27th Annual INCOSE International Symposium (IS 2017)
Jul 15-20, 2017, Adelaide, Australia

Graph-Based Digital Blueprint for Model Based Engineering of Complex Systems

Manas Bajaj[1], PhD
Chief Systems Officer
Intercax

Jonathan Backhaus
Staff Systems Engineer
Rotary & Mission Systems
Lockheed Martin Corporation

Tim Walden, LM Fellow
Chief Engineer
Advanced Systems
Lockheed Martin Corporation

Manoj Waikar
Senior Software Arch.
Intercax

Dirk Zwemer
President, Intercax

Chris Schreiber
Space Systems Company
Lockheed Martin Corporation

Ghassan Issa
Systems Engineering Asc.
Space Systems Company
Lockheed Martin Corporation

Intercax
47 Perimeter Center East, Suite 410,
Atlanta, GA 30346, USA
www.intercax.com

Lockheed Martin
Corporate Engineering,
Technology & Operations, USA
www.lockheedmartin.com

Abstract

Complex, cyber-physical systems must be founded on a digital blueprint that provides the most accurate representation of the system by federating information from engineering models across multiple enterprise repositories. This blueprint would serve as the digital surrogate of the system and evolve as the actual system matures across its lifecycle, from conception and design to production and operations. This paper presents a graph-based approach for realizing the digital blueprint, which we refer to as the Total System Model. The paper is divided into five parts. Part 1 provides an introduction to use cases for model-based systems engineering. Part 2 introduces graph concepts for the Total System Model. Part 3 provides a demonstration of the graph-based approach using Syndeia software as a representative application. Part 4 provides a summary of this paper, and Part 5 lays out potential directions for future work.

Part 1: Model Based Systems Engineering (MBSE)

The complexity in modern systems is growing at an unprecedented pace. This complexity arises from a range of factors such as increasing number and types of system components, increasing types of interfaces and interactions between system components, availability of system functionality outside the system boundaries, and increasing interactions with the operating environment (system-of-

[1] Corresponding author: Dr. Manas Bajaj (manas.bajaj@intercax.com)

systems, internet-of-things). Most mechanical and electrical hardware-dominated systems have increasingly become cyber-physical in nature. This complexity challenges the organizations developing modern systems that must now be equipped with advanced software and hardware tools to support the system through its lifecycle—design, manufacturing, operations, and maintenance—and must also respond to rapidly evolving market forces and competition for new and improved systems.

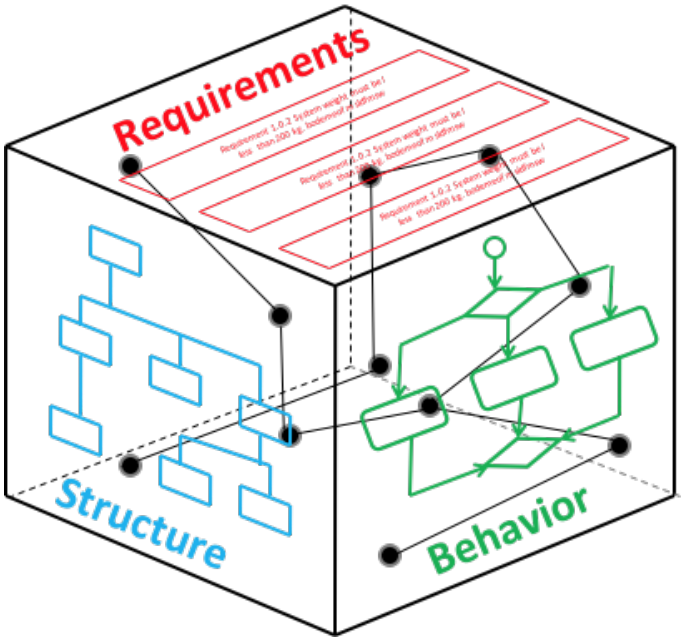

Figure 1: Goal of MBSE is to develop and maintain a single unified system architecture

The goal of Model-Based Systems Engineering (MBSE) is to create and manage a single unified model of the system that can represent all of its varied aspects, such as requirements, structure, and behavior, as shown in Figure 1. The system model is conceptualized as a graph of information that can be viewed from or projected onto different perspectives. The transition from document-based systems engineering (DBSE) to MBSE is similar to the transition from 2D mechanical drawings to 3D CAD and the ability to automatically derive 2D views from a 3D CAD model.

The OMG Systems Modeling Language (OMG SysML) has emerged as an open, international, and industry standard for representing the system architecture. However, most of the detailed engineering is performed in a variety of modeling and simulation tools, as shown in Figure 2.

The digital blueprint of a system, referred to as the **Total System Model (TSM)** hereafter, can be conceptualized as the system architecture model (as a SysML model) connected to a variety of domain-specific models via fine-grained digital connections, as shown in Figure 2. Enterprises developing complex systems use a variety of configuration management systems, such as product lifecycle management (PLM) systems for versioning and managing the engineering/manufacturing bills-of-materials, and application lifecycle management (ALM) systems for versioning and managing software code, builds, and related artifacts. The TSM includes models managed in different repositories, and inter-model connections between elements of these versioned models. This approach provides organizations the flexibility to use the best-in-class tools for each discipline and connect them to other models in the federation. The conception, development, and management of the TSM is a fundamental shift underway in organizations developing complex systems.

The TSM federation evolves over time as each of its constituent models evolve in a multi-disciplinary and collaborative environment. As shown in Figure 3, snapshots of the TSM can be taken at different stages in the system lifecycle and shared with stakeholders or archived for future queries. For any complex system, the Total System Model provides a backbone for traceability, system analyses, impact assessment, and “what-if” trade studies across the disciplines.

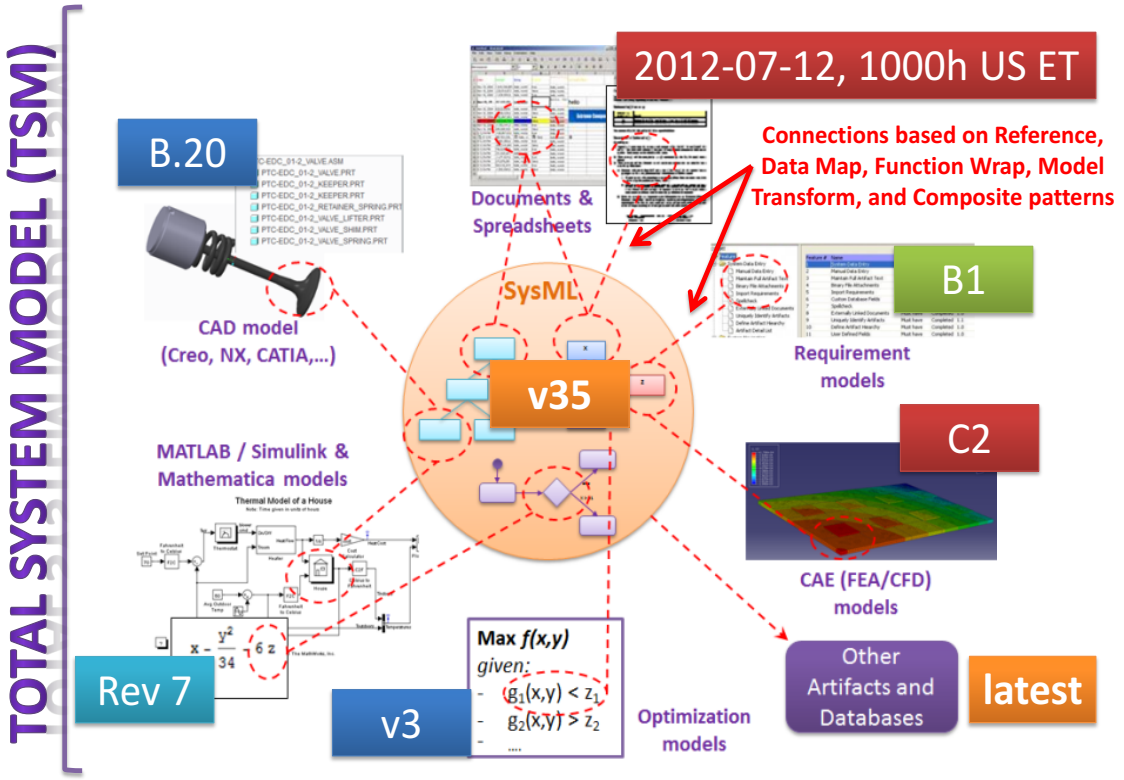

Figure 2: Total System Model as the digital blueprint of a system (snapshot in time)

