Enabling Systems Modeling Language Authoring in a Collaborative Web-based Decision Support Tool

Daniel Browne*\textsuperscript{a}, Robert Kempf\textsuperscript{a}, Aaron Hansen\textsuperscript{a}, Michael O’Neal\textsuperscript{b}, William Yates\textsuperscript{b}

\textsuperscript{a}Georgia Tech Research Institute (GTRI), 250 14th St. NW, Atlanta, GA 30318, USA
\textsuperscript{b}Marine Corps Systems Command, Quantico, VA, USA

Abstract

The Systems Modeling Language (SysML) is gaining traction within the systems engineering community to define complex systems. To date all SysML authoring tools are single-user, thick-client software applications geared towards only model definition, forcing sequential model development and losing an opportunity to parallelize effort. Additionally, the completed model is not in and of itself able to execute tradespace exploration or constraint optimization. Most complex system development occurs in geographically disparate teams working on various components of the model concurrently, which introduces a process complexity in configuration management and slows progress. Although extensions to thick-client applications and separate software applications have attempted to solve these issues, the authors pose a different solution. This paper describes a web-based SysML authoring tool which provides real-time collaboration. The goal is to offer an experience where all users have up-to-date information and can concurrently modify the model. The authors describe how this tool has been incorporated with the Framework for Assessing Cost and Technology (FACT), an effort spearheaded by the Marine Corps System Command as a real-time, collaborative modeling, simulation and design web-service. FACT’s data schema is based on SysML, providing a unique use case implementation of the benefits of the SysML authoring tool described.

© 2013 The Authors. Published by Elsevier B.V. Open access under CC BY-NC-ND license.
Selection and/or peer-review under responsibility of Georgia Institute of Technology

Keywords: SysML; UML; real-time collaboration; systems-of-systems; modeling and simulation; web-based; decision support
1. Motivation

Complex system development, such as military vehicle acquisition programs, requires robust systems engineering tools to facilitate the design, tradespace analysis of design alternatives, and configuration/baseline management throughout the entire lifecycle. In the past decade the Systems Modeling Language (SysML) has gained traction as a means to describe complex systems-of-systems by being able to formally define subsystem interaction and organization, requirements, parametric relationships, and more [1]. The authors, however, argue that current authoring tools lack the necessary functionality to properly facilitate development of SysML models. Three primary shortcomings and a unique pairing with a Marine Corps modeling and simulation effort provided the motivation for the authors to design a web-based SysML authoring capability. Each of the four motivators is described in detail below.

1.1. What can I do with my SysML model?

SysML expands upon the Unified Modeling Language (UML) to break beyond software-centric design to include hardware components [2,3]. The specification provides a formal means to describe a system, most notably the decomposition and organization of the system components as well as the parametric relationships between value properties distributed throughout the systems. Once a model including Block Definition (BDD) and Parametric (PAR) diagrams has been defined, however, the model in and of itself is not executable. Some software packages offer plug-ins to tie to engines, such as Mathematica®, to execute the systems of equations. These plug-ins also allow integration with existing models such as Microsoft Excel® or Systems Tool Kit® (STK), however completing the integration can be cumbersome [4]. An example of such an implementation is ParaMagic® offered as a plug-in to NoMagic’s MagicDraw® [5,6]. Paramagic will execute properly formatted constraint blocks and the associated system of equations using Mathematica® as well as allow MatLab®, Excel®, or other models to be linked to it directly. Each thick-client authoring tool, however, offers a different means to do this which is not captured in the SysML XMI specification (further described in Section 2.2) [7]. For this reason, the authors argue that SysML models are not executable even with existing authoring tools.

Although the web-based SysML authoring tool described in this paper does not directly solve this shortcoming, the authors argue that a web-based SysML tool with an open application programming interface (API) provides a superior means to provide executability. Additionally, the current use case implementation pairs the SysML authoring capability with the Framework for Assessing Cost and Technology (FACT) [8,9]. FACT is a modeling and simulation framework, enabling real-time collaboration in a web environment, primarily geared towards conducting real-time tradespace analysis for complex systems-of-systems. FACT is spearheaded by the Marine Corps System Command to provide acquisition programs with “should cost, will cost” analysis. More details on FACT being a motivation for this work are described in section 1.4, but of note is that FACT’s data schema is based on SysML constructs.

In summary, the need to make SysML models executable was one of the first motivators for the authors to pursue an in-browser SysML authoring tool. Making a SysML model executable significantly increases the overall utility of such a model and provides a powerful tool for following a Model-Based Systems Engineering (MBSE) process.

