

Maestro – A model-based systems engineering environment for complex electronic systems

Copyright © 2012 InterCAX LLC and Sandia National Laboratories. Published and used by INCOSE with permission.

Manas Bajaj^{1*}, Andrew Scott¹, Douglas Deming², Gregory Wickstrom², Mark De Spain²,
Dirk Zwemer¹, Russell Peak¹

1. InterCAX LLC
75 5th Street, Suite 312,
Atlanta, GA 30308 USA
www.intercax.com

2. Sandia National Laboratories
1515 Eubank SE
Albuquerque, NM 87123 USA
www.sandia.gov

Abstract

In this paper we present Maestro, a model-based systems engineering (MBSE) environment for design and simulation of complex electronic systems using Orchestra—a simulation tool developed at Sandia National Laboratories. Maestro is deployed as a plugin for MagicDraw and uses Orchestra domain-specific language (DSL) which is based on SysML. Maestro enables a model-based design and analysis approach that replaces the traditional document-based systems engineering process. It provides a unified graphical modeling environment to domain experts who have had to depend on drawing tools for defining system architecture and manual transcription of system topology in creating complex simulation models.

1 Introduction

The OMG Systems Modeling Language (SysML) has provided a solid foundation for model-based engineering of complex systems. However, every system domain is different. The stakeholders are different. Adoption of model-based systems engineering (MBSE) in different system domains will depend on the availability of rich *domain-specific* MBSE applications that provide stakeholders the tools to architect and analyze systems in that domain.

An example of such a domain is the design and simulation of complex electronic systems. Orchestra is a general-purpose system-level modeling tool developed at Sandia National Laboratories for discrete event simulation of embedded systems. It captures system functionality in multiple formalisms, including software, microcode and digital logic.

Maestro is an MBSE environment for design and simulation of complex electronic systems using Orchestra. Maestro is being developed by InterCAX as part of a collaboration effort with Sandia National Laboratories. The goal of Maestro is to provide a rich, model-based design and analysis environment that replaces the traditional document-based systems engineering (DBSE) approach.

* Corresponding author: Dr. Manas Bajaj, email: manas.bajaj@intercax.com, phone: +1-404-592-6897, x101.

A critical advantage of MBSE in this application is knowledge capture of the process of converting graphical design information into executable Java code. The need for close familiarity with a Java design environment is a barrier to wider adoption of the simulation tool. The facilitation of this process, however, must not obstruct the analyst’s ability to modify the design for analysis, e.g. to substitute lower or higher fidelity component models where appropriate. A graphical MBSE language supports this step effectively.

In this paper, we present our technical effort leading to the development of Maestro in the following manner. In section 2, we present the DBSE approach that was the motivation for developing Maestro; in section 3, we present the MBSE challenge problem as a precursor to Maestro; in section 4 we present the use cases, design flow, and the features of Maestro; and in section 5, we present a summary of this paper and next steps in our ongoing effort.

2 Document-based Systems Engineering (DBSE) Approach

Figure 1 illustrates a simple view of the document-based systems engineering process for design and analysis of complex electronic systems before Maestro. The figure shows the process as a SysML activity model. Actions represent the different steps in the process, vertical swim lanes represent the stakeholder (designer, analyst) or the software responsible for the actions in the swim lanes, and the object flows indicate model/data flow between different steps.

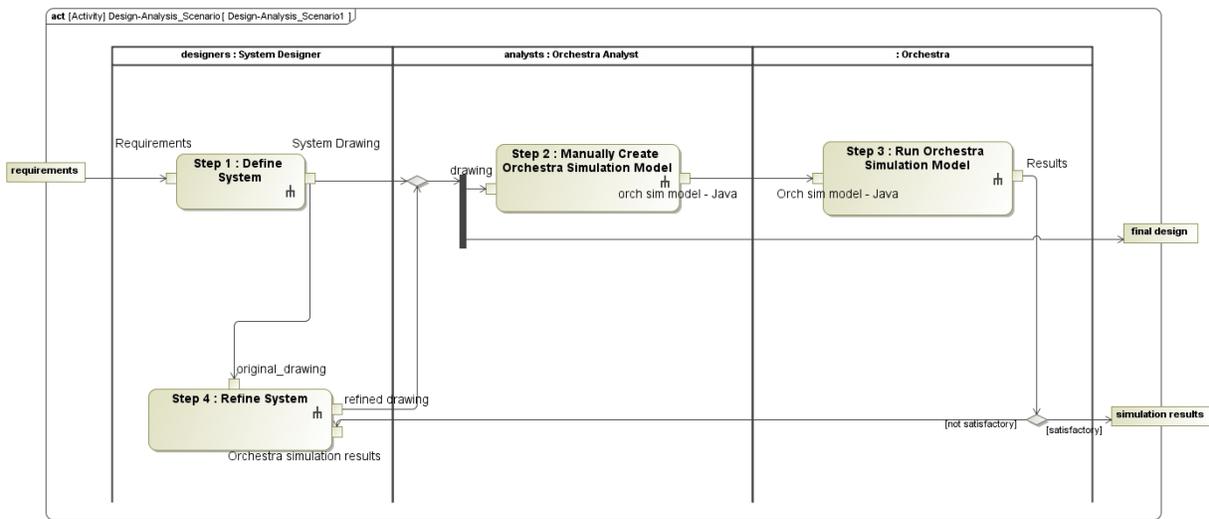


Figure 1: Document-based systems engineering of electronic systems before Maestro

The process started with the definition of the system as a drawing in Visio or PowerPoint by the system designer. Figure 2 illustrates drawings of an electronic system created by the system designer. System model defined in a drawing tool has little intelligence about the components, interfaces, connections, behavior, and requirements. For example, attributes of components, interfaces, and connection are available as text-based notes and connections between components cannot be validated for compatibility of connected ends (interfaces) and the types of signal flowing at the connections. Lack of an integrated model-based system design description makes it impossible to define and automate model transformations for generating analysis and optimization models in different electrical and mechanical analysis tools, generating reports for design reviews, and performing topological and parametric trades at the system-level. Multiple

disconnected views of the system model in different design and analysis environment often leads to miscommunication among the domain experts potentially risking system failure and scheduled delivery and maintenance for mission-critical applications.

