronics recycling network

## 4 Summary & Future Work

This paper (Part 1) presents our vision of SLIM as a collaborative model-based systems engineering workspace that uses SysML as the front-end for orchestrating system engineering activities from the earliest stages of system development. The SysML-based system model serves as a unified, conceptual abstraction of the system independent of the specific design and analysis tools that shall be used in the development process. SLIM provides analysis tools to compute system MoEs and KPPs, orchestrate simulations, perform trades, analyze risk, automatically verify requirements, and perform other systems engineering activities from SysML. It also provides plugins to integrate the system model (in SysML) to a variety of design and analysis tools, such as CAD tools, STK, CAE/FEA tools, Simulink, and math solvers—MATLAB, Mathematica, OpenModelica, and Excel. SLIM enables a consolidated design environment where the SysML-based system model is used for model-based X, where X = design, analysis, reviews, V&V, deployment, and procurement. SLIM tools and capabilities are driven from a SysML authoring environment, and work with models that are configuration controlled in the enterprise PLM environment.

Several key components of the SLIM framework are available for production usage, as described in section 3. Other important capabilities are under-development. In section 2, we have presented the key systems engineering use cases that define the roadmap for SLIM. We invite interested users and organizations to help shape this roadmap, use existing SLIM tools and provide feedback, and contribute to the development of SLIM.

In Part 2 paper—*SLIM Applications*—we present applications of SLIM in aerospace, energy, infrastructure, manufacturing and supply chain, and bank system domains.

## References

AGI (2010). "STK." Retrieved Oct 24, 2010, from <http://www.stk.com/>.

Atego (2010). "Artisan Studio." Retrieved Oct 24, 2010, from <http://www.atego.com/products/artisan-studio/>.

Bajaj, M. (2008). *Knowledge Composition Methodology for Effective Analysis Problem Formulation in Simulation-based Design*. G.W. Woodruff School of Mechanical Engineering, PhD, Georgia Institute of Technology; PhD Dissertation: <http://hdl.handle.net/1853/26639>.

Dassault Systèmes SIMULIA (2010). "Isight." Retrieved Oct 24, 2010, from <http://www.simulia.com/products/isight.html>.

Georgia Tech - MyroMagic (2011). "MyroMagic Plugin for Mobile Rovers." Retrieved Mar 28, 2011, from <http://www.buzztoys.gatech.edu/buzztoys/index.html>.

Georgia Tech - Panorama (2011). "Panorama Plugin for MagicDraw." Retrieved Mar 28, 2011, from <http://www.buzztoys.gatech.edu/buzztoys/index.html>.

IBM (2010). "Rational Rhapsody." Retrieved Oct 24, 2010, from <http://www-01.ibm.com/software/rational/products/rhapsody/designer/>.

INCOSE. "Space Systems Working Group (SSWG) and Challenge Team." 2010, from <http://www.omgwiki.org/MBSE>  
<http://www.incose.org/practice/techactivities/wg/sswg/>.

InterCAX - Apogee (2011). "Apogee plugin for SysML- STK / AGI Components integration." Retrieved Mar 30, 2011, from [www.intercax.com/sysml](http://www.intercax.com/sysml).

InterCAX - Melody (2010). "Melody - SysML Parametric Solver and Integrator for Rational Rhapsody." Retrieved Oct 24, 2010, from <http://www.intercax.com/melody>.

InterCAX - ParaMagic (2010). "ParaMagic<sup>®</sup> - SysML Parametric Solver and Integrator for MagicDraw." Retrieved Oct 24, 2010, from <http://www.intercax.com/sysml>  
<http://www.magicdraw.com/paramagic>.

InterCAX - ParaSolver (2011). "*Artisan Studio ParaSolver*". Retrieved Jan 31, 2011, from [www.atego.com/products/artisan-studio-pararesolver/](http://www.atego.com/products/artisan-studio-pararesolver/)  
[www.intercax.com/sysml](http://www.intercax.com/sysml).

InterCAX - Shape (2011). "*Shape plugin for MagicDraw*." Retrieved Mar 30, 2011, from [www.intercax.com/sysml](http://www.intercax.com/sysml).