1.2. I have a team of n engineers. How do they all work on the model?

All of the SysML authoring tools the authors identified are thick-client software applications. This means that a user must first receive approval to install the software, acquire a license (if required), install the software, and maintain the software up-to-date. Experience by the authors has shown some of these steps can be tedious and time/resource consuming especially for users on government-owned systems. By deploying in a web environment, a near-zero client footprint greatly broadens accessibility and removes the need for software approval and installation. Additionally, none of the thick-client authoring tools can, by themselves, allow for multiple users to be editing a single model at the same time. The best example of a solution the authors identified is No Magic’s Teamwork Server® [10]. Teamwork Server’s® collaboration is not available in real-time, but rather uses a more traditional version control system where users must first lock a portion of the model resulting in read-only access for all other contributors. The software does allow for a model to be divided into pieces so that multiple users can work
on different parts of the model at the same time, but other users are not aware of changes being made until a commit occurs. For highly interconnected models, defining those isolated regions of the model can be challenging, resulting in little parallelizable work. Once locking a portion of the model the user performs work individually. Changes are not available to the rest of the team until the author commits changes back to the server when completed.

The authors envision an authoring tool where $n$ users can simultaneously modify any part of the model at the same time, meaning that new blocks, block relationships, value properties, constraints, or other concepts can all be added/modified/removed concurrently. One of the main motivators for developing FACT, the modeling and simulation framework used as the implementation use case environment for this web-based authoring tool, was that during the design phase for an acquisition program different disciplines worked in stove-piped manners. This stove-piped approach resulted in sub-teams constantly working with outdated information. Cost, performance, reliability, etc. models are all tightly integrated through data but in the conventional design approach these disciplines are split apart. The authors argue that the same motivations for the development of FACT can in turn apply to the need for a more collaborative SysML authoring capability.

The web-browser based solution envisioned by the authors provides an experience like that of Google Docs® for SysML diagrams. Previous experience by the authors provides the basis for programmatically manipulating scalable vector graphics (SVG) diagrams and underlying data structures in real-time based on distributed users’ interaction. Providing truly real-time collaboration in conjunction with a strong configuration management solution enables rapid design definition and analysis in a means which does not exist elsewhere.

1.3. Does my system design satisfy my requirements?

SysML provides a general requirement construct which offers a title and human-readable descriptive statement. The authors have found that often, though not always, requirements can be represented in a quantitative manner. SysML’s requirement construct is insufficiently strong to map a value property of one block to a requirement which could define threshold and objective values. As FACT was being developed, this shortcoming was identified so the data schema utilized by FACT strengthened the requirement concept. The authors envision that by extending the current SysML specification requirement construct, any type of SysML model execution engine could easily provide an automated means to determine how an instantiation of a system meets/exceeds/falls short of its requirements. In order to do this in FACT, each quantitative requirement is associated with a value property which is a calculated value for a defined constraint. By enforcing this relationship, requirements move beyond just being human-readable descriptions and can provide insight into just how feasible a set of requirements is.

Within FACT, requirements are taken one step further where overall system qualitative scores can be provided. An additional concept of weight or importance is associated with each requirement. Using this value and comparing an actual instantiation’s value with the requirement threshold/objective, a qualitative score can be calculated for each requirement. By conducting a sum-product a total system score can be calculated to be used as a means of comparison against alternatives. This is an example of the benefit of strengthening the SysML requirement and its associate concepts.

To summarize, the identified shortcoming of existing SysML authoring tools (or SysML in general) is that the current requirement, satisfy, and verify constructs are not enforceable or executable. When creating a set of requirements using a defined specification such as SysML, one would expect that some off-the-shelf capability such as comparing an instance against that set of requirements should be provided. In the web-based authoring tool discussed in this paper, the authors argue that by strengthening the requirement-related concepts, addition of such verification functionality becomes straightforward.

1.4. Use Case: Framework for Assessing Cost and Technology

The United States Marine Corps Systems Command (MCSC) commissioned the development of the Framework for Assessing Cost and Technology (FACT) after the service had experienced multiple acquisition programs that ended due to cost overruns or overoptimistic performance criteria being deemed unattainable (given programmatic constraints). MCSC needed a framework to tie all of the disparate modeling efforts, currently conducted in a stove-piped fashion, together in order to expose the interdependencies between vehicle performance, cost (acquisition and operations & sustainment), reliability, maintainability and transportability. By providing a means for engineers to
see the impacts of their decisions on other disciplines, and vice versa, risks can be identified much earlier allowing for superior mitigation strategies to be put in place. FACT was required to be web-based so that information was centralized and accessible by anyone on a Navy Marine Corps Intranet (NMCI) workstation. By choosing a web deployment environment, it is relatively straightforward to enable real-time collaboration.