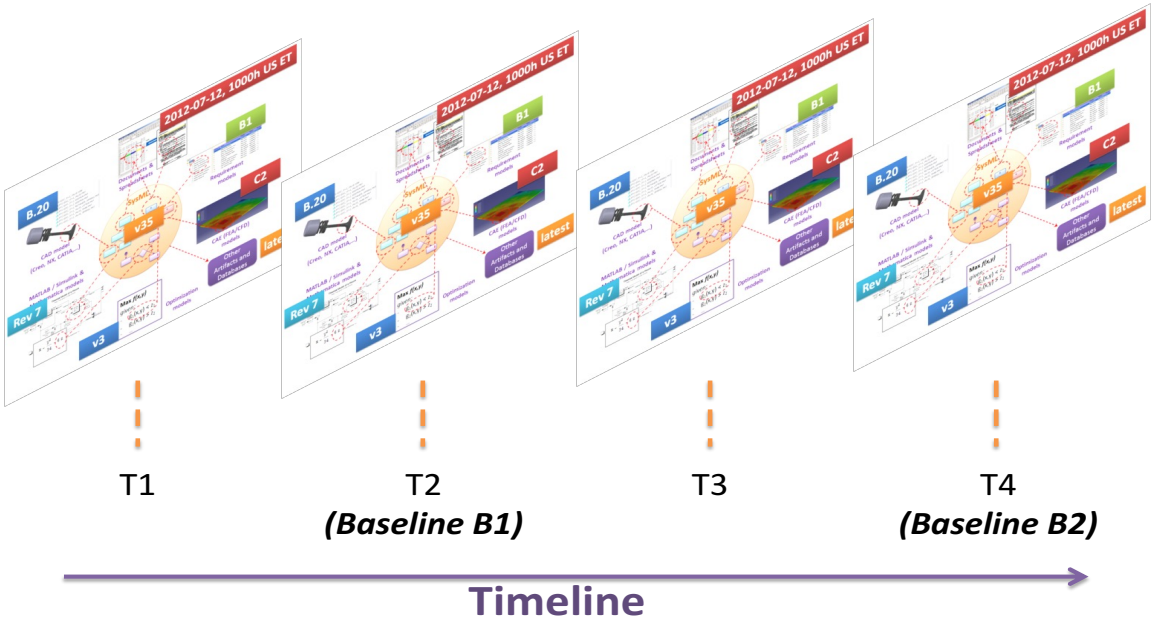

Figure 3: Total System Model evolves as the system definition matures across its lifecycle

The Total System Model may have different topologies. The topology shown in Figure 2 is a single-hub-and-spoke system with the system architecture model in SysML as the hub coordinating and connected with other disciplines. This may not be the only topology. For example, there are inter-model relationships between the models in various disciplines that are too detailed to be tracked at

the system architecture level. Consider the case of model-based connections between a CAD model and a FEA (finite element analysis) or CFD (computational fluid dynamics) model, or consider the case of connecting software requirements to code (managed in Git or Subversion) and related tickets in an issue tracking system (such as JIRA). Another alternate topology is a multi-hub-and-spoke system where we may have different hubs coordinating disciplines at different or same phases of a system's lifecycle. For example, a SysML model may be a hub for systems engineering related activities, connected with a PLM hub for coordinating mechanical/electrical design and manufacturing activities and an ALM hub for coordinating software activities. A topology without hub(s) is also a possibility but without an architecture model in place, disciplines start working as silos with ad-hoc information flows between their models.

Part 2: Graph-based Approach for the Total System Model

In this section, we discuss how graphs provide a common mathematical formalism for integrating different types of engineering models in the Total System Model federation.

Information models are graphs – Fundamentally, information models of a product or a system are composed of entities and relationships between those entities. This network of entities and relationships is a mathematical *graph* data structure. The entities are the vertices in the graph and the relationships are the edges in the graph. For example, a SysML model of a system is a graph where requirements, blocks, activities, and other elements are the vertices in the graph, and associations, dependencies, and other relationships are the edges between the vertices in the graph. Similarly, the bill-of-materials information in a PLM system is a graph. Parts have part versions, and each part version uses other part versions. A simulation model, such as a Simulink or Modelica model, is also fundamentally a graph. Graphs provide a common formalism for representing information models.

Total System Model (TSM) is a graph of graphs – The TSM is a graph of models, where each participating model is itself a graph. The vertices in the TSM graph are elements in the participating domain models, such as requirements (or requirement revisions) in a requirements database, parts (or part versions) in a PLM system, or blocks in a SysML model. The edges in the TSM graph are relationships between elements across models or within a model.

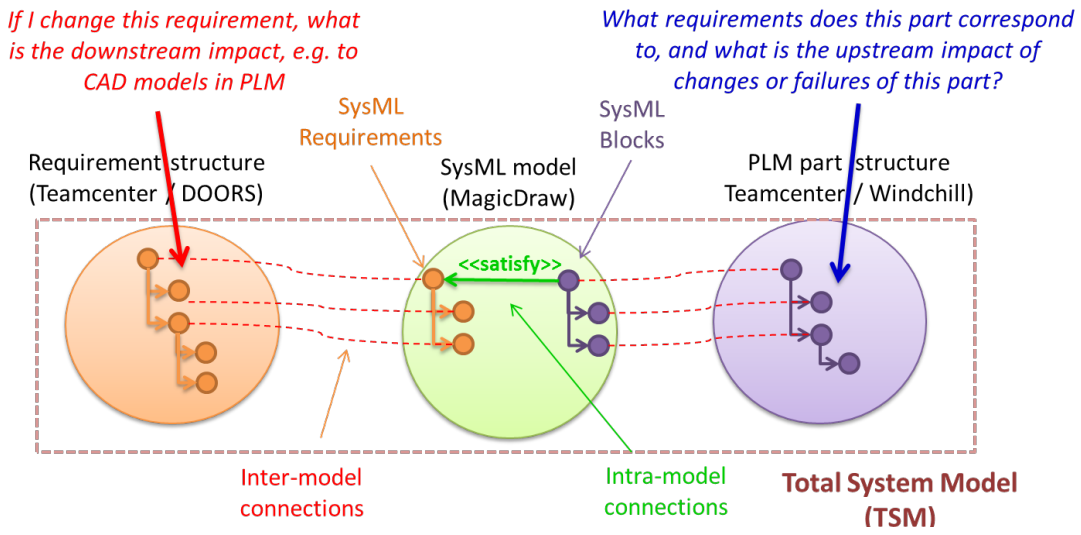

Figure 4: Both inter-model and intra-model connections (edges) exist in the TSM

Intra-Model and Inter-Model Connections (Edges) – The TSM graph includes two main types of edges, as shown in Figure 4. The **Inter-Model Connections** are the edges between model elements

in different models or repositories, such as a connection between a requirement in DOORS-NG to a requirement or block in the SysML model. These are illustrated in Figure 4 using dashed red lines between the circles representing models or repositories. The **Intra-Model Connections** are the edges/relationships between elements in the same model, such as a *satisfy* relationship from a SysML block to a SysML requirement, or a *trace link* relationship between a Teamcenter part revision to a requirement revision, or a *link* relationship between two issues or tasks in JIRA.

Since the TSM is a federation, it only owns the inter-model relationships between the participating models in the federation. The intra-model relationships are owned by the participating models but are accessible on demand to the TSM, such as when traversing the TSM graph for impact analyses.

Patterns and Services on Edges – The inter-model and intra-model connections (edges) in the TSM graph are of different types. For inter-model connections, we have identified four fundamental types as below based on the services they can provide.

Connection pattern	**Purpose**
Reference Connection	This type of connection can be used to track the versions of the related model elements.
Data Map Connection	This type of connection can be used to map and track the values of attributes of model elements, in addition to the versions of model elements.
Function Wrap Connection	This type of connection can be used to wrap executable models (e.g. Java, Python, Simulink, or Modelica), send inputs, execute the model, and get outputs.
Model Transform Connection	This type of connection can be used to map and track the structures of model elements, in addition to attributes and versions of the connected model elements.

As the TSM graph for a system is woven using different types of edges (as above), one can perform model comparisons and synchronizations across the complete spectrum of model-based connections, thereby ensuring that changes in one model can be propagated to other models.

Collaborative and automated development of the TSM Graph – As the system develops, one must consider how the TSM graph is built. Though we can connect model elements across multiple disciplines manually, it is neither a scalable nor a robust approach. The approach we have taken is that the specific parts (sub-graphs) of the TSM graph can be automatically generated as we move information across the disciplines. For example, when SysML block structure (architecture) is used to generate a simulation model, such as in Simulink or Modelica, then the inter-model connections are automatically created between the SysML blocks and the newly generated simulation model elements. The inter-model connections then provide conduits for model comparisons, data flows, transformations, comparisons, and synchronization on an ongoing basis. So, the TSM graph is woven as part of the engineering workflow. Additionally, we envision the use of machine learning techniques to detect patterns in connected data or inference data that should be connected in the TSM graph.

Models Transformations as Graph Transformations – Since we conceptualize information models as graphs, model transformations such as transforming a SysML block to a simulation model or PLM part structure, or vice versa, can be grounded in graph transformations.

Queries as Graph Pattern Matching – One of the key goals of building a Total System Model graph is to be able to perform queries to search for model elements, relationships, and patterns.

Conceptualizing the connected set of models as a graph makes it possible to leverage graph pattern matching and graph engines as a fundamental capability to formulate and execute queries. Examples include searching for any requirement in a requirements management system that has not been allocated to a structural or a functional element in the system architecture model (in SysML), or searching for all allocation relationships between hardware and software systems. A key advantage of graph pattern matching over writing functions/methods for specific queries is that it is a declarative approach—define what you are searching versus describing how to search. For example, with graph pattern matching one needs to only declare the pattern in terms of the nodes and relationships and then a graph engine could execute and search the TSM graph for all instances of the pattern.

Impact Analyses as Graph Traversals – Beyond queries, graphs provide a solid foundation for impact analyses across the engineering model space. Two examples are shown in Figure 4. Given a change in a requirement in a requirements management system, the TSM graph with intra-model and inter-model connections can be used to assess the impact of the change across the model space, including if the requirement change will affect hardware parts or CAD designs in a PLM system. Alternatively, the same graph can be used by a mechanical engineer working on the hardware design to trace the upstream requirements or system functions defined in the architecture model that may get affected if the given part fails during operation.

Part 3: Demonstration of the Graph-based Approach using Syndeia

Syndeia is a software platform for integrated model-based engineering (MBE/MBSE) developed by Intercax with collaboration and support from leading industry organizations. In this section, we use Syndeia as a representative software application to demonstrate the concepts related to TSM, especially the graph-based aspects which are the theme of this paper.