InterCAX - Solvea (2011). "*Solvea<sup>TM</sup> - SysML Parametric Solver and Integrator for MagicDraw*." Retrieved Mar 28, 2011, from [www.intercax.com/solvea](http://www.intercax.com/solvea).

Microsoft (2010). "*PowerPoint*." Retrieved Oct 24, 2010, from <http://office.microsoft.com/en-us/powerpoint/>.

Microsoft (2010). "*Visio*." Retrieved Oct 24, 2010, from <http://office.microsoft.com/en-us/visio/>.

NASA (2007). *Systems Engineering Handbook, NASA/SP-2007-6105 Rev 1*. Washington, D.C. 20546.

No Magic (2010). "*MagicDraw SysML modeling tool, version 16.9*." Retrieved Oct 24, 2010, from <http://www.magicdraw.com/sysml>.

Object Management Group (2010). "*SysML and Modelica Integration*." Retrieved Oct 24, 2010, from [http://www.omgwiki.org/OMGSysML/doku.php?id=sysml-modelica:sysml\\_and\\_modelica\\_integration](http://www.omgwiki.org/OMGSysML/doku.php?id=sysml-modelica:sysml_and_modelica_integration).

OSMC (2010). "*OpenModelica 1.5*." Retrieved Oct 24, 2010, from [www.openmodelica.org](http://www.openmodelica.org).

Peak, R., Paredis, C. J. J., Tamburini, D. R., Bajaj, M., Kim, I. and Wilson, M. (2005). *The Composable Object (COB) Knowledge Representation: Enabling Advanced Collaborative Engineering Environments (CEEs), COB Requirements & Objectives (v1.0)*. The Georgia Institute of Technology, Oct 31, 2005. [http://www.eislab.gatech.edu/projects/nasa-ngcobs/COB\\_Requirements\\_v1.0.pdf](http://www.eislab.gatech.edu/projects/nasa-ngcobs/COB_Requirements_v1.0.pdf)

Peak, R. S., Burkhart, R. M., Friedenthal, S. A., Wilson, M. W., Bajaj, M. and Kim, I. (2007). *Simulation-Based Design Using SysML Part 1: A Parametrics Primer*. The Seventeenth International Symposium of the International Council on Systems Engineering, San Diego, California, USA, June 24 -28, 2007. <http://eislab.gatech.edu/pubs/conferences/2007-incose-is-1-peak-primer/2007-incose-is-1-peak-primer.pdf>.

Peak, R. S., Burkhart, R. M., Friedenthal, S. A., Wilson, M. W., Bajaj, M. and Kim, I. (2007). *Simulation-Based Design Using SysML Part 2: Celebrating Diversity by Example*. The Seventeenth International Symposium of the International Council on Systems Engineering, San Diego, California, USA, June 24 -28, 2007. <http://eislab.gatech.edu/pubs/conferences/2007-incose-is-1-peak-primer/2007-incose-is-1-peak-primer.pdf>.

Phoenix Integration (2010). "*ModelCenter*." Retrieved Oct 24, 2010, from [http://www.phoenix-int.com/software/phx\\_modelcenter.php](http://www.phoenix-int.com/software/phx_modelcenter.php).

PTC (2010). "*Windchill*." Retrieved Oct 24, 2010, from <http://www.ptc.com/products/windchill/>.

Siemens PLM (2010). "*NX*." Retrieved Oct 24, 2010, from [http://www.plm.automation.siemens.com/en\\_us/products/nx/](http://www.plm.automation.siemens.com/en_us/products/nx/).

Siemens PLM (2010). "*Teamcenter*." Retrieved Oct 24, 2010, from [http://www.plm.automation.siemens.com/en\\_us/products/teamcenter/](http://www.plm.automation.siemens.com/en_us/products/teamcenter/).

The MathWorks (2010). "*MATLAB*." Retrieved Oct 24, 2010, from <http://www.mathworks.com/products/matlab/>.

Wolfram Research (2010). "*Mathematica*." Retrieved Oct 24, 2010, from <http://www.wolfram.com/products/mathematica/index.html>.