SysML was highly leveraged as the point of reference for the data schema implemented as FACT was initially developed. At the beginning of development an XMI (XML Metadata Interchange) specification had yet to be adopted, therefore the authors deemed it necessary to form a standard for storing the same data (and more) related to a SysML model in a database. Systems that were to use FACT as a means to conduct tradespace exploration first used a SysML model to capture the system definition including a work breakdown structure using a BDD and the various constraints in parametric diagrams. To import the SysML diagrams into the FACT schema a vendor-specific MagicDraw® XML (MDXML) parser was written. Once imported, the system representation could not be modified without direct interaction of a database administrator.

Because of this manual import and then lack of user modification, FACT is the perfect initial use case as a proof of concept for the in-browser SysML authoring tool. By incorporating the tool as a modular plug-in to the greater framework, users could go directly into FACT and start with a blank slate. For this paper, the inclusion of the SysML authoring capability in terms of FACT is the primary demonstration of capability.

2. Background

As the authors incorporated authoring of SysML diagrams into their engineering efforts, the need for a web-based authoring tool became evident. In order to follow a standard MBSE approach with a large, disparate team of design engineers, a tool allowing multiple simultaneous authors with built-in configuration management is almost necessary to work efficiently and effectively. For completeness of this paper, the authors offer a review of existing SysML authoring tools, overview of SysML and its associated XMI standard, and FACT as the demonstration platform for the web-based authoring tool.

2.1. Overview of Existing SysML Authoring Tools

Several commercial and open source authoring tools are available to aid systems engineers in the development, organization, and exploration of SysML models. Many of these tools are marketed as generic or UML modeling environments and have plug-ins or modules available to support SysML stereotypes by extending UML through the language’s profile mechanism. Assuming that the user is familiar with the SysML specification, key features of these authoring tools are that they present an intuitive interface for completing modeling tasks, store information about model diagrams and elements in an organized and easy-to-explore repository, and help enforce semantic and notational standards through model validation. Beyond the direct SysML modeling aspect, many of the tools provide additional plug-ins or extensions to tie the model to external tools such as Mathematica®, MatLab®, or Excel® which house parametric relationships.

No Magic’s MagicDraw® is an example of such a tool that typifies the nature of existing SysML modeling software. Developed primarily as a UML modeling environment, a plug-in expands the environment to include SysML metaclasses. An additional ParaMagic® [6] plug-in is available as a parametric solver that can integrate external M&S tools, such as MATLAB® code, Excel® spreadsheets or other external software with SysML models and execute their simulations. An additional version control system, called Teamwork Server® [10], is available to foster collaboration between engineers across different groups. This collaboration is not available in real-time, but rather uses a more traditional version control system where users perform work individually and must commit changes to a server in order to share them with a team. Furthermore, in order for a user to retrieve the changes committed by another a user, it is often the case that the entire model must be reloaded. The authors see this as an interruption to workflow as opposed to an automated update which is provided by the web-based authoring tool.

With the current state of SysML authoring tools available on the market and from open sources, none provide the means to directly update an online database to provide real-time collaboration. To the extent that existing software can provide even a basic means of version control and collaboration, all options require that user workstations be installed with the necessary software, which can become an expensive proposition requiring IT support for proper setup and maintenance.
2.2. XMI Specification

In late 2011 the Object Management Group (OMG) officially adopted an XMI specification providing a true standard for describing a SysML model [7]. Adoption of this specification allowed any SysML authoring tool vendor to understand, at least at the model level, a SysML model developed in any other authoring tool adhering to the specification. Prior, the means for storing the model data varied from tool to tool and tool (and with respect to the visualization aspect such as the diagrams, the same is still true). Because OMG has published this open source specification, new authoring tools, such as the web-based one discussed in this paper, have a straightforward approach for ingesting existing models or providing users the ability to move among authoring tools.

The first set of SysML parsers developed by the authors used one specific vendor’s format which closely followed, though not completely, the specification finally adopted by OMG. The authors envision a capability to import or export a SysML model which adheres to the XMI specification, encouraging users to try the web-based capability. These import and export capabilities are still being developed, but significant work has been completed in how to understand a model described using the XMI specification.

2.3. Framework for Assessing Cost and Technology (FACT)

FACT’s approach to model based systems engineering (MBSE) is in response to the need of defense acquisition programs for decision support tools that facilitate the concurrent management of system requirements, performance, cost and reliability in the context of rapid tradespace analysis. FACT’s development followed guidance from the Department of Defense, mandating that it be web-based and accessible from common computer workstations, be built entirely from open source software, and offer an open and extensible architecture [11,12]. Although the development of FACT was originally envisioned for vehicle acquisition programs