A Semantic Platform Infrastructure for Requirements Traceability and System Assessment

Parastoo Delgoshaei
Department of Civil and Environmental Engineering
University of Maryland
College Park, MD 20742, USA
Email: parastoo@umd.edu

Mark A. Austin
Department of Civil and Environmental Engineering
University of Maryland
College Park, MD 20742, USA
Email: austin@isr.umd.edu

Daniel A. Veronica
Energy and Environment Division
National Institute of Standards and Technology (NIST)
Gaithersburg, MD 20899, USA
Email: daniel.veronica@nist.gov

***Abstract*—This work-in-progress paper describes a new approach to requirements traceability and system assessment through the use of semantic platforms. The platform infrastructure corresponds to an integration of traceability mechanisms with reusable domain-specific ontologies, and associated sets of mathematical/logical rules for design rule checking. Engineering system components are modeled as instances of the domain ontologies with values for their properties filled in. We expect that when the proposed infrastructure is fully developed it will enhance systems engineering practice in several ways. First, by filling the gap between system requirements and system models, semantic platforms will lower validation costs by allowing for rule checking early in design. Second, semantic platforms will support performance assessment during the system operation. Our medium-term research and development objective is semantic platform infrastructures capable of supporting the design, simulation, verification, and management of engineering systems having mixtures of discrete and continuous behavior.**

Keywords-Systems Engineering; Semantic Modeling; Design Platform; Requirements; Rule Checking.

I. INTRODUCTION

Under financial sponsorship of the National Institute for Standards and Technology in Gaithersburg, Maryland, the authors are working on the design and implementation of procedures and software for the model-based systems engineering, integration, and performance-assessment of cyberphysical systems (CPS). The distinguishing feature of CPS is a coupling of physical and cyber systems, with the cyber affecting the physical and vice versa. Present-day design procedures are inadequate for the design of modern CPS systems. Among the many challenges that CPS presents, design space exploration and trade studies are difficult to conduct because decision variables span parametric, logical, and dependency relationship types. Components are often required to serve multiple functions – as such, cause-and-effect mechanisms are no longer localized and obvious. System relationships can reach laterally across systems hierarchies and/or intertwined network structures.

II. PROJECT OBJECTIVES

With the objective of addressing the the long-term systems engineering challenges that CPSs present, this paper describes a new approach to requirements traceability and system assessment through the use of semantic platforms. As we will soon see in Sections IV and V, the proposed platform infrastructure corresponds to an integration of traceability mechanisms with reusable domain-specific ontologies, and associated sets of mathematical and logical rules for design rule checking. Engineering system components are modeled as instances of the domain ontologies with values for their properties filled in.

A key element of our long-term research objective is development of knowledge – methodologies, algorithms, software – required to design and build scalable platform infrastructures, with the latter supporting the simulation and design, verification, and management of multidisciplinary (e.g., mechanical, electrical, software) sensor-enabled engineering systems having mixtures of discrete and continuous component-level behavior. A series of progressively capable software prototypes will be built. Each iteration of development [3], [4] will employ a combination of software design patterns (e.g., networks of model, view, controllers), software libraries and languages for semantic applications development (e.g., OWL, Jena) [7], [13] and executable statecharts, finite element procedures for the computation of behaviors over continuous physical domains (e.g., fluid flow in a pipe network), and customized visualization for the requirements, ontology and engineering models.

III. RESEARCH METHODOLOGY

Model-based systems engineering development is an approach to systems-level development in which the focus and primary artifacts of development are models, as opposed to documents. Our research methodology is driven by a need to achieve high levels of productivity in system development. We believe this can be achieved through the use of high-level visual abstractions coupled with lower-level (mathematical) abstractions suitable for formal systems analysis. The high-level abstractions provide a "big picture" summary of the system under development and highlight the major components, their connectivity, and performance. The lower-level abstractions are suitable for formal systems analysis – for example, verification of component interface compatibilities and/or assessment of system performance through the use of simulation methods.