## Biography

**Manas Bajaj, PhD** is the Chief Systems Officer at InterCAX ([www.InterCAX.com](http://www.InterCAX.com)). He has successfully led several government and industry-sponsored projects, including prestigious SBIR awards from NIST and NASA. Dr. Bajaj's research interests are in the realm of SysML and model-based systems engineering (MBSE), computer-aided design and engineering (CAD/CAE), advanced modeling and simulation methods, and open standards for product and systems lifecycle management (PLM/SLM). He is the originator of the *Knowledge Composition Methodology* for simulation-based design of complex variable topology systems. At INCOSE, he is a core team member of three MBSE Challenge Teams – Space Systems, Modeling and Simulation, and Smart Grid. He has authored several publications and won best paper awards. Dr. Bajaj earned his PhD (2008) and MS (2003) in Mechanical Engineering from the Georgia Institute of Technology, and B.Tech. (2001) in Ocean Engineering and Naval Architecture from the Indian Institute of Technology (IIT), Kharagpur, India. He has been actively involved in the development, implementation, and deployment of the OMG SysML standard and the ISO STEP AP210 standard for electronics. He is a Content Developer (author) for the OMG Certified Systems Modeling Professional (OCSMP) certification program, and coaches organizations on SysML and MBSE. Dr. Bajaj is a member of INCOSE and a contributor in OMG and PDES Inc. working groups.

**Dirk Zwemer, PhD** is President and CEO of InterCAX ([www.InterCAX.com](http://www.InterCAX.com)). Dr. Zwemer has over thirty years experience in the electronics industry with Bell Labs, Exxon, ITT, SRI Consulting and other organizations. He is the author of three patents and multiple technical papers, trade journal articles, and market research reports. Prior to joining InterCAX, he held positions as VP Technology, VP Operations, and President of AkroMetrix LLC, a leader in mechanical test equipment and services for the global electronics industry, where he was actively engaged in market channel development, advertising and tradeshow, and licensing. He received a PhD in Chemical Physics from UC Berkeley and an MBA from Santa Clara University. Dr. Zwemer provides strategic consulting for customers and shows how to apply SysML in a wide variety of domains. He is certified systems modeling professional (OCSMP Model Builder Advanced) and a member of the Smart Grid Challenge Team within the INCOSE MBSE Initiative.

**Russell Peak, PhD** is a Senior Researcher at the Georgia Institute of Technology where he serves as Director of the Modeling & Simulation Lab ([www.msl.gatech.edu](http://www.msl.gatech.edu)) and Associate Director of the Model-Based Systems Engineering Center (MBSE-C) ([www.mbse.gatech.edu](http://www.mbse.gatech.edu)). Dr. Peak specializes in knowledge-based methods for modeling & simulation, standards-based product lifecycle management (PLM) frameworks, and knowledge representations that enable complex system interoperability. Dr. Peak originated the multi-representation architecture (MRA)—a collection of patterns for CAD-CAE interoperability—and composable objects (COBs)—a non-causal object-oriented knowledge representation. Dr. Peak leads the INCOSE MBSE Challenge Team for Modeling & Simulation Interoperability<sup>†</sup> with applications to mechatronics (including mobile robotics testbeds) as a representative complex systems domain. He is a Content Developer (author) for the OMG Certified Systems Modeling Professional (OCSMP) program, and coaches organizations on SysML and MBSE. Dr. Peak earned his Ph.D., M.S., and B.S. in Mechanical Engineering from Georgia Institute of Technology.

**Alex Phung** is a senior research engineer at InterCAX LLC. He has 12 years of experience in a wide range of software development technologies. Mr. Phung is a certified systems modeling professional (OCSMP Model Builder Intermediate) and has over five years of experience in UML and SysML software development. Mr. Phung earned his M.S. and B.S. in Mechanical Engineering from Texas A&M University.

**Andrew Scott** is a research engineer at InterCAX LLC. He is an experienced SysML modeler and software developer. Mr. Scott earned his B.S. in Mechanical Engineering at Georgia Institute of Technology.

**Miyako Wilson** is a research engineer at the Manufacturing Research Center at Georgia Institute of Technology. She specializes in object-oriented software development. Ms. Wilson earned her M.S. in Mechanical Engineering at Georgia Institute of Technology and B.S. in Mechanical Engineering at Arkansas State University.

---

<sup>†</sup> <http://www.omgwiki.org/MBSE/doku.php?id=mbse:modsim>