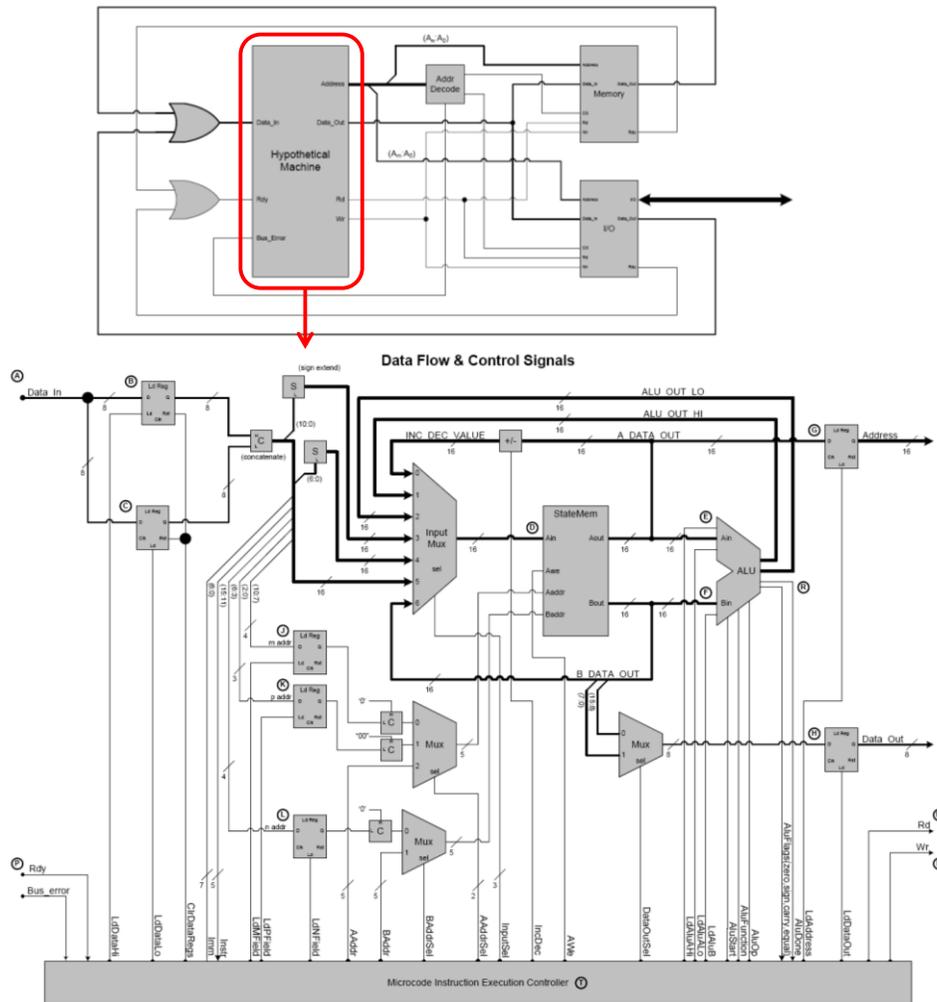


Figure 2: Drawings of an electronic system created by the system designer. The internal structure of the Hypothetical Machine sub-system (top) is shown (bottom).

The second step in the traditional DBSE process is the manual creation of Orchestra analysis models (Java-based) from the system design drawings. An Orchestra analyst manually reviews each of the components, their interfaces, and the interconnections presented across several design drawing documents and creates a corresponding Java class structure for the Orchestra analysis model. This approach poses significant challenges:

1. The Orchestra Java API has a significant learning curve for potential Orchestra users, esp. because not all analysts are programmers. To some extent, this inhibits wider adoption of Orchestra's high-end simulation capabilities.
2. For specific sub-systems, Orchestra analysts refine the design description directly in Java. So, there are portions of detailed system design that are non-existent in the system design

definition but have a corresponding simulation model. This problem of inconsistencies between design and simulation models is further aggravated when other simulation models are derived from the system model.

3. The design-analysis change cycle is significantly large. System designers modify design drawings and describe these changes in ad-hoc forms such as documents, emails, and drawing notes, which may be potentially misinterpreted. Analysts view these updates and manually modify Java code to update the Orchestra simulation model.
4. In the absence of a detailed design-analysis transformation specification, managing simulation models of different fidelities and scope is extremely challenging.

The third step in the process is to execute the simulation models in Orchestra and provide feedback to designers on system performance. Based on the results, designers manually modify the system-level design drawings, spending significant time and resources in maintaining consistencies between views of the same sub-system/component in different drawings. Further, domain-specific detailed design models, such as ECAD and MCAD models, and other corresponding analysis models have to be manually updated. This process iterates until a satisfactory set of design alternatives is selected for further refinement (next design phase).

3 Challenge Problem: Model-Based Systems Engineering (MBSE) Approach for Electronic Systems

In the context of our effort, the DBSE approach outlined above was not suitable for dealing with increasing system complexity and stringent requirements for system performance and reliability. It was evident that a model-based systems engineering approach has to be developed. Rich model-based representation of system design drawings was the first challenge problem that we took on. The goal of this challenge problem was to investigate the use of OMG Systems Modeling Language (SysML) as a foundation for model-based representation of complex electronic systems, and to investigate approaches for generating Orchestra simulation models from SysML-based design models. Figure 3 and Figure 4 illustrate SysML models of the Hypothetical Machine system (Figure 2) developed as part of our challenge problem in MagicDraw SysML modeling tool. Figure 3 is a SysML internal block diagram showing the internal structure (components, ports, connectors) of the Hypothetical Machine block, and Figure 4 illustrates the successive decomposition of the Hypothetical Machine system from the top-level system block to the leaf level sub-system.