As engineering systems become increasingly complex the

 ISBN: 978-1-61208-319-3

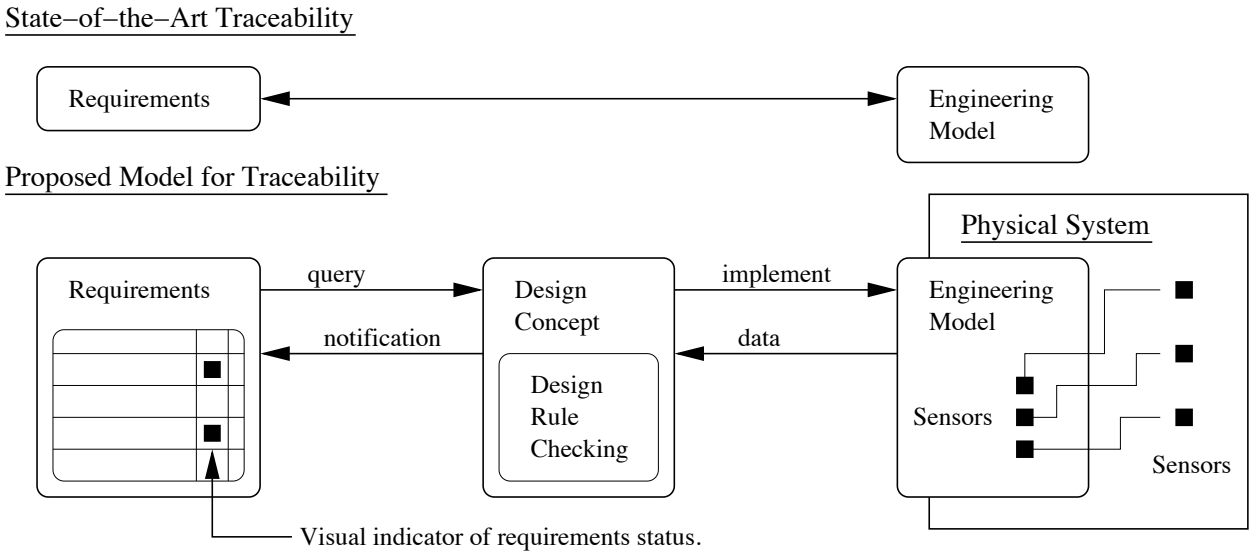

Figure 1. Schematics for: (top) state-of-the-art traceability, and (bottom) proposed model for ontology-enabled traceability for systems design and management.

need for automation arises. A key element of required capability is an ability to identify and manage requirements during the early phases of the system design process, where errors are cheapest and easiest to correct. We believe that methodologies for strategic approaches to design will employ semantic descriptions of application domains, and use ontologies and rule-based reasoning to enable validation of requirements, automated synthesis of potentially good design solutions, and communication (or mappings) among multiple disciplines [1], [2], [11]. Semantic Web-based technologies can play a central role in the design of tools to support these design methodologies. Present-day systems engineering methodologies and tools, including those associated with SysML [6] are not designed to handle projects in this way.

IV. Semantic Platform Infrastructure

The systems architecture for state-of-the-art requirements traceability and the proposed platform model is shown in the upper and lower sections of Figure 1. In state-of-the-art traceability mechanisms design requirements are connected directly to design solutions (i.e., objects in the engineering model). Our contention is that an alternative and potentially better approach is to satisfy a requirement by asking the basic question: What design concept (or group of design concepts) should I apply to satisfy a requirement? Design solutions are the instantiation/implementation of these concepts.

The proposed architecture is a platform because it contains collections of domain-specific ontologies and design rules that will be reusable across applications. In the lower half of Figure 1, the textual requirements, ontology, and engineering models provide distinct views of a design: (1) Requirements are a statement of "what is required." (2) Engineering models are a statement of "how the required functionality and performance might be achieved," and (3) Ontologies are a statement of "concepts justifying a tentative design solution." During design, mathematical and logical rules are derived from textual requirements which, in turn, are connected to elements in an engineering model. Evaluation of requirements can be include checks for satisfaction of system functionality and performance, as well as identification of conflicts in requirements themselves.

A key benefit of our approach is that design rule checking can be applied at the earliest stage possible – as long as sufficient data is available for the evaluation of rules, rule checking can commence; the textual requirements and engineering models need not be complete. During the system operation, key questions to be answered are: What other concepts are involved when a change occurs in sensing model? What requirement(s) might be violated when those concepts are involved in the change? To understand the inevitable conflicts and opportunities to conduct trade space studies, it is important to be able to trace back and understand cause-and-effect relationships between changes at system-component level and their affect on stakeholder requirements.

V. Work in Progress

Our testbed application area and driver for this work is performance-based modeling and design of energy-efficient building environments.