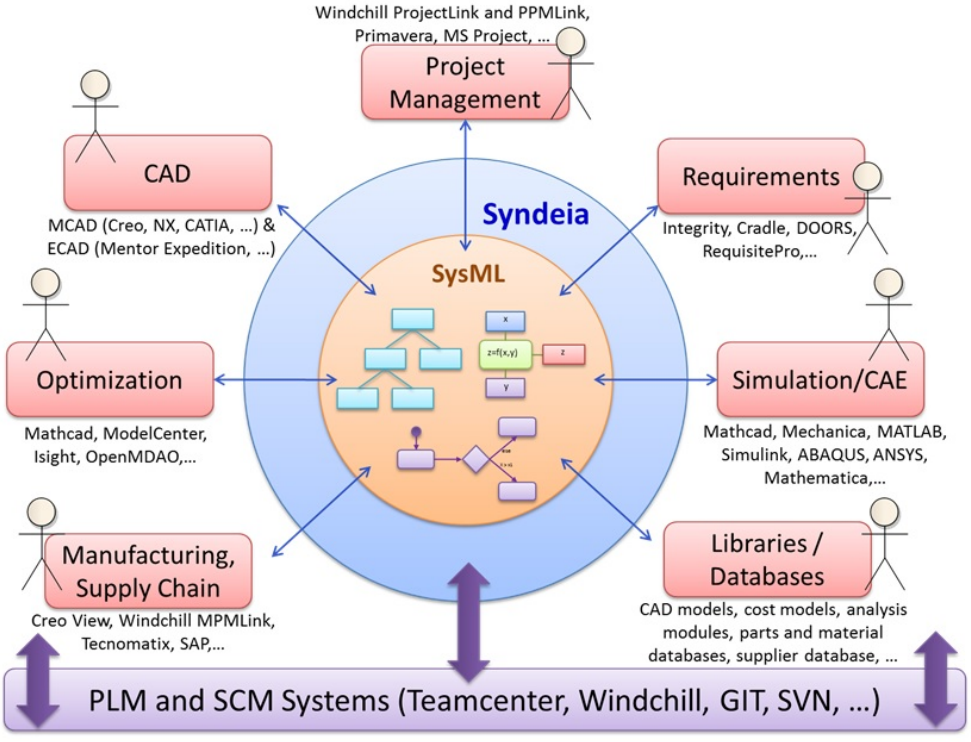

Figure 5: Syndeia is a software platform for creating, managing, querying, analyzing, and visualizing the connected graph of models for a complex system.

Syndeia enables engineering teams to collaboratively develop and manage the Total System Model (TSM) graph of a complex system (or project) by combining the system architecture model (in SysML) with models in a variety of enterprise repositories and tools, such as PLM (e.g. Teamcenter, Windchill), CAD (e.g. NX, Creo), ALM (e.g. GitHub), Project Management (e.g. JIRA), Requirements Management (e.g. DOORS-NG), Simulations (e.g. Mathematica and MATLAB/Simulink), Databases (e.g. MySQL), and other data sources (e.g. Excel), as illustrated in Figure 5. Syndeia leverages open standards, such as REST/HTTP, JSON, STEP, JDBC, OSLC, as well as native APIs to connect to, query, update and search models in enterprise repositories.

Syndeia serves as a "CM-of-CM systems" (where CM implies configuration management) by managing the configuration of the entire federation of models, while the models individually are managed in different CM systems, such as PLM, ALM, ERP (Enterprise Resource Planning systems), and databases.

Part 3.1: Generating the TSM graph

As show in Figure 6, Syndeia provides a Dashboard interface to view all the repositories and modeling environments on the RHS, and the ability to drag and drop elements from SysML architecture model to repositories, or vice versa. Depending upon the selected connection type, the drag and drop operations can create connections (e.g. for Reference connections) or generate/transform models in addition to creating connections. This makes it possible to build the TSM graph in an automated manner as part of the engineering workflow.

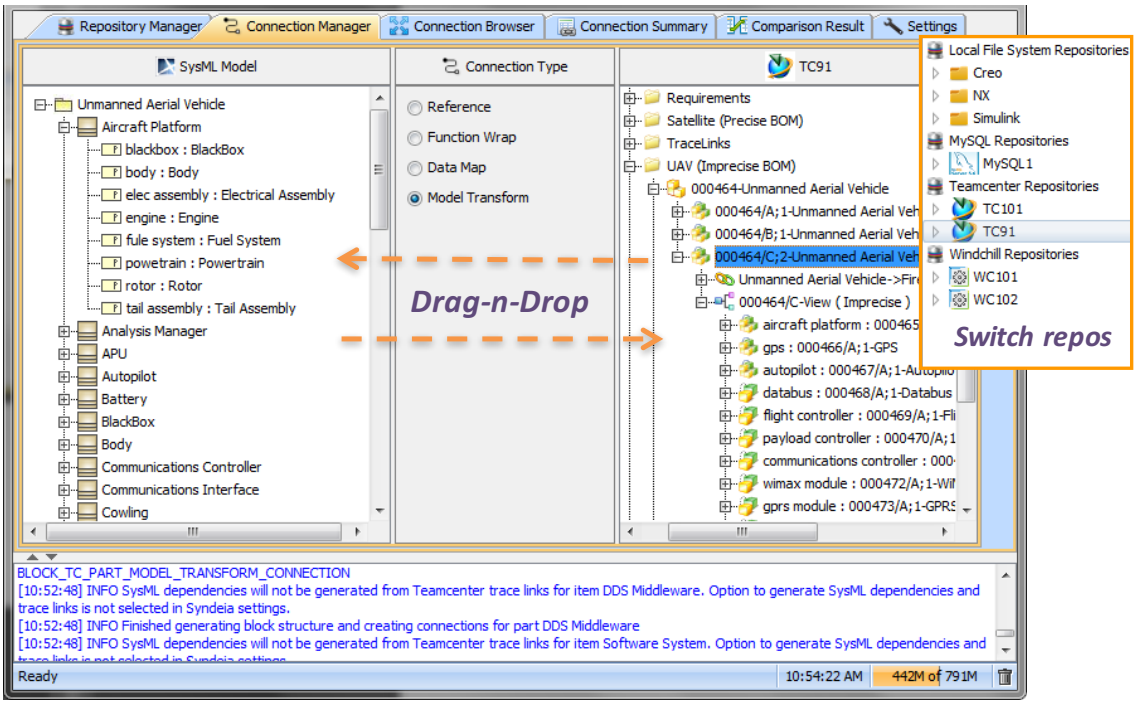

Figure 6: Simple drag and drop interface to connect, map, transform models and generate the inter-model connections (edges) of the TSM graph

Part 3.2: Visualizing the TSM Graph

The inter-model connections (edges) can be shown in a tabular format (Figure 7). The example shows model transform connections between blocks in a SysML model and the connected part versions in Teamcenter PLM system.

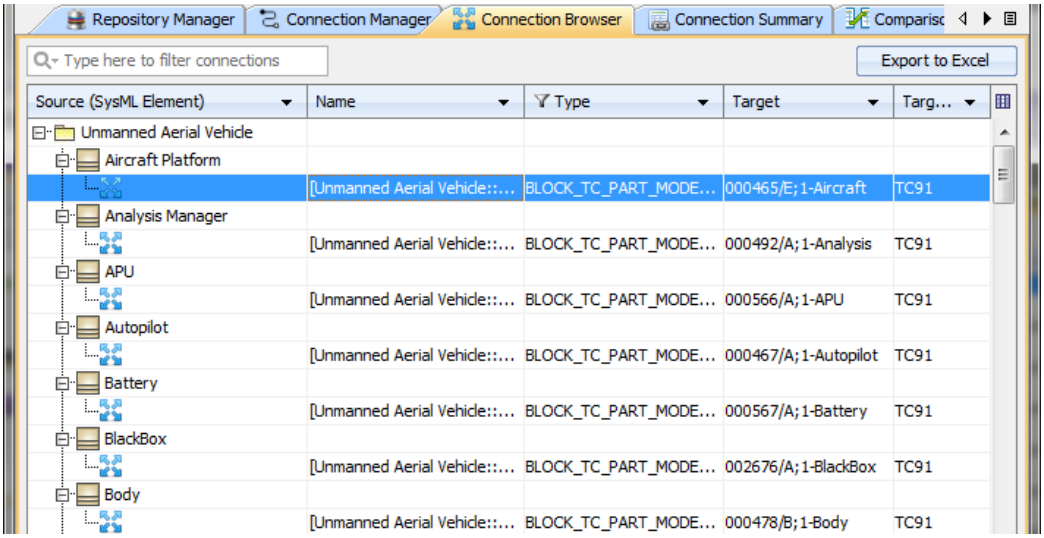

Figure 7: Inter-model connections in the TSM graph shown as a table.

Inter-model connections in the TSM graph can also be visualized interactively using a circle chord diagram, as shown in Figure 8. The figure shows elements from models in SysML, Teamcenter PLM, GitHub, JIRA, and Simulink participating in the TSM graph for an Unmanned Aerial Vehicle system.