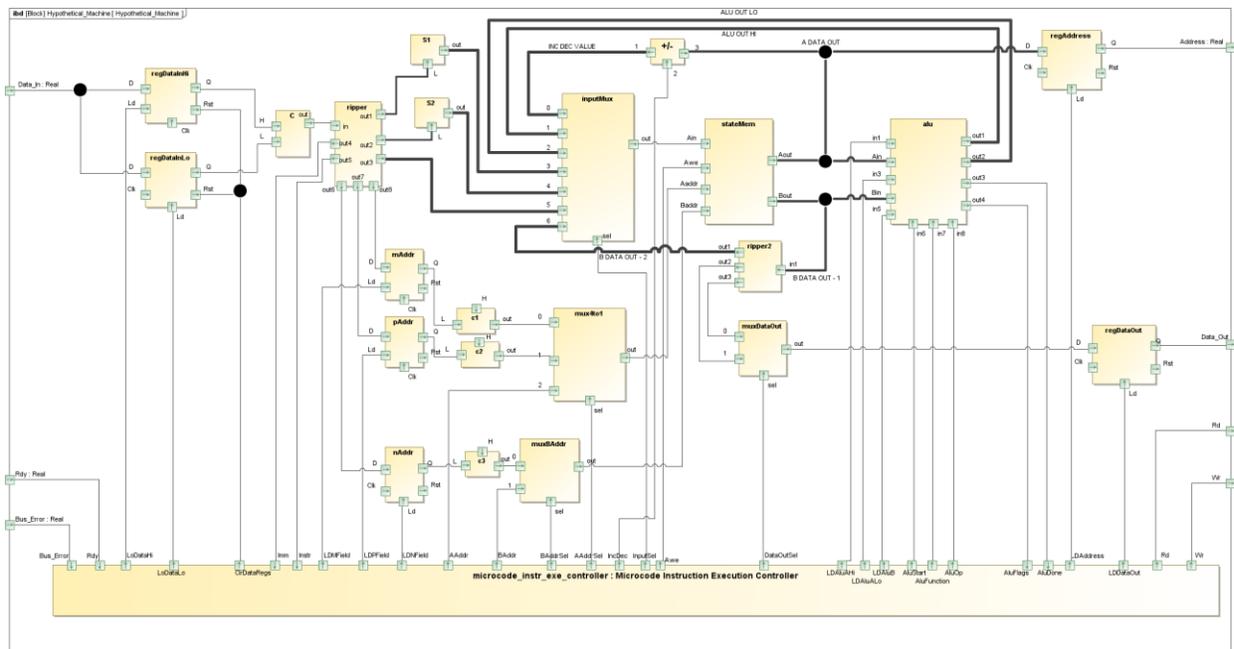


Figure 3: Internal structure of the Hypothetical Machine system in SysML – shown as an IBD in MagicDraw.

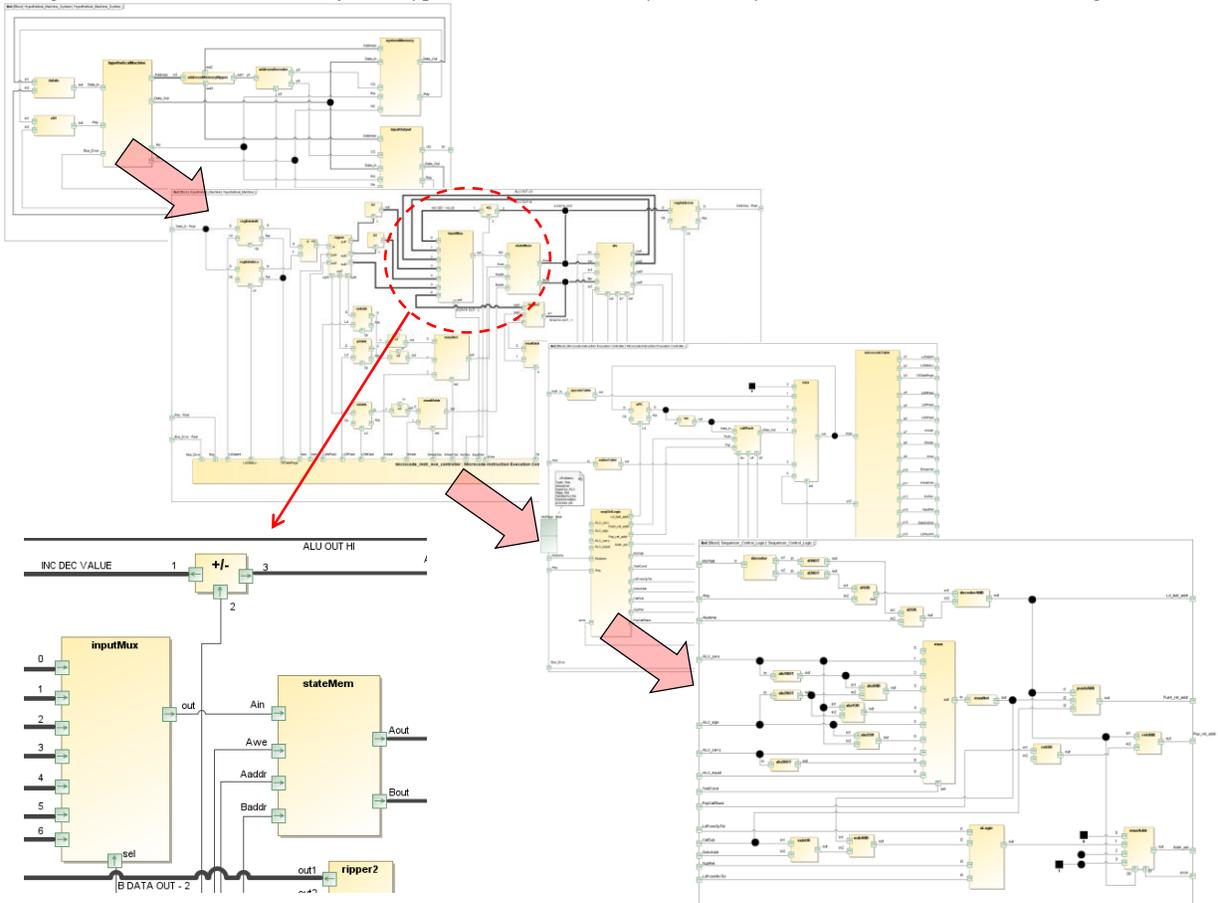


Figure 4: System decomposition is an integral part of the unified and consistent SysML model as opposed to a set of isolated drawings. The figure shows successive decomposition of components (top left to bottom right).

In contrast to the traditional DBSE approach, representation of the electronic system design using SysML offered immediate advantages as stated below.

1. The SysML-based design model is a single integrated and consistent model of the electronic system. A SysML model contains definitions of the system and sub-system components and their interconnections, as shown in the LHS model tree in Figure 5 below, in addition to multiple diagrams (views) of the system model, as shown on the RHS of Figure 5. Unlike system drawings in the DBSE approach, diagrams are views of the system model and hence self-consistent. Changes made to system and sub-system components are reflected across all diagrams (views). Part are defined once in the model and reused multiple times in the definition of the system. As a result system engineers do not have to spend valuable resources in maintaining consistency across drawings or representing the same part multiple times in same or different drawings.