To this end, we are currently working toward the platform infrastructure implied by Figures 2 and 3. Figure 2 pulls together the different pieces of the proposed architecture shown in Figure 1. On the left-hand side the textual requirements are defined in terms of mathematical and logical rule expressions. From a CPS viewpoint, modern buildings contain networks for the arrangement of spaces throughout the building, for the fixed circulatory systems (e.g., power and hvac), for the dynamic circulatory systems (e.g., air and water flows), and for wired and wireless communications. Predictions of dynamic behavior will correspond to the solution of nonlinear differential algebraic equations (e.g., for water, air, and thermal flow) coupled to discrete equations (e.g., resulting from cyber decisions). While there is a desire for each network to operate as independently as possible, in practice the need for new forms of functionality will drive components from different network types to connect in a variety of ways. Within the

 ISBN: 978-1-61208-319-3

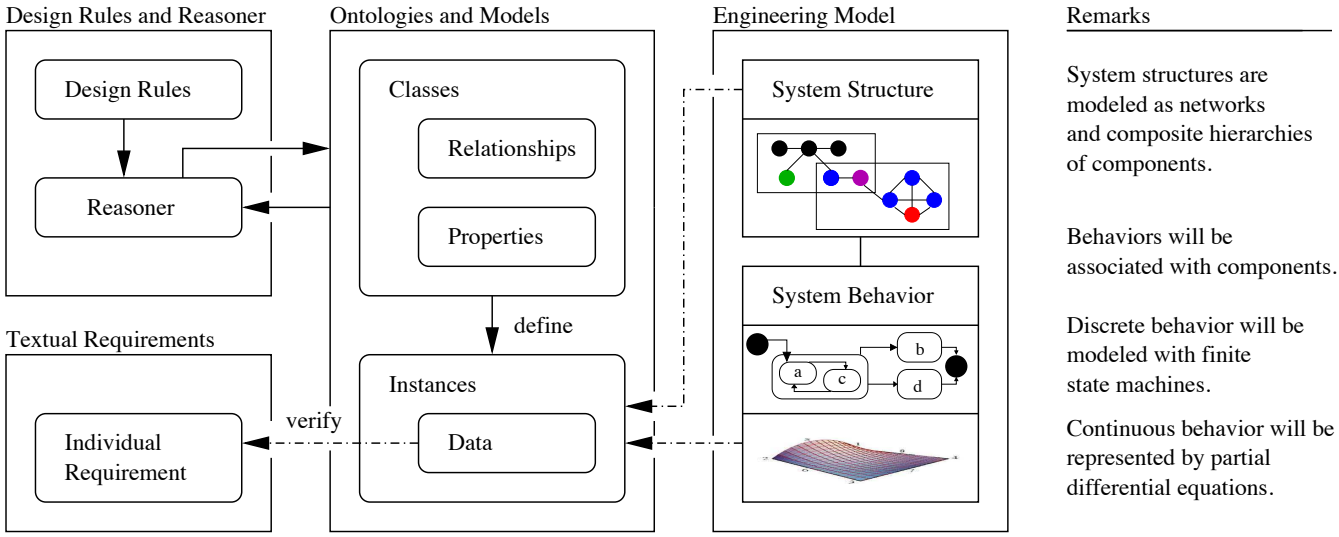

Figure 2. Framework for implementation of ontology-enabled traceability and design assessment.

building simulation community state-of-the-art dynamic simulation is defined by Modelica, and steady-state simulation by DOE-2 and eQuest. Building systems are modeled as networks of connected components, mathematical equations describe component-level behaviors, and models of system-level behavior are defined by graphs of equations synthesized from descriptions of component behavior and their connectivity.

Topic 1. Modeling and Reasoning with Ontologies. The upper section of Figure 1 shows state-of-the-art traceability mechanisms with requirements being connected directly to solutions. The proposed platform provides a base to implement requirements as rules and related parts of the engineering model as classes of the ontology. The lower section of Figure 1 shows the essential details of our three-part framework – textual requirements, ontology model and engineering model – for the implementation of ontology-enabled traceability mechanisms and rule-based design assessment.

Textual requirements are connected to the ontology model, and logical and mathematical design rules, and from there to the engineering model. Ontology models encompass the design concepts (ontology classes) in a domain, as well the relationships among them. Classes are qualified with properties (c.f., attributes in classes) to represent the consequence of constraint and design rule evaluations. Examples of valid relationships are: containment, composition, uses, and "is Kind of". These classes are placeholders for the data extracted from the engineering model. Individuals are the object counterpart of classes, with data and object property relationships leading the to RDF graph infrastructure. Each instance of an individual holds a specific set of values obtained the engineering model.