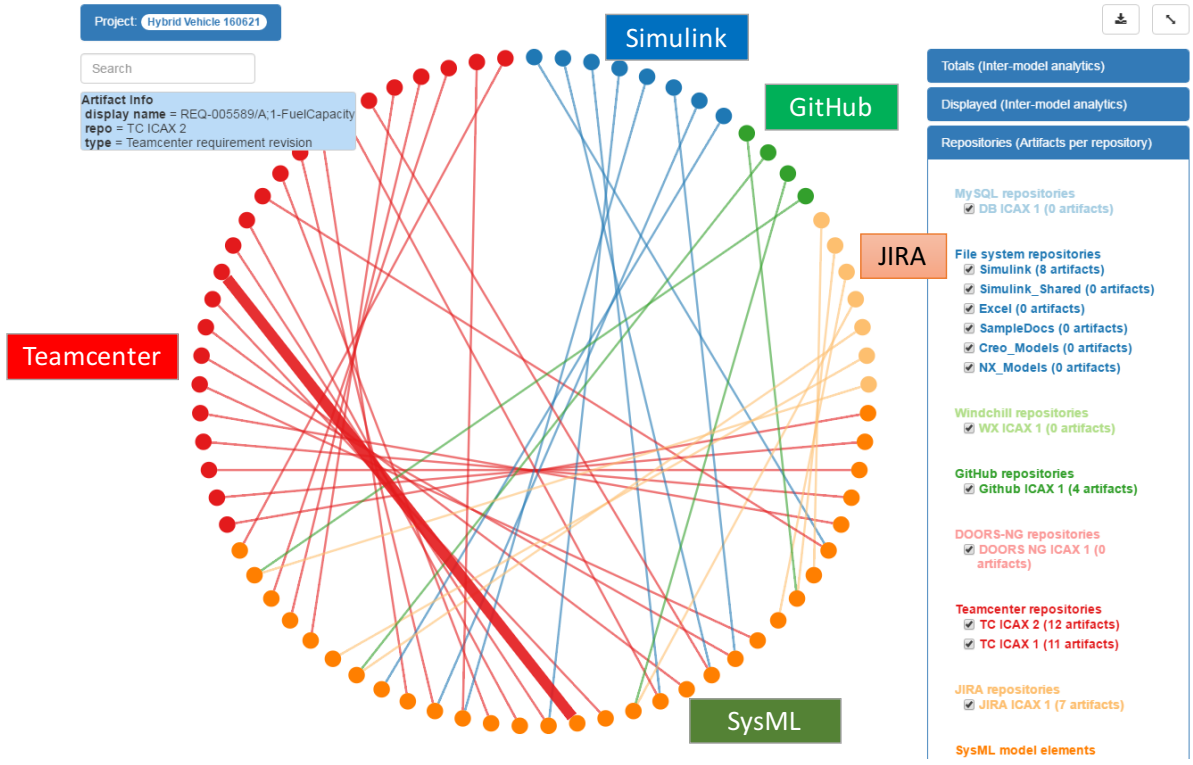

Figure 8: Inter-model connections in the TSM graph shown as a circle chord diagram

Figure 9 shows an alternative visualization of the TSM graph. Given a specific model element, such as a SysML block or Teamcenter part version or DOORS-NG requirement, one can explore all the connected model elements, one degree of separation at a time. The colored edges in the graph are the *intra*-model connections. In this example, all colored edges are in orange color and represent relationships in the SysML model, such as dependencies, associations, and containments. The black-

colored relationships are the *inter*-model relationships—between model elements in different models and tools.

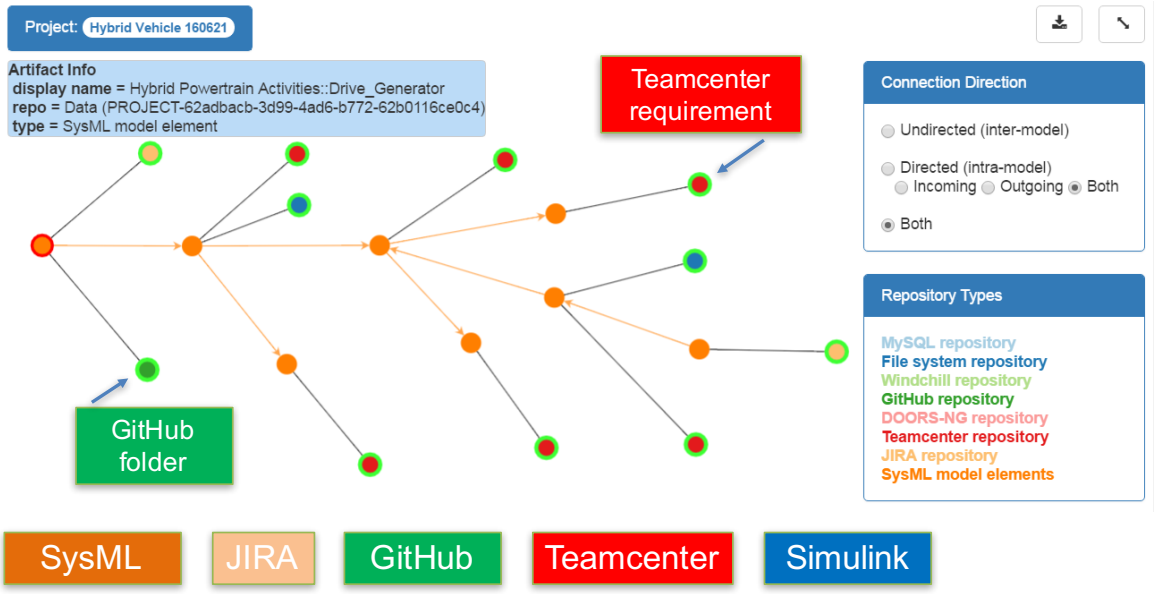

Figure 9: Exploring intra- and inter-model connections in the TSM graph

Part 3.3: Querying the TSM Graph

In this section, we will illustrate examples of graph-based queries on the Total System Model. We present these queries in 2 parts. In Part 3.3.1, we present graph queries on the system architecture model in SysML, and in Part 3.3.2, we present graph queries on the Total System Model graph which includes the SysML model and the inter-model connections to elements in requirements, PLM, ALM, databases, and other repositories.

Part 3.3.1: Graph queries on the system architecture (SysML-based) part of the Total System Model graph

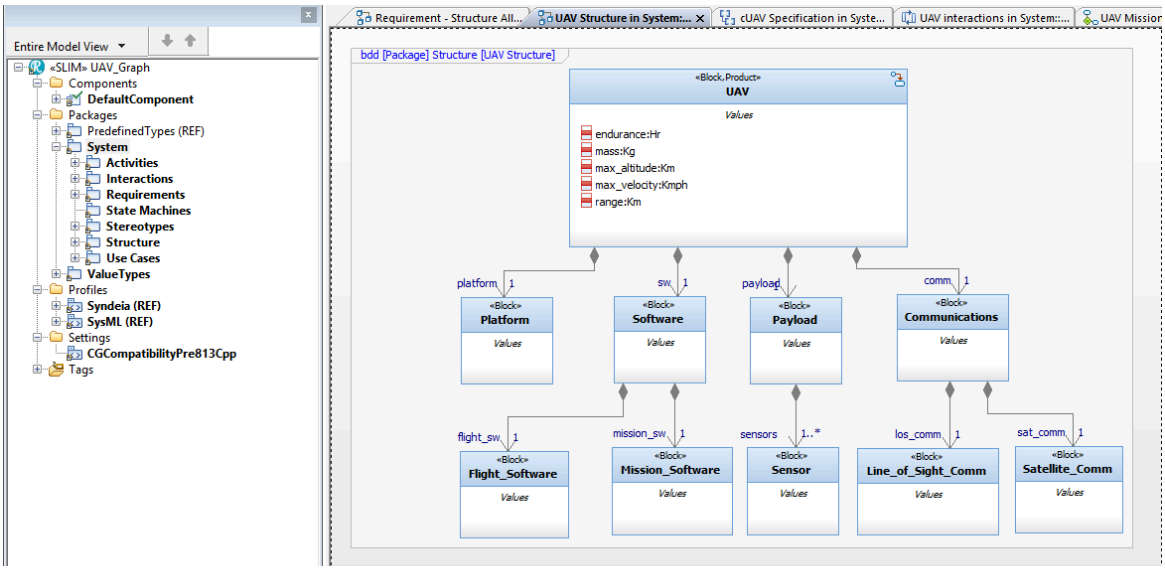

Figure 10: SysML model of a UAV with structural, functional, and requirement elements

Figure 10 illustrates an example architecture model of an Unmanned Aerial Vehicle (UAV) system in SysML (IBM Rational Rhapsody). The model contains requirements, block structure, use cases, and interactions, state machine, and activity-based definitions of UAV behaviors. Figure 11 illustrates a graph structure generated in Neo4j graph database from the SysML-based architecture model using Syndeia.