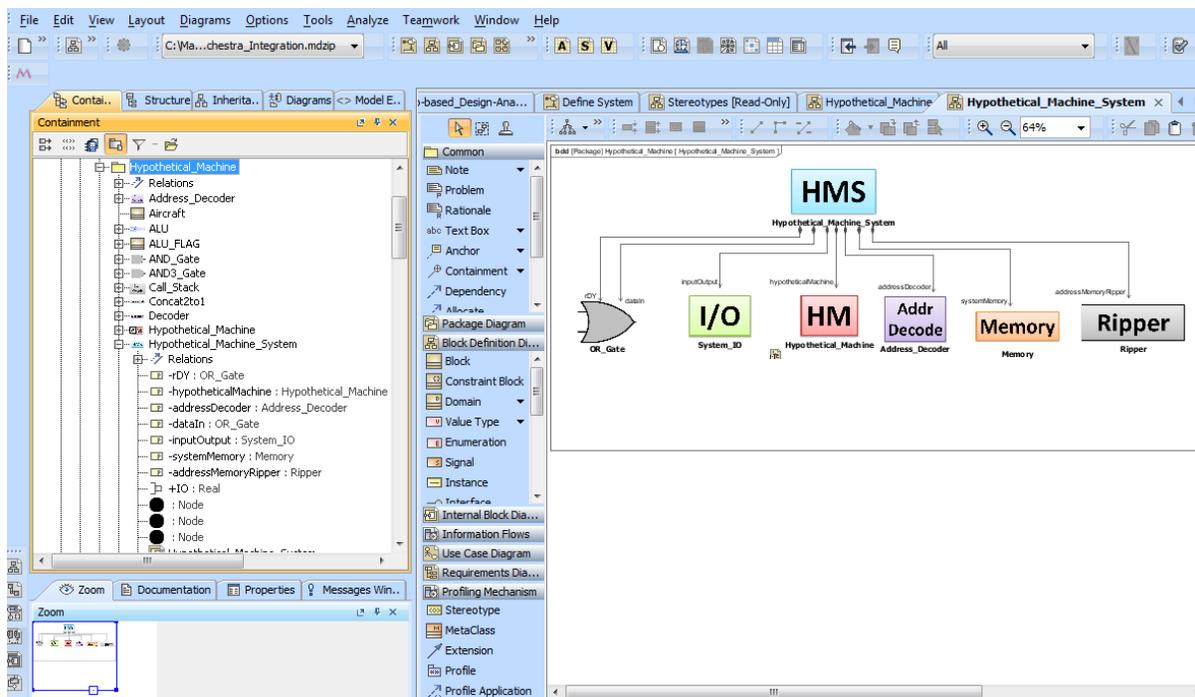


Figure 5: Hypothetical Machine system model in MagicDraw – model and diagram views

2. The decomposition of the system into sub-systems and eventually to leaf-level components is inherent in the system definition itself. System engineers can automatically explore the design model across the hierarchy without having to co-relate symbols in multiple drawing documents. As illustrated in Figure 4, system engineers can bring up the internal structure of any system component and explore recursively.
3. Characteristics (parts, references, attributes) of system components are captured as part of the system model—see characteristics for the Hypothetical_Machine_System block in the LHS model tree in Figure 5.
4. System and component interfaces and connections can be modeled explicitly, as shown in the zoomed view in Figure 4. SysML provides strong rules for compatibility of

4 Maestro – a DSL environment for MBSE of electronic systems

Maestro is a domain-specific language (DSL) environment for model-based systems engineering of complex electronic systems in the context of our effort. Maestro is deployed as a DSL plugin for MagicDraw. The primary goal of Maestro (at this stage of our effort) is to provide a modeling and simulation environment for system designers and Orchestra analysts. While the SysML standard and MagicDraw tool provide the foundational language and modeling environment respectively for general purpose systems modeling, Orchestra DSL profile (section 4.2) and Maestro provide a domain-specific layer on top for model-based design of complex electronic systems with detailed simulation in Orchestra. In this section, we present Maestro in the following manner. In section 4.1 we present the use cases and overall design-analysis flow using Maestro; in section 4.2 we present the Orchestra DSL profile; and in section 4.3 we illustrate the end user capabilities of Maestro.

4.1 Use Cases

Figure 7 illustrates the use cases of Maestro design-analysis environment. The system designers and Orchestra analyst are the main stakeholders. Maestro provides system designers the capability to define and update SysML-based system design model, generate Orchestra simulation models from the SysML-based design model, execute Orchestra models, and verify system requirements in an automated manner. For the Orchestra analyst, Maestro provides a graphical modeling environment for creating Orchestra analysis models (based on Orchestra DSL), generate executable Java-based Orchestra analysis models, and visualize and verify simulation results.

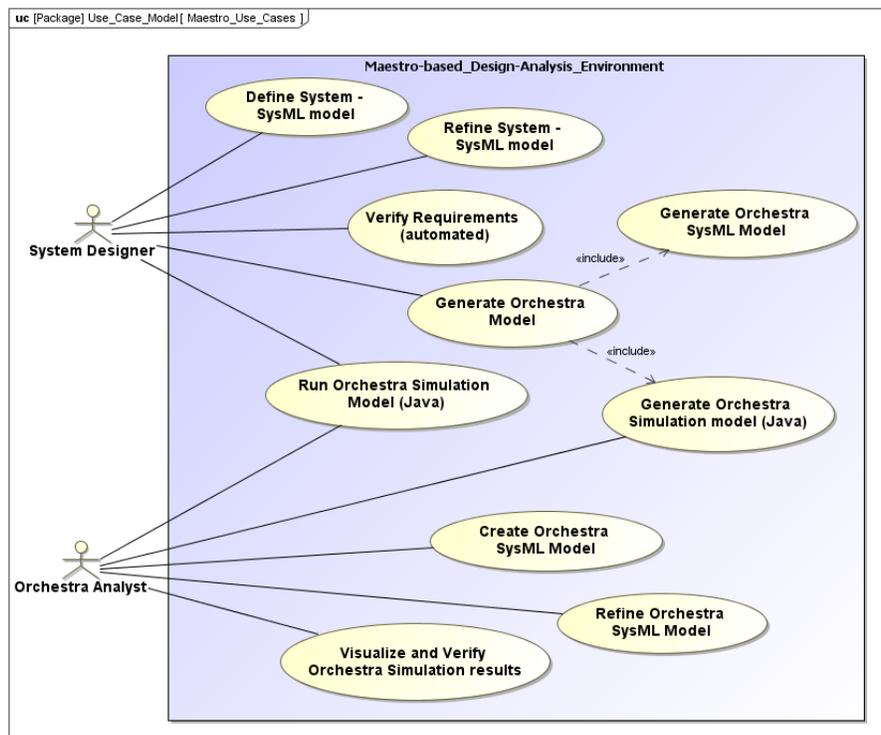


Figure 7: Maestro use cases

Figure 9 below illustrates a simple view of the MBSE process (as a SysML activity diagram) for designing complex electronic systems using Maestro. System designers create SysML-based design model based on requirements, which are then used to generate Orchestra models in the Maestro environment and XML and Java-based representation of Orchestra models. The Java-based models are executed using Orchestra and the simulation results are used to refine the system model or move to the next design phase. Alternatively, the process can initiate directly with the Orchestra analyst using Maestro to manually create Orchestra analysis models from which Java-based models are generated and executed using Orchestra. In this scenario, Maestro provides a graphical modeling environment for creating Orchestra analysis models.