Rules serve the purpose of constraining the system operation and/or system design. They provide the mechanisms for early design validation, and ensuring the intended behavior is achieved at all the times during system operation. We are currently working with reasoners provided in the Jena API. A reasoner works with the RDF graph infrastructure and sets of user-defined rules to evaluate and further refine the RDF graph. Rule engines are triggered in response to any changes to the ontological model. This process assures that the model is consistent with respect to the existing rules. Traceability from ontologies to requirements is captured via implementation of the listeners that are notified as a result of change in the semantic model.

In a departure from past work, we are exploring the feasibility of creating built-in functions to capture and evaluate performance criteria, i.e., energy efficiency ratio of the HVAC system. A second potential use of built-in functions is as an interface to packages that provide system improvements through optimization and performance related queries. We note that a rule-based approach to problem solving is particularly beneficial when the application logic is dynamic (i.e., where a change in a policy needs to be immediately reflected throughout the application) and rules are imposed on the system by external entities [10], [12]. Both of these conditions apply to the design and management of engineering systems.

Topic 2. Modeling Engineering Systems with Components and Finite Elements. Modern buildings contain a variety of intertwined networks for the arrangement of architectural spaces and circulatory systems. They can also be thought of as a hierarchical arrangement of spaces – for example, buildings have floors, floor contain rooms, rooms contain furniture, and so forth. Unfortunately, the time-history analysis and control of building system performance is complicated by the need to model combinations of discrete and continuous behaviors. To address these concerns, we are exploring the feasibility of modeling system structures as networks and composite hierarchies of components, and implementing software prototypes through use of the composite design pattern. Behaviors will be associated with components. Components will be organized into component hierarchies – for example, in order to sense the water level in a tank, sensor components will be positioned inside water tank components. Discrete behaviors (e.g., the operation of hvac machinery) will be modeled with executable statecharts. Continuous behaviors will be represented by partial differential equations.