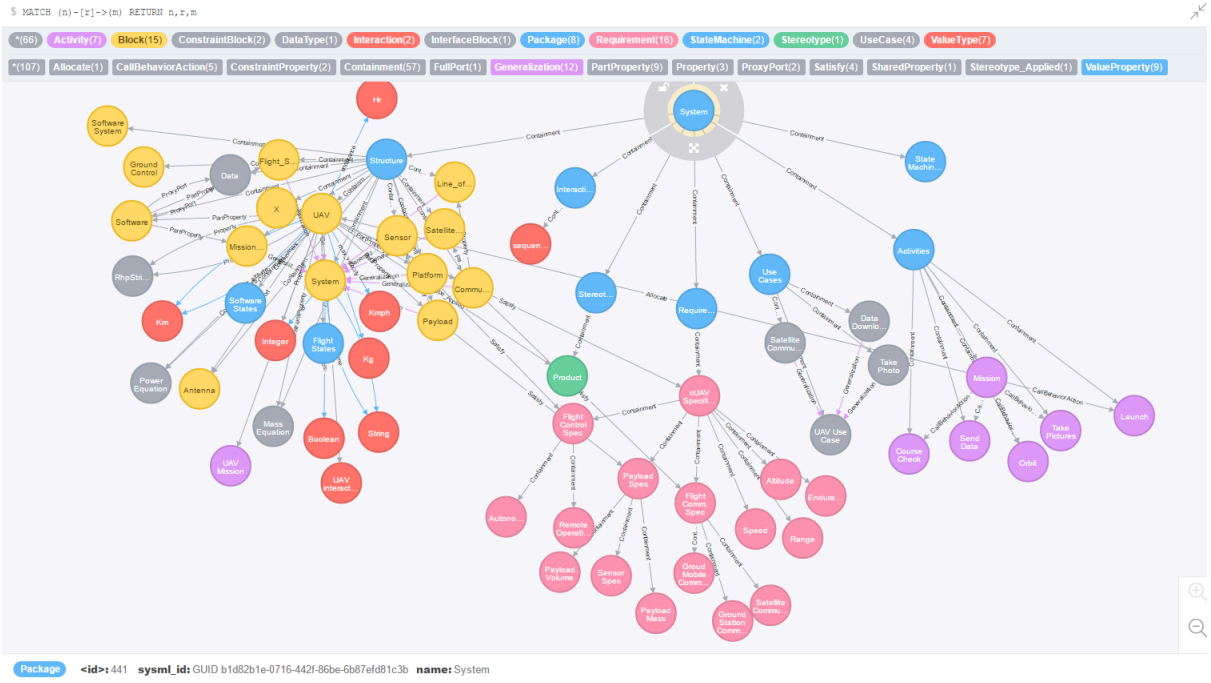

Figure 11: UAV graph in Neo4j graph database generated from the SysML model

Given a graph structure, we can now run queries on the architecture model using graph patterns. Some example queries and resulting sub-graphs are illustrated below. The queries are formulated using the Cypher query language for Neo4j and are shown in the top part of the figures. For example, the Cypher query **match(p:Package) return p** fetches the SysML package structure.

1. Query – Get all packages in the SysML-based architecture model of the UAV

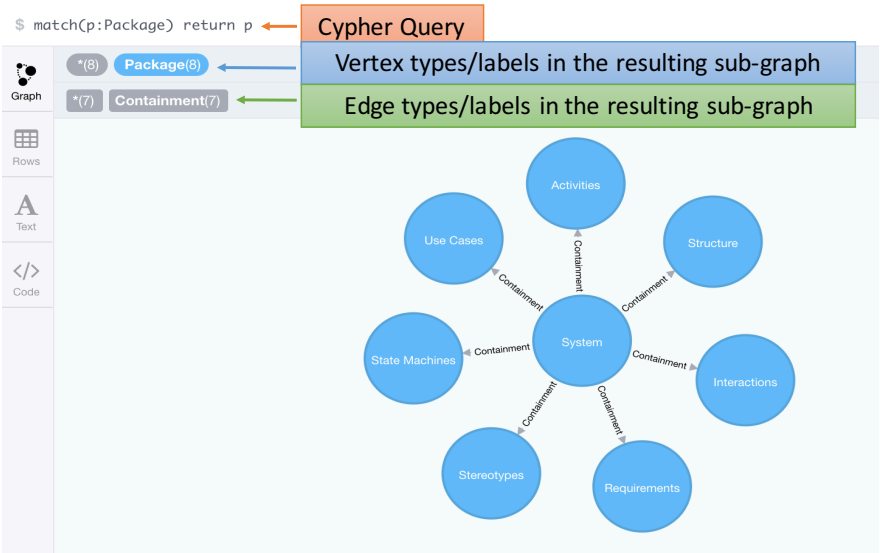

Figure 12: Query to get all packages in the model

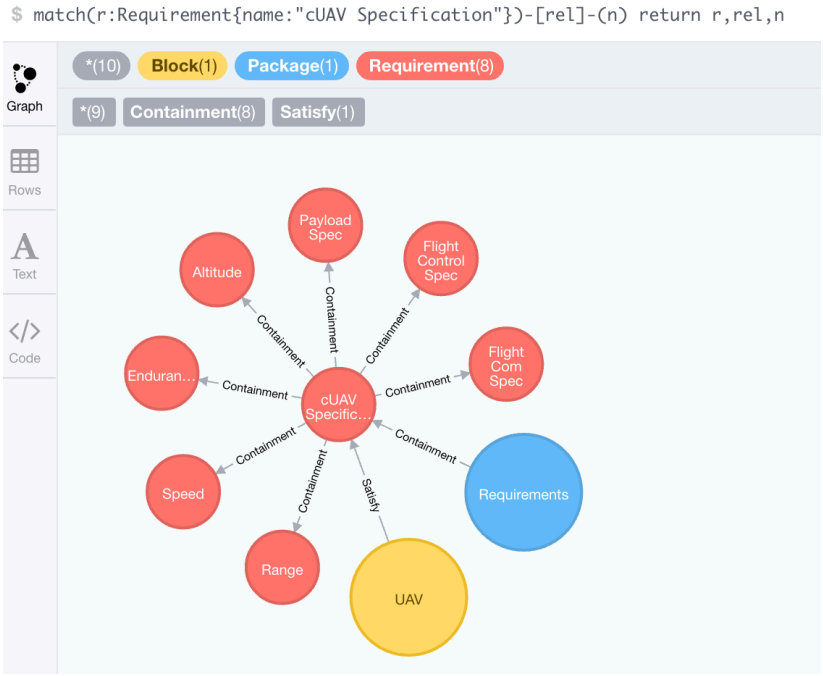

Figure 13: Query to get all relationships for UAV requirement specification

2. Query – Get all incoming and outgoing relationships (edges) for the UAV requirement specification.

Figure 14: Query to get all relationships for the UAV block

3. Query – Get all relationships for the UAV block. The results show all properties of the UAV, and requirements and behaviors allocated to the UAV.

4. Query – Get all behaviors of the UAV system

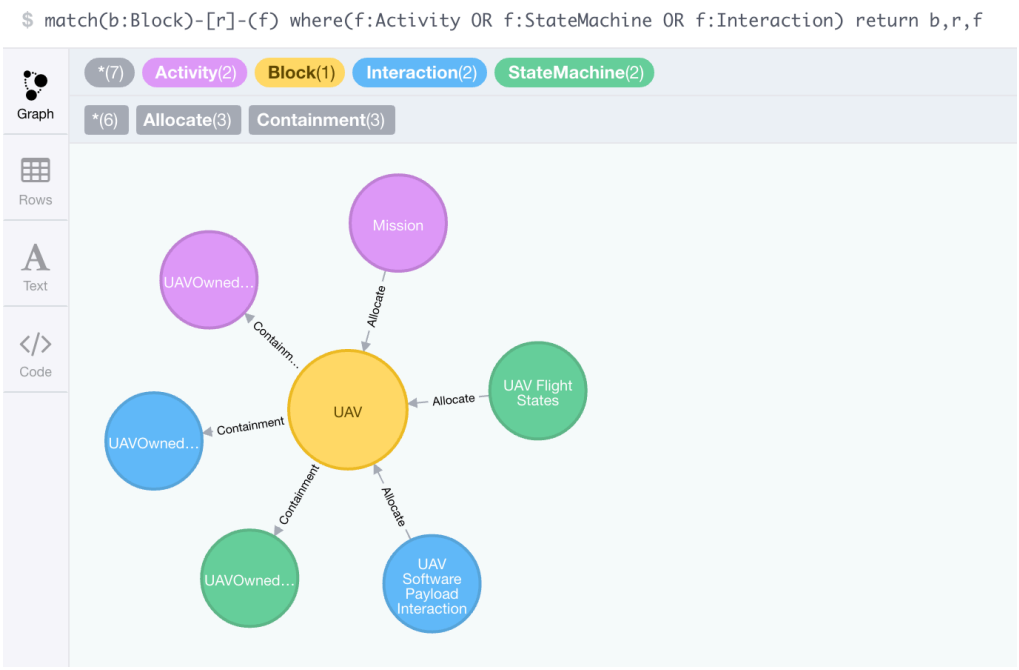

Figure 15: Query to get all behavior models associated with the UAV block

Part 3.3.2: Graph queries on the complete Total System Model graph

5. Query – Get the complete Total System Model graph, including the SysML model and inter-model connections. Note the types of vertices and edges returned.

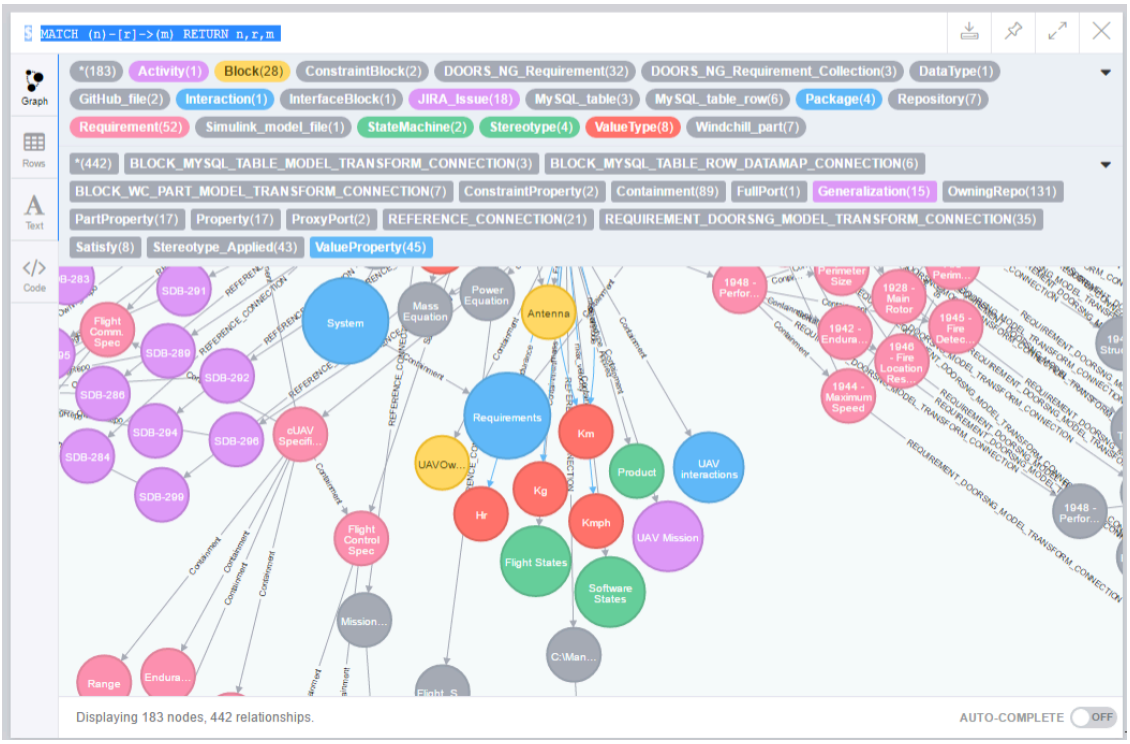

Figure 16: Get all nodes and edges in the TSM graph

6. Query – Get all connections between SysML and DOORS-NG requirements. In the resulting sub-graph, the red nodes are SysML requirement and grey nodes are requirements or requirement collections in DOORS-NG.