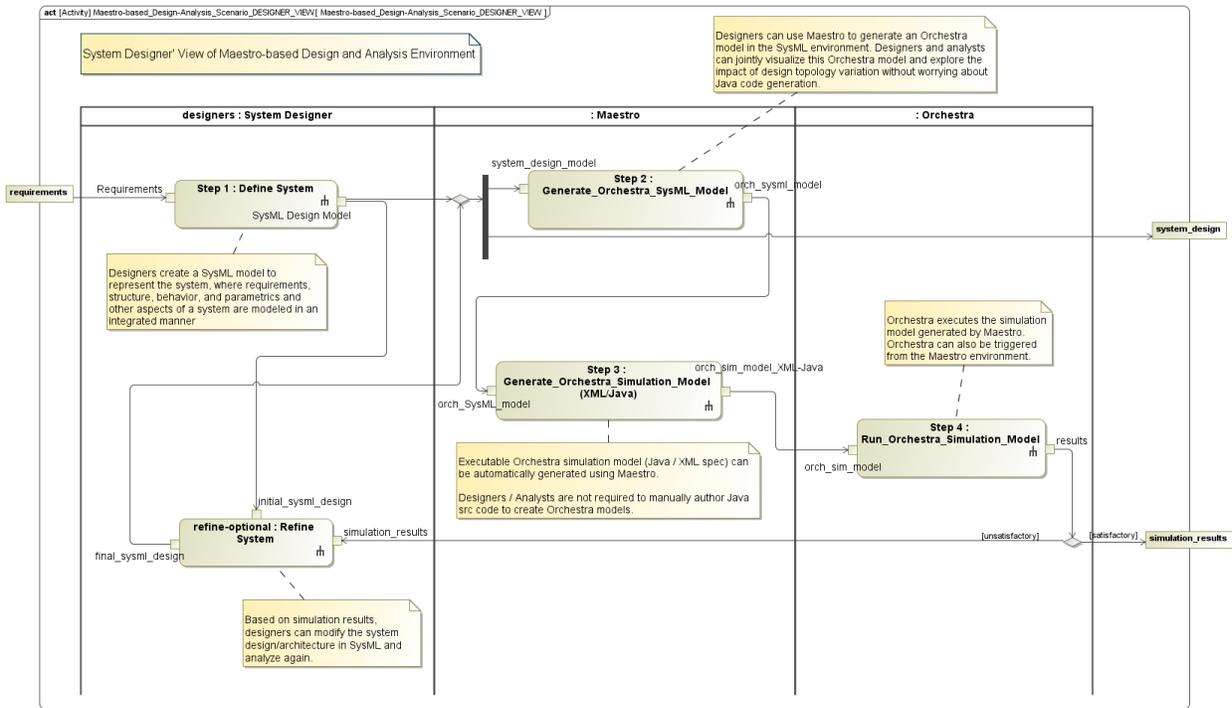


Figure 8: MBSE of electronic systems with Maestro – simple view

4.2 Orchestra DSL profile

The Orchestra DSL profile provides the high-level concepts and related constraints for representing Orchestra analysis models as well as the symbols for visualizing the concepts. The concepts in the Orchestra DSL profile are represented using stereotypes specializing SysML stereotypes. Figure 9 shows the high-level concepts in the Orchestra DSL profile used by Maestro. Module is an atomic concept in Orchestra and is represented by a stereotype that specializes the SysML block stereotype. PortType concept is used for classifying ports, and SimpleConnectionType and ComplexConnectionType concepts are used for classifying connectors between module components. An Orchestra Port concept represents an interaction point for a module through which it communicates with other modules in the context of a system module. An Orchestra Port specializes the SysML FlowPort concept. A ModuleComponent is the usage of a module in the context of another module. Hence, the ModuleComponent concept specializes the SysML PartProperty concept. For Orchestra modeling, we have identified two main types of connections that can be created between module components, namely

SimpleConnection and ComplexConnection. The SimpleConnection concept represents binary connectors, i.e. connectors between ports of two module components, and the ComplexConnection concept represents n-ary connectors, i.e. connectors between ports of three or more module components.

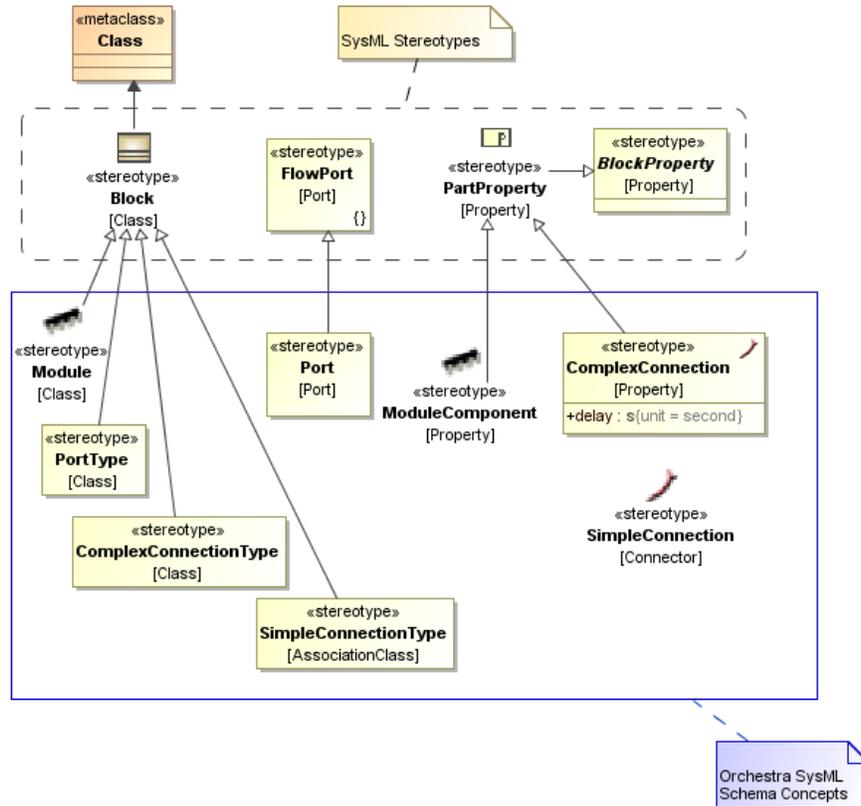


Figure 9: Orchestra DSL profile used by Maestro. The figure shows the

The Orchestra DSL profile provides the basic language constructs for representing Orchestra elements. The DSL profile is used for developing libraries of modules, port types, simple and complex connection types that can be shared as modules and used by system designers in developing problem-specific Orchestra analysis models. The ability to develop and share libraries of commonly used concepts is a key capability provided by Maestro. The Orchestra DSL profile also defines two diagram types, namely the Module Hierarchy Diagram (MHD) and the System Builder Diagram (SBD). MHD specializes the SysML block definition diagram (BDD) and is used for defining and viewing the hierarchy (decomposition and specialization) of modules, and module attributes and ports. SBD specializes the SysML internal block diagram (IBD) and is used for defining and viewing the connections and related data flows between ports of module components. Like an IBD, a SBD is defined in the context of a module and represents the internal structure of that module.