Our current strategy is to compute solutions to continuous

 ISBN: 978-1-61208-319-3

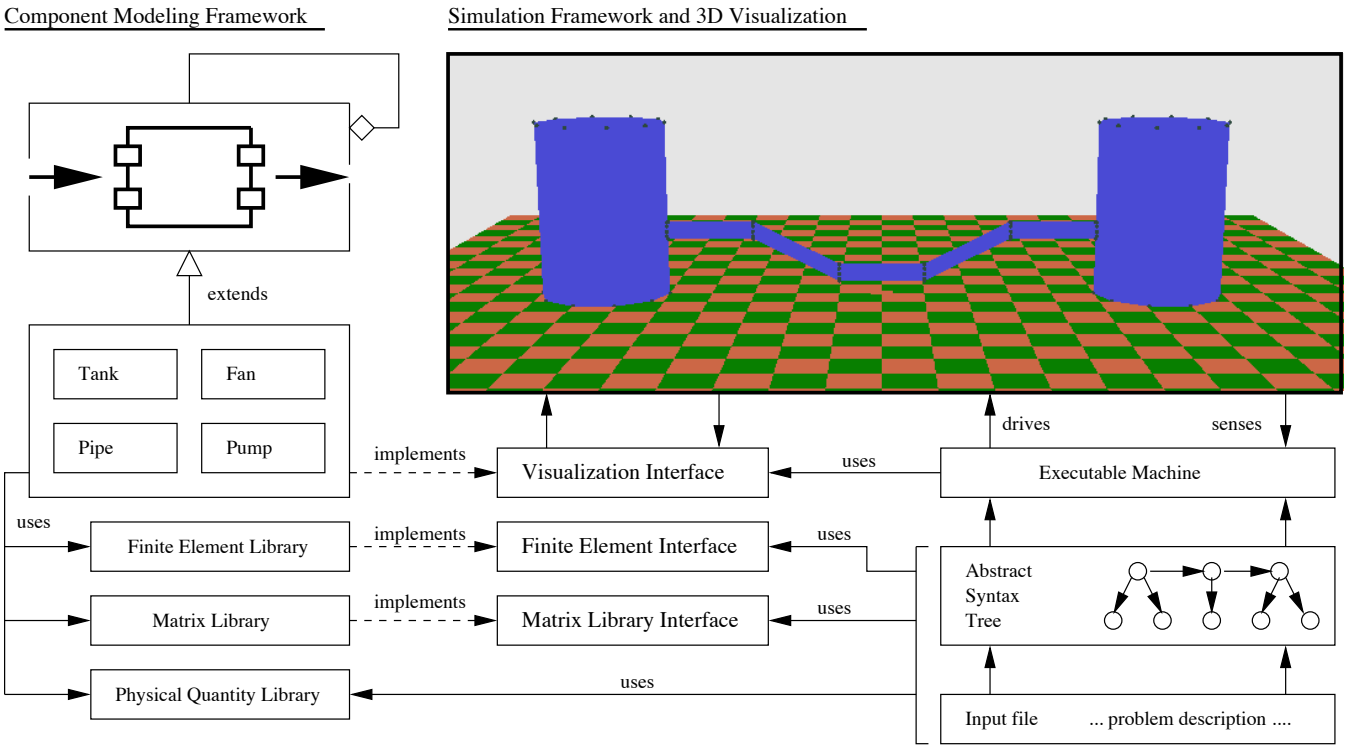

Figure 3. Architecture for modeling hvac systems as networks of connected components, and using finite element solution procedures for computing and visualizing time-history behavior.

behavior with the finite element method, although this certainly isn't the only way it could be done. In a departure from established approaches to engineering systems modeling, our goal is to design a single software architecture that can support this range of organization and behavioral abstractions. The key to making this work is software interfaces designed to support a multitude of system viewpoints – a visualize interface for 2D- and 3D- visualization, and a finite element interface for the description of element-level behaviors cast in a matrix format, a communications interface for sensor to controller communication. Since a Java object can implement and arbitrary number of interfaces, there is reason to believe this might actually work!

Topic 3. Scripting Language Support for Systems Integration. To support the organization and integration of physical components and computation of discrete and continuous behaviors, we are in the process of designing and implementing a scripting language where physical units are embedded within the basic data types, physical quantities and matrices of physical quantities, and scripting language constructs for controlling branching and looping control. For example, the fragment of code:

```
Force    = [ 2 N, 3 N, 4 N ];
Distance = [ 1 m; 2 m; 3 m ];
Work     = Force*Distance;
```

is a simple calculation for the work done by a force moving through a prescribed distance. The output is as follows:

```
Matrix: Force
row/col                  1             2             3
        units            N             N             N
   1          2.00000e+00  3.00000e+00  4.00000e+00

Matrix: Distance
row/col                  1
        units            m
   1          1.00000e+00
   2          2.00000e+00
   3          3.00000e+00

Matrix: Work
row/col                  1
        units          Jou
   1          2.00000e+01
```

The language supports the representation of differential equations in their discrete form, and solution via numerical integration techniques. For example, if the transient flow of fluid between the two tanks shown in Figure 3 is defined by:

$$\left[\frac{dU(t)}{dt}\right] + \left[\frac{f_1}{2D}\right] U(t)|U(t)| = \left[\frac{g}{L}\right] [H_1(t) - H_2(t)], \quad (1)$$

then the script:

```
velFluid  = pRoughness/(4.0*pRadius)*velOld*Abs(velOld)*dt;
velUpdate = g/pLength*( h01Old - h02Old )*dt;
velNew    = velOld + velUpdate - velFluid;
```

shows the essential details of computing the fluid velocity

update with Euler integration. During the executable phases of simulation (right-hand side of Figure 3), the runtime interpreter checks for dimensional consistency of terms in statements before proceeding with their evaluation.

VI. Related Work

Our program of research balances the need to build upon the work of others when it makes sense, while also moving boundaries to create new functionality and knowledge. Energy-efficient buildings are highly complex multidisciplinary entities. Part of the challenge we face stems from the difficulty in using a multitude of ontologies to adequately describe system structure, system behavior, and dependency relationships among domains. For example, an HVAC ontology contains concepts like fan, controller, damper, pipe and the relationships between them. Spatial ontologies contain concepts of room, zone and spatial information about the building. While each ontology will have its own distinct rule sets, the domains are loosely connected via common concepts and classes. This leads to the need for rules that are part cross-cutting and part domain-specific (e.g., If sensor measurement drops below 60 in room 1, then open the valve up to 80%).