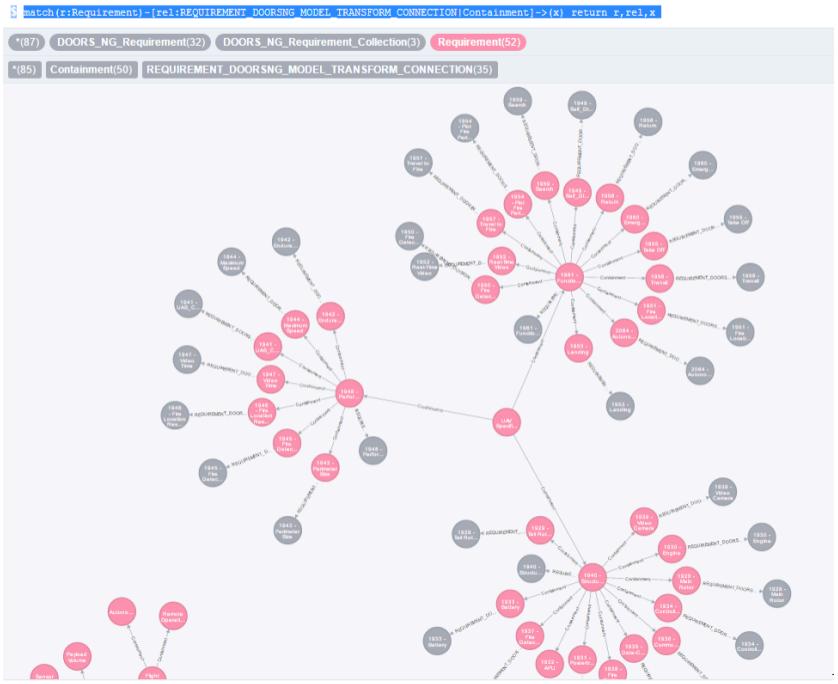

Figure 17: Get all SysML – DOORS-NG relations

7. Query – Get all inter-model connections in the TSM graph that are of type "REFERENCE_CONNECTION". The resulting graph (Figure 18) shows reference connections between SysML blocks and JIRA issues (purple vertices), SysML blocks (yellow vertices) and GitHub files (red vertices), and SysML blocks and Simulink models (green vertices).

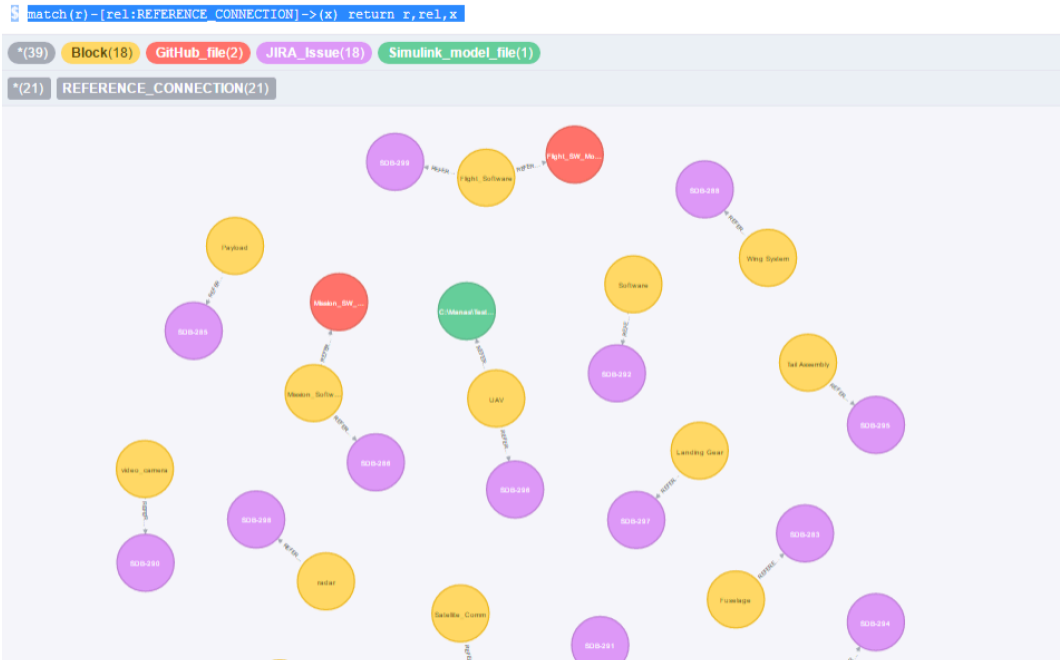

Figure 18: Get all reference connections in the TSM graph

8. Query – Get all connections to MySQL tables and rows. The resulting graph shows SysML blocks (yellow vertices)—radar, video_camera, thermal_camera—connected to MySQL tables with the same names (red vertices), and specialization of those blocks connected to rows in MySQL tables (green vertices).

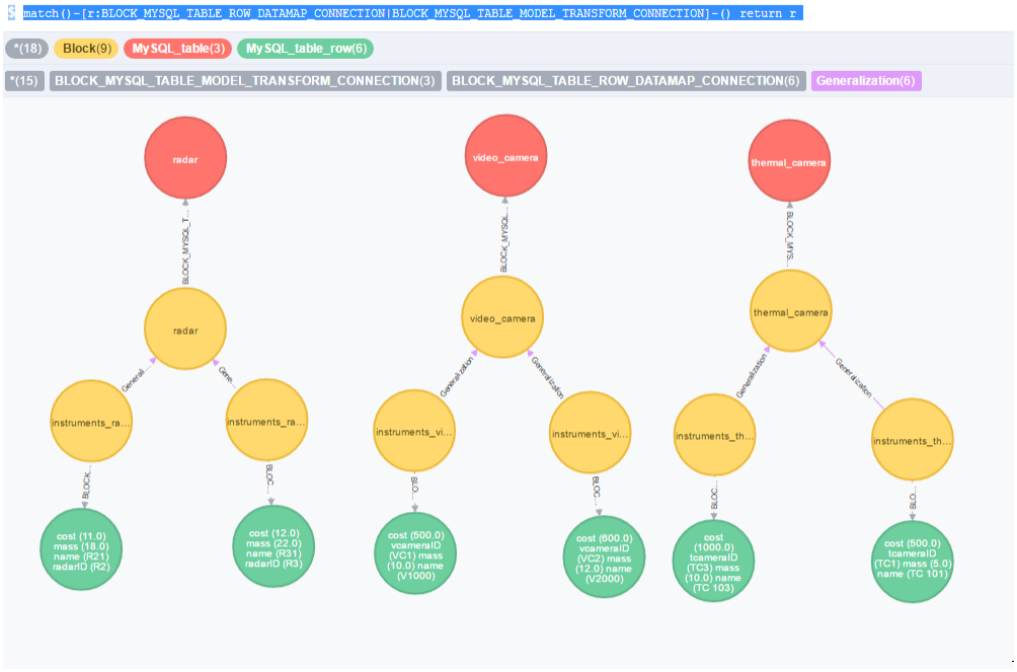

Figure 19: Get all connections to MySQL tables and rows in the TSM graph

9. Query – Trace between the UAV block in the SysML model and the Electrical System part (version A.1) in Windchill. The resulting sub-graph indicates that the UAV system uses the Platform sub-system which uses an Electrical sub-system that is connected to the Electrical System part version A.1 in Windchill.