4.3 Maestro DSL Plugin - Modeling and Simulation Environment

In this section, we illustrate the end user capabilities of Maestro, deployed as a DSL plugin for the MagicDraw SysML modeling environment.

After installing the Maestro plugin for MagicDraw, users can create a new Orchestra model, as shown in Figure 10. Maestro provides project initialization templates as well perspectives that control the specific toolbars that a user can view in a perspective.

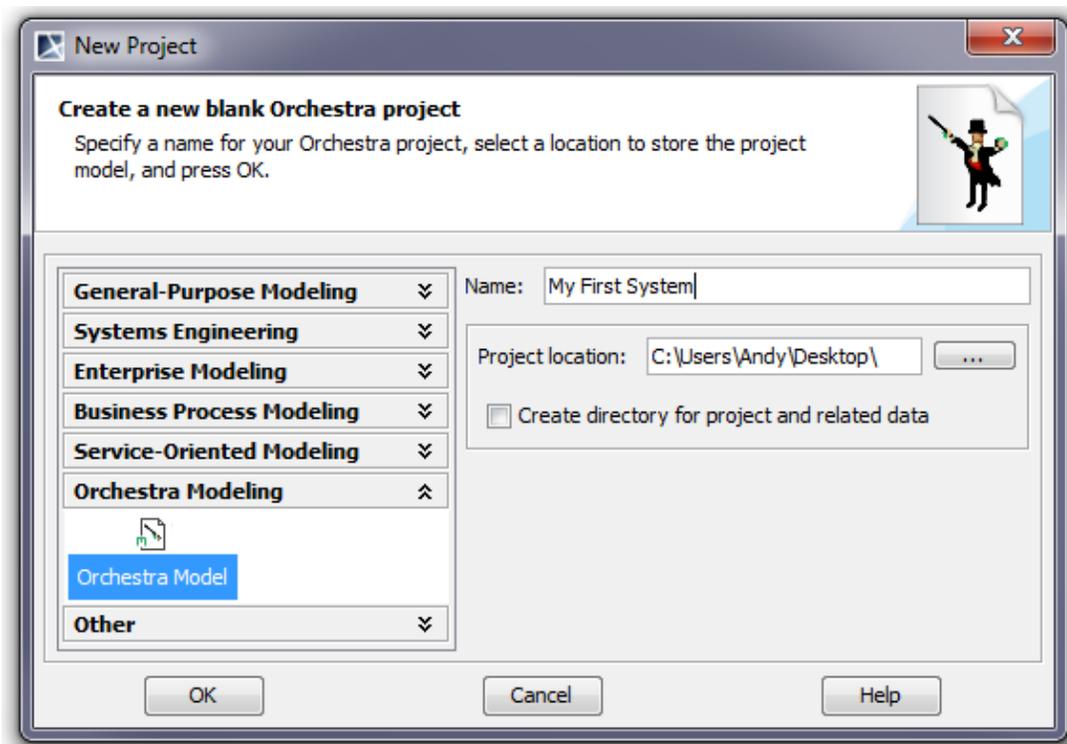


Figure 10: Creating a new Orchestra modeling project using Maestro

Figure 11 illustrates a module hierarchy diagram (MHD) created using Maestro. As illustrated in the figure, Maestro provides a diagram toolbar for MHDs that contains menu items for defining Orchestra concepts, such as Module, Port, and Composition. The Common toolbar contains menu items for defining concepts common to all Orchestra diagrams, such as Note and Text Box.

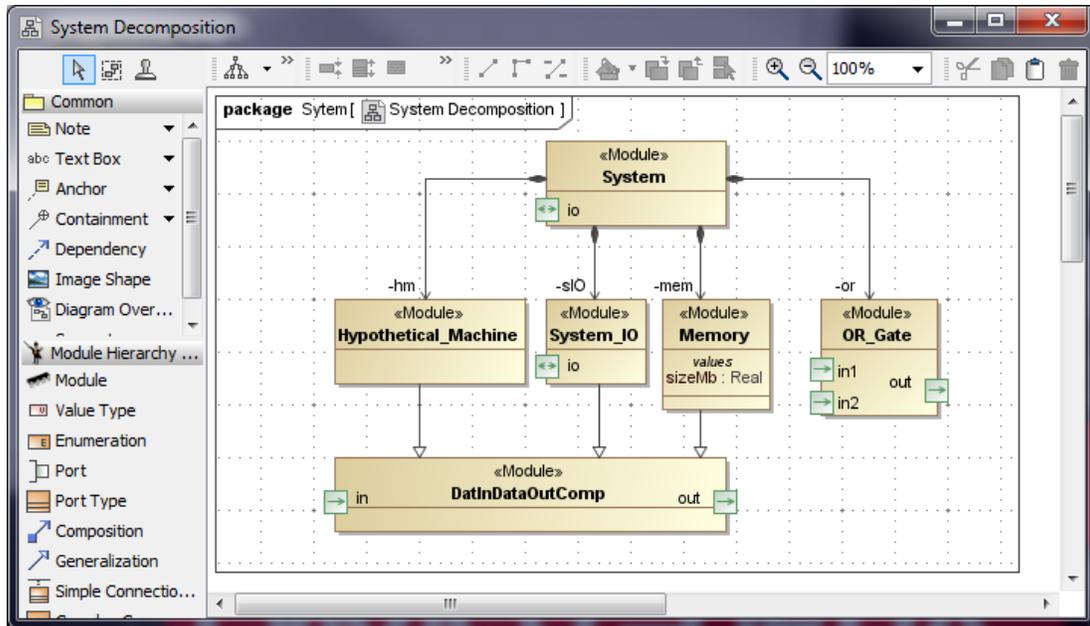


Figure 11: Module Hierarchy Diagram (MHD) showing the decomposition of an example System module

Figure 12 below illustrates a system builder diagram (SBD) created for the System module. The diagram shows four module components (usages of modules), their ports, and the connections (SimpleConnection and ComplexConnection) between the ports. The data flows on the connections are not shown. Similar to MHDs, Maestro provides diagram toolbars that contain menu items for creating Orchestra concepts on SBDs, such as Module Component, Input/Output ports, SimpleConnection and ComplexConnection.