An important facet of our work is use of Semantic Web technologies as both system models and mechanisms to derive system behavior. While the vast majority of Semantic Web literature has used ontologies to define system structure alone, this is slowly changing. Derler and co-workers explain, for example, how ontologies along with hybrid system modeling and simulation, concurrent models of computation can help us better address the challenges of modeling cyber-physical systems (CPSs). These challenges emerge from the inherited heterogeneity, concurrency, and sensitivity to timing of such systems. Domainspecific ontologies are used to strengthen modularity, and to combine model of system functionality with system architecture [5]. The Building Service Performance project proposes use of ontologies and rules sets to enhance modularity, and perform cross-domain information exchange and representation [9]. Koelle and Strijland are investigating the design and implementation of a software tool to support semantic-driven architecture with application of rules for security assurance of large systems in air navigation [8].

VII. Conclusion

The proposed semantic platform infrastructure will enhance systems engineering practice by lowering validations costs (through rule checking early in design), and providing support for performance assessment during the system operation. We envision cyber-physical systems having behaviors that are both distributed and concurrent, and defined by mixtures of local- and global- rule-based control. Further research is need to understand how the RDF graph structure and supporting software infrastructure can be extended in ways that make it more closely mimic object-oriented practices, specifically through increased use of generalization and access modifiers.

For the time-history behavior modeling and control of energy-efficient buildings, the finite element method is attractive because problem solutions can be formulated from first principles of engineering such as momentum balance. Solution procedures will be robust, scalable, and extensible to energy-balance calculations. Our plans are to design a family of component model interfaces (left-hand side of Figure 3), extend them for the implementation of a build components library (e.g., tanks, pipes, etc) and where needed, participate in finite element analysis, actuation, and control. The scripting language will act as the glue for systems integration and specification of simulation and visualization procedures.

Acknowledgment

The work reported here is part of a US National Institute of Science and Technology (NIST) funded program dedicated to the development of standards for CPS design, modeling, construction, verification and validation.

References

[1] M.A. Austin, V. Mayank V., N. and Shmunis Ontology-Based Validation of Connectivity Relationships in a Home Theater System. *International Journal of Intelligent Systems*, 21(10):1111–1125, October 2006.

[2] M.A. Austin, V. Mayank, and N. Shmunis PaladinRM: Graph-Based Visualization of Requirements Organized for Team-Based Design. *Systems Engineering: The Journal of the International Council on Systems Engineering*, 9(2):129–145, May 2006.

[3] P. Delgoshaei, and M.A. Austin Software Design Patterns for Ontology-Enabled Traceability. In *Conference on Systems Engineering Research (CSER 2011)*, Redondo Beach, Los Angeles, April 15-16 2011.

[4] P. Delgoshaei, and M.A. Austin Software Patterns for Traceability of Requirements to Finite-State Machine Behavior: Application to Rail Transit Systems Design and Management. In *22nd Annual International Symposium of The International Council on Systems Engineering (INCOSE 2012)*, Rome, Italy, 2012.

[5] P. Derler, E.A. Lee, and A.S. Sangiovanni-Vincentelli Modeling CyberPhysical Systems. *Proceedings of the IEEE*, 100, January 2012.

[6] S. Fridenthal, A. Moore, and R. Steiner *A Practical Guide to SysML*. MK-OMG, 2008.

[7] Jena - A Java API for RDF. See http://www.hpl.hp.com/semweb/. 2003.

[8] R. Koelle, and W. Strijland Semantic Driven Security Assurance for System Engineering in SESAR/NextGen. In *Integrated Communications, Navigation and Surveillance Conference (ICNS)*, 2013.

[9] D. Macpherson, and M. Raymond Ontology Across Building, Emergency, and Energy Standards. In *The Building Service Performance Project, Ontology Summit*, 2009.

[10] Q.H. Mahmoud Getting Started With the Java Rule Engine API (JSR 94): Toward Rule-Based Applications, 2005. Sun Microsystems. For more information, see http://java.sun.com/developer/technicalArticles/J2SE/JavaRule.html (Accessed, March 10, 2008).

[11] N. Nassar, and M.A. Austin Model-Based Systems Engineering Design and Trade-Off Analysis with RDF Graphs. In *11th Annual Conference on Systems Engineering Research (CSER 2013)*, Georgia Institute of Technology, Atlanta, GA, March 19-22 2013.

[12] G. Rudolf. Some Guidelines For Deciding Whether To Use A Rules Engine, 2003. Sandia National Labs. For more information see http://herzberg.ca.sandia.gov/guidelines.shtml (Accessed, March 10, 2008).

[13] w3 See http://www.w3.org/TR/owl-features/, February 2004.

 ISBN: 978-1-61208-319-3