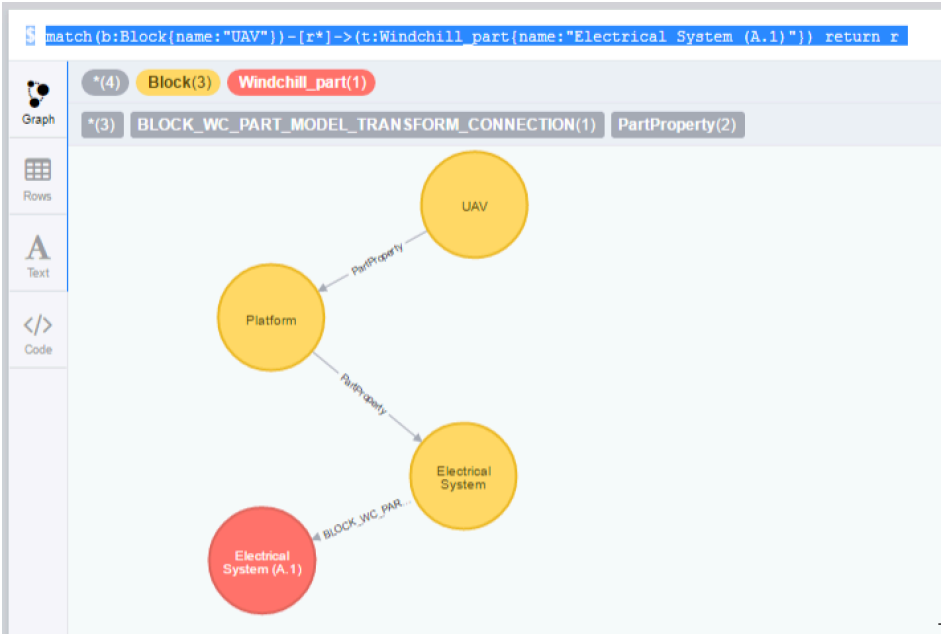

Figure 20: Trace relationships between UAV block and a Windchill part in the TSM graph

10. Query – Get all connections between SysML blocks in the architecture model and PLM (Windchill) parts.

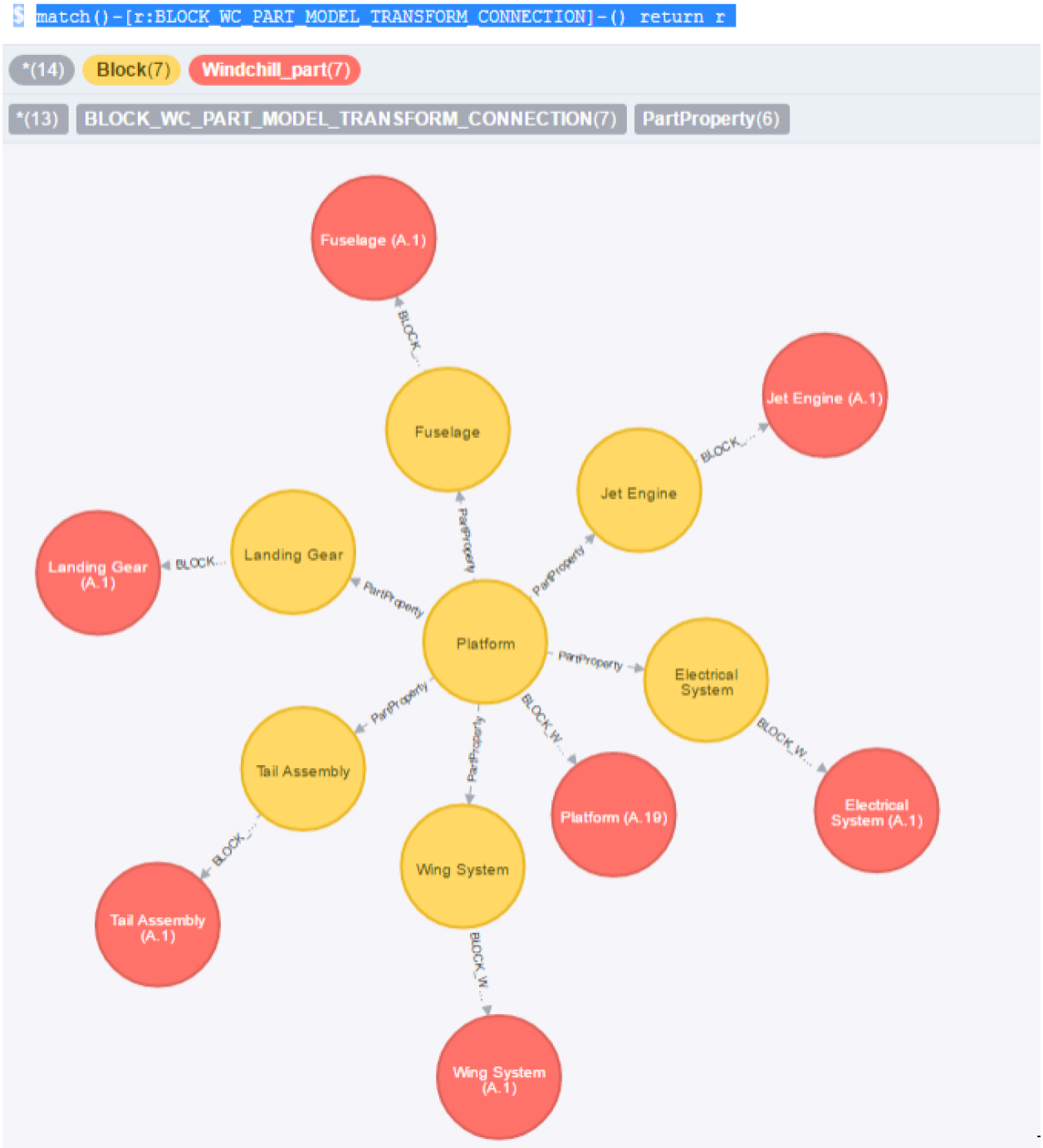

Figure 21: Get all connections between SysML blocks and PLM (Windchill) parts in the TSM graph

11. Query – Trace between a DOORS-NG requirement collection and Windchill part. The resulting sub-graph indicates that the DOORS-NG requirement collection (grey vertex) is connected to the Windchill part (red vertex) via a SysML requirement (pink vertex) and SysML block (yellow vertex). This truly demonstrates inferencing traceability across models.

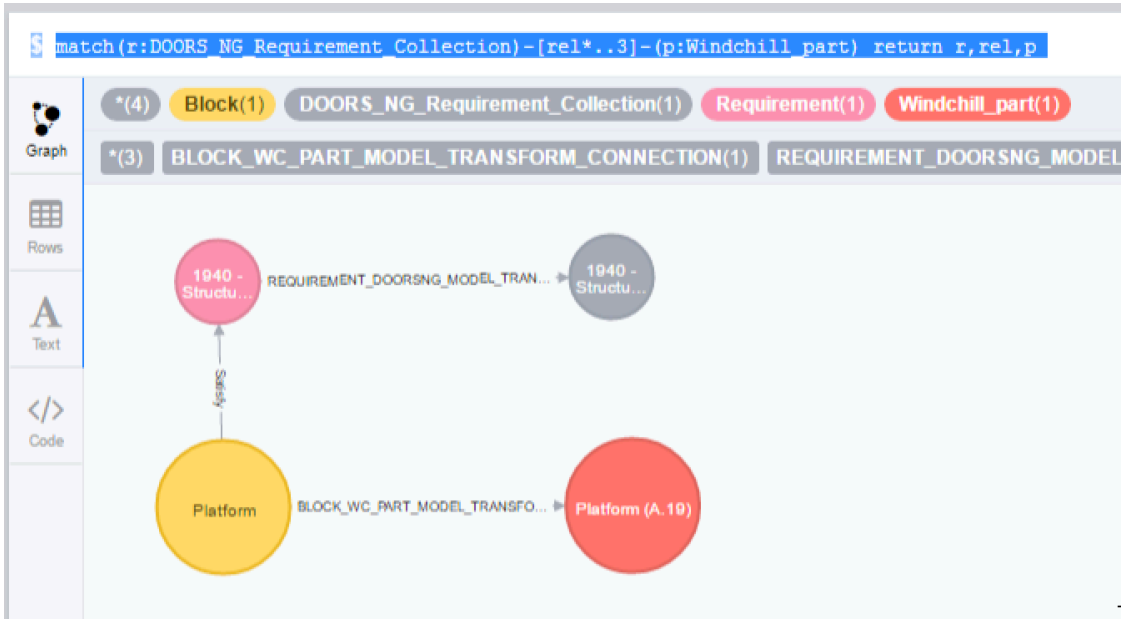

Figure 22: Get traceability between DOORS-NG requirements / requirement collections and Windchill parts within 3 degrees of separation

Part 3.4: Maintaining consistency across the TSM Graph

Syndeia leverages the inter-model connections in the TSM graph to provide services for model comparisons and bi-directional synchronization. As show in Figure 23, a difference table can be generated across a spectrum of inter-model connections to indicate elements that are out of sync. For the example shown in Figure 23, new tracking sensors have been added to the UAV system architecture model in SysML but are not a part of the UAV bill-of-material in the PLM system (Teamcenter). This is highlighted in red in the difference table.

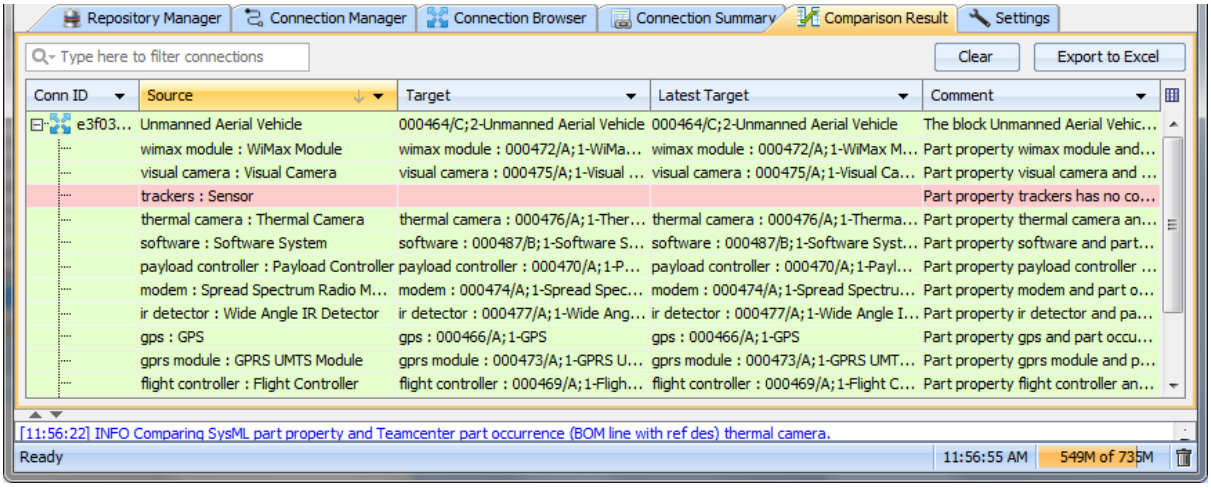

Figure 23: Compare and synchronize connected model elements in the TSM graph

Part 4: Summary

In this paper, we have introduced the concept of a *Total System Model* as the digital blueprint of a system, federating versioned models and model-elements from multiple enterprise repositories and software tools, such as PLM, CAD/CAE, ALM, requirements, database, and software configuration management systems. This approach provides the benefit to use best-in-class tools and repositories for modeling different aspects of the system, such as structure, behavior, requirements, simulation, and integrating them using a common system architecture model (as in SysML). We have abstracted the Total System Model as a graph structure where the nodes are the models (or model elements) in different tools and the edges are the relationships between the models (or model elements). We have presented concrete use cases for creating, visualizing, querying, and maintaining consistency in the Total System Model graph, using a representative software application—Syndeia. A very broad set of query examples (graph pattern matching) have been presented to illustrate the value of abstracting the Total System Model as a graph.

Part 5: Future Work

We envision the following potential directions for future work.

1. Develop a library of graph patterns to represent frequently asked questions and basic model verification rules that can be executed on the Total System Model, such as detecting all requirements that are not satisfied by a structure or function element, or don't have a test case.
2. Create mechanisms to develop verification and validation plans that can be executed on the TSM graph to continuously check for the consistency of the system model. This may include verifying the physics of the problem or continuous verification of requirements.
3. Develop an approach to model the history of the graph as it evolves across the system lifecycle and playout the evolution of a system requirement, part, or function across the lifecycle.
4. Formulate approaches to generate sub-graphs of the Total System Model that can be shared outside the boundary of the organization to enable model-based system integration without compromising information security and intellectual property.
5. Enrich relationships (edges) in the TSM graph with parametrized mathematical expressions to perform mathematical analyses, varying from simple mass roll-ups to system trade studies and assessing the quantitative impact of changes.

References

- Bajaj, M., Cole, B., Zwemer, D. "Architecture to Geometry Integrating System Models with Mechanical Design". AIAA Space 2016 Conference, Long Beach, CA, USA, Sep 13-16, 2016 - https://goo.gl/6CZIcw
- Bajaj, M., Zwemer, D., Yntema, R., Phung, A., Kumar, A., Dwivedi, A., Waikar, M. "MBSE++ Foundations for Extended Model-Based Systems Engineering Across System Lifecycle". 26th Annual INCOSE International Symposium (IS 2016) Edinburgh, Scotland, UK, July 18-21, 2016 - https://goo.gl/qpFaOT
- 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 - http://goo.gl/ga5kG
- 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 - http://goo.gl/C1awN
- CATIA (Dassault Systèmes) - http://www.3ds.com/products-services/catia, as accessed on Mar 30, 2016
- Creo (PTC) - http://www.ptc.com/cad/creo, as accessed on Mar 30, 2016

- Dropbox - https://www.dropbox.com/, as accessed on Mar 30, 2016
- Egnyte - https://www.egnyte.com, as accessed on Mar 30, 2016
- Enterprise Architect (Sparx Systems) - http://www.sparxsystems.com/products/ea/, as accessed on Mar 30, 2016
- Enovia (Dassault Systèmes) - http://www.3ds.com/products-services/enovia, as accessed on Mar 30, 2016
- Fisher, A., Nolan, M., Friedenthal, S., Loeffler, M., Sampson, M., Bajaj, M., VanZandt, L., Hovey, K., Palmer, J. and Hart, L. (2014), *Model Lifecycle Management for MBSE*. INCOSE International Symposium, 24: 207–229. doi: 10.1002/j.2334-5837.2014.tb03145.x (PDF available at http://goo.gl/ZWywBH)
- FMI - https://www.fmi-standard.org/, as accessed on Mar 30, 2016
- Git - https://git-scm.com, as accessed on Mar 30, 2016
- GitHub - https://github.com/, as accessed on Mar 30, 2016
- Google Drive - https://www.google.com/drive/, as accessed on Mar 30, 2016
- ISO 10303 (STEP) - http://goo.gl/qH7Rdw, as accessed on Mar 30, 2016
- JIRA (Atlassian) - https://www.atlassian.com/software/jira/, accessed on Mar 30, 2016
- JSON - http://www.json.org/, accessed on Mar 30, 2016
- MagicDraw (No Magic) - http://www.nomagic.com/products/magicdraw.html, as accessed on Mar 30, 2016
- MATLAB/Simulink (MathWorks) - http://in.mathworks.com/products/simulink/, as accessed on Mar 30, 2016
- Mathematica (Wolfram Research) - http://www.wolfram.com/mathematica/, as accessed on Mar 30, 2016
- Melody (Intercax) - http://intercax.com/products/melody/, as accessed on Mar 30, 2016
- Neo4j (Neo Technology) - https://neo4j.com/, as accessed on Nov 20, 2016
- NX (Siemens) - http://www.plm.automation.siemens.com/en_us/products/nx/, as accessed on Mar 30, 2016
- OSLC - http://open-services.net/, as accessed on Mar 30, 2016
- ParaMagic (Intercax) - http://intercax.com/products/paramagic/, as accessed on Mar 30, 2016
- ParaSolver (Intercax) - http://intercax.com/products/parasolver/, as accessed on Mar 30, 2016
- PLCS - https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=plcs, as accessed on Mar 30, 2016
- PTC Integrity Modeler (formerly Artisan Studio from Atego) - http://www.ptc.com/model-based-systems-engineering/integrity-modeler, as accessed on Mar 30, 2016
- RDF - http://www.w3.org/RDF/, as accessed on Mar 30, 2016
- REST - http://www.w3.org/2001/sw/wiki/REST, as accessed on Mar 30, 2016
- Rhapsody (IBM) - http://goo.gl/qMXwn6, as accessed on Mar 30, 2016
- ReqIF - http://www.omg.org/spec/ReqIF/, as accessed on Mar 30, 2016
- SOAP - http://www.w3.org/TR/soap/, as accessed on Mar 30, 2016
- Solvea (Intercax) - http://intercax.com/products/solvea/, as accessed on Mar 30, 2016
- Subversion (Apache) - https://subversion.apache.org, as accessed on Mar 30, 2016
- Syndeia (Intercax) - http://intercax.com/products/syndeia/, as accessed on Mar 30, 2016
- Systems Modeling Language (SysML, OMG) - http://www.omgsysml.org/, as accessed on Mar 30, 2016
- Teamcenter (Siemens PLM) - http://goo.gl/bv7iEB, as accessed on Mar 30, 2016
- UPDM - http://www.omg.org/spec/UPDM/, as accessed on Mar 30, 2016
- Windchill (PTC) - http://www.ptc.com/product-lifecycle-management/windchill, as accessed on Mar 30, 2016
- WSDL – http://www.w3.org/TR/wsdl, as accessed on Mar 30, 2016

Biography

Manas Bajaj, PhD is the Co-Founder and Chief Systems Officers at Intercax. He has led multiple government and corporate sponsored R&D projects over last 15 years, including SBIR Phase 1 & 2 awards. He has led the development of several commercial software applications, including the

Syndeia application referenced in this paper. Dr. Bajaj earned his PhD (2008) and MS (2003) in Mechanical Engineering from Georgia Tech, and BTech (2001) from Indian Institute of Technology (IIT), Kharagpur, India. He has been actively involved in the development of the OMG SysML standard and the ISO STEP standards, and is a Content Developer for the OCSMP certification program. Dr. Bajaj is the author of numerous technical papers and articles. He is a co-developer of a widely popular SysML and MBSE training program with over 4500 participants since 2008.

Dirk Zwemer, PhD is Co-Founder and President/CEO of Intercax, directing business development and providing strategic consulting for customers adopting model-based systems engineering practices. He is a certified systems modeling professional (OCSMP Level 4 —Model Builder Advanced). He has over 30 years of experience, and is the author of numerous patents, technical papers, trade journal articles, and market research reports. He received a PhD in Chemical Physics from UC Berkeley and an MBA from Santa Clara University.

Manoj Waikar is a Senior Software Architect and development lead at Intercax. He is a core contributor to the Syndeia platform for MBE/MBSE. He has a rich background in next-generation web-based applications and technologies.

Jon Backhaus is a Staff Systems Engineer at Lockheed Martin within the Rotary & Mission Systems business area, supporting the Advanced Systems organization Digital Transformation Initiative. Jon has a background in systems engineering and applied mathematics. He earned a Master's in Systems Engineering from Cornell University and a Bachelor's in Electrical Engineering from Bucknell University.

Tim Walden is a Lockheed Martin Fellow and the Advanced Systems Chief Engineer, within Corporate Engineering & Program Operations. Tim has led the corporate Digital Transformation Initiative, bringing the advancements of the 4th Industrial Revolution to the diverse Lockheed Martin portfolio. He has 30 years in the Defense Industry, including 20 years in the satellite ground system domain. He has a Bachelor of Science in Computer Science from West Virginia University.

Chris Schreiber is a Systems Engineering Manager at Lockheed Martin Space Systems Company, leading Model-Based Systems Engineering implementation across all Space Systems Company programs, and is the acting Senior Manager for the Systems Engineering Modernization department. Chris has a background in software and systems engineering in the aerospace and defense and manufacturing industries. He earned his B.S. in Finance and Economics from the University of Montana, and is completing his M.S. in Computer Science Systems Engineering from the University of Denver.

Ghassan Issa is a Systems Engineering Associate at the Lockheed Marting Space Systems Company.