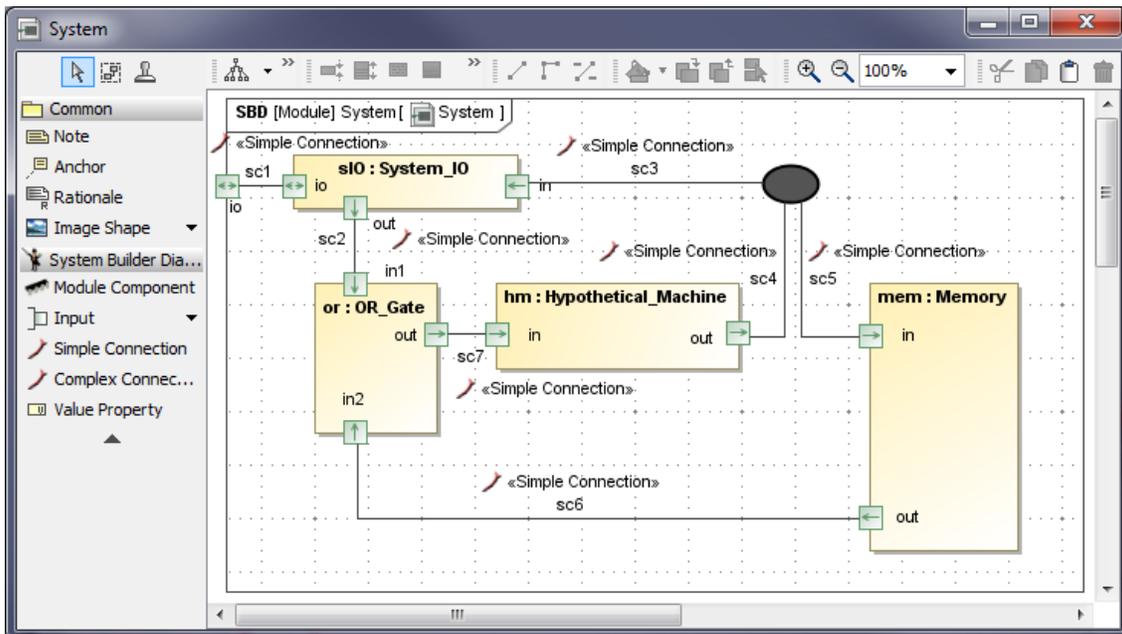


Figure 12: System Builder Diagram (SBD) showing the connections between ports of module components in the context of the example System module.

Figure 13 illustrates the XML-based representation of the Orchestra analysis model generated from the Maestro model.

```

<!--
  Orchestra Maestro XML (O2)
  Generated: 31/04/2011 at 02:29 PM
  Maestro v1.0
-->
<OrchestraXML
xsi:schemaLocation="http://sandia.gov/ns/O2 O2.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns="http://sandia.gov/ns/O2">
  <Module name="System_IO" id="Sytem::System_IO">
    <Generalization module="Sytem::DatInDataOutComp"></Generalization>
    <Port type="" name="io"></Port>
  </Module>
  <Module name="System" id="Sytem::System">
    <ModuleComponent type="Sytem::System_IO" name="sIO"></ModuleComponent>
    <ModuleComponent type="Sytem::Memory" name="mem"></ModuleComponent>
    <ModuleComponent type="Sytem::OR_Gate" name="or"></ModuleComponent>

```

Figure 13: XML-based representation of the Orchestra analysis model generated from the Maestro model

Maestro provides a simple way to edit values of attributes owned by module, port, and simple and complex connections. Figure 14 below illustrates the Maestro value editor which when invoked on a module lists the attributes of component modules, ports, and connections through the entire decomposition hierarchy. If the value field is left blank, Maestro uses the value in the default value field.

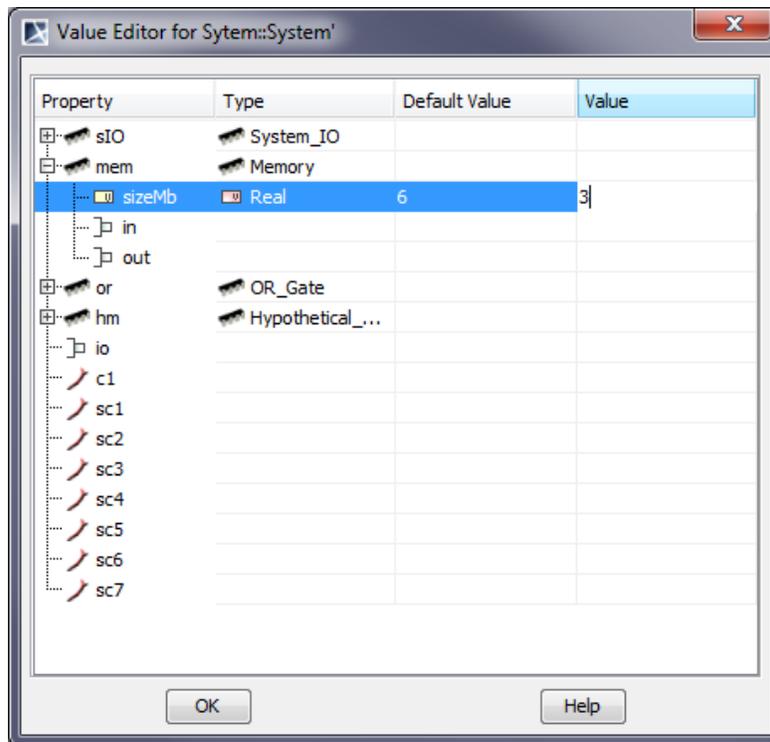


Figure 14: Maestro value editor

Maestro makes it easy to share module packages as libraries which can be imported in other projects and used for creating more complex modules. As shown in Figure 15, users can easily select a module package and share it as a library which can then be imported in a new Orchestra project.

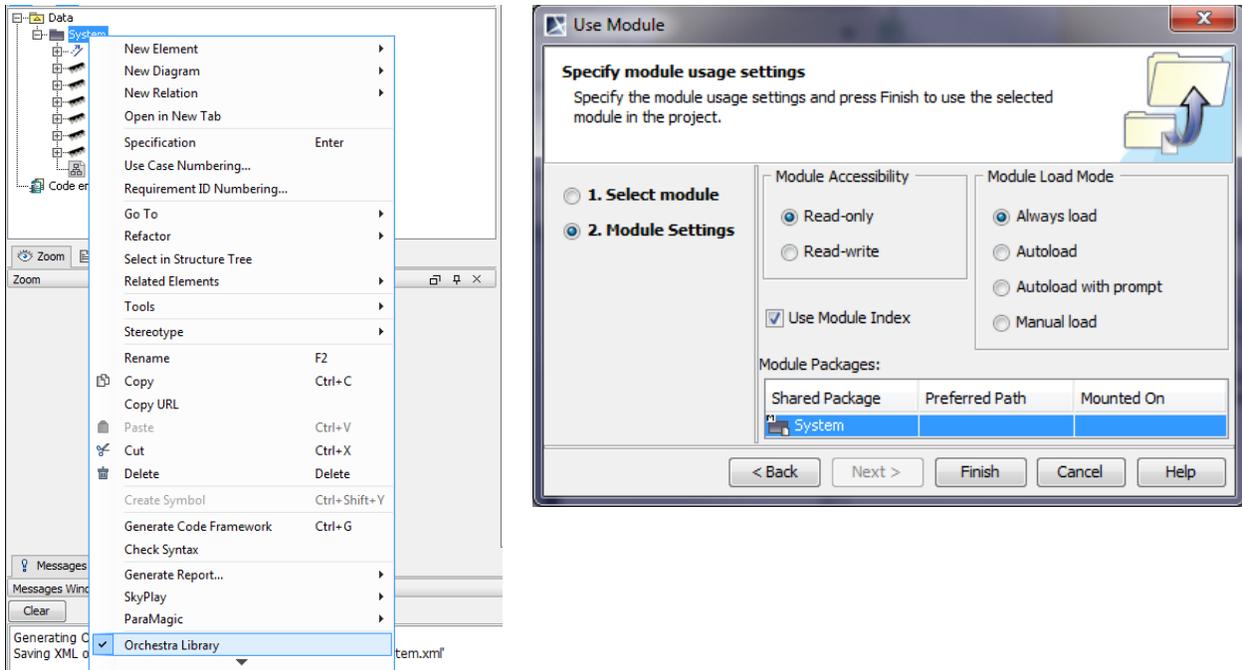


Figure 15: Sharing module package as a library (LHS) and importing a library in a new Orchestra project

When imported in a project, Orchestra library elements are visible in the MHD and SBD toolbars and ready to be used in composing complex modules (systems). For example, Figure 16 shows the OR_Gate, System, and Hypothetical_Machine_System modules available in the System library imported in the current project.

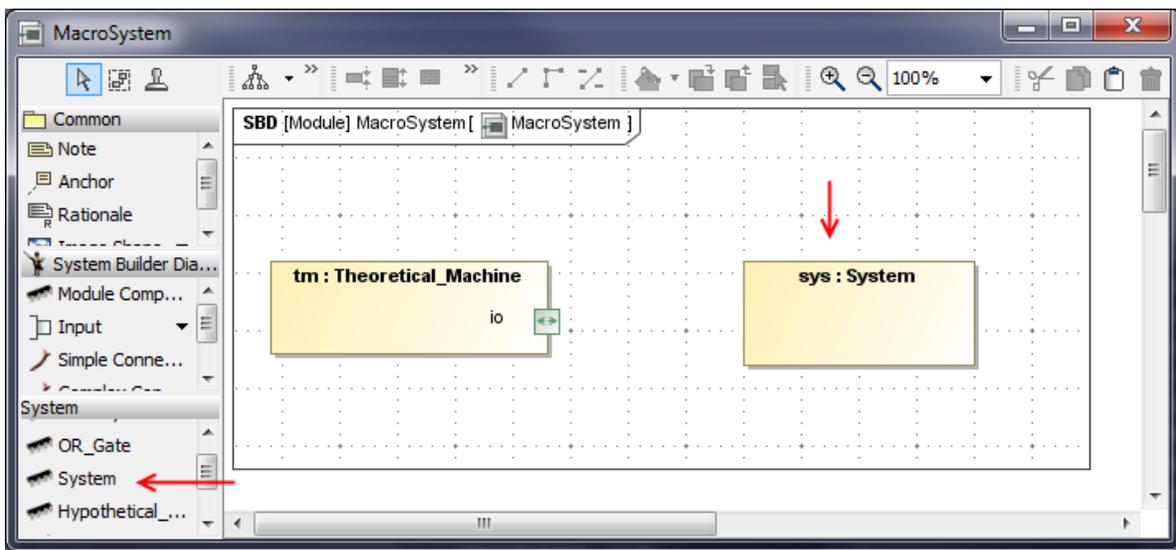


Figure 16: Orchestra modules in imported libraries can be viewed in the diagram toolbar and are ready to be composed into more complex modules (systems)

5 Summary and Next Steps

In this paper, we have presented Maestro as a MBSE environment for design and simulation of complex electronic systems using Orchestra, a simulation tool developed at Sandia National Laboratories. Maestro provides a model-based design environment, replacing the traditional document-based systems engineering process. The Orchestra DSL profile in Maestro provides the high-level Orchestra concepts, diagram definitions, and symbols for graphical modeling using these concepts. Maestro provides an easy and intuitive user interface that allows system designers and Orchestra analysts to work in their own perspectives and develop system design models (based on SysML) and Orchestra analysis models (based on Orchestra DSL). With Maestro, Orchestra DSL-based models can be automatically generated from system design models, and XML and Java-based representations of Orchestra analysis models can be generated from the Orchestra DSL-based models.

So far our technical effort with Maestro has primarily focused on structural aspects of the system, for both design and analysis models. We are actively working to include behavior and requirements modeling capabilities in Maestro. With behavior modeling, Orchestra analysts will be able to define abstract behaviors and allocate them to structural elements, such as Orchestra modules, ports, and connections. The implementation of these abstract behaviors can be done in the Java-based simulation model generated by Maestro. System designers can allocate property-based requirements to structural or behavioral elements which can then be verified based on the simulation results from Orchestra.

Broader applications of DSL approach for MBSE include

- bridging the design-manufacturing for vehicles and heavy equipment,
- development of a space sciences DSL for end-to-end mission planning, and
- modeling of communications networks for capacity and latency simulation.

References

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.

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 2: Applications of SLIM*. 21st Annual INCOSE International Symposium, Denver, CO, June 20-23, 2011.

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>.

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*[®] - 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-parasolver/
www.intercax.com/sysml.

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

InterCAX - Solvea (2011). "*Solvea*[™] - SysML Parametric Solver and Integrator for MagicDraw." Retrieved Mar 28, 2011, from 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/>.

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

Object Management Group (2010). "*Systems Modeling Language*." Retrieved Oct 24, 2010, from <http://www.omg.sysml.org>

Biography

Manas Bajaj, PhD is the Chief Systems Officer at InterCAX (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.

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.

Dirk Zwemer, PhD is President and CEO of InterCAX (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) and Associate Director of the Model-Based Systems Engineering Center (MBSE-C) (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[†] 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.

Douglas Deming, Gregory Wickstrom, and Mark De Spain (retired) are system engineers at Sandia National Laboratories, Albuquerque NM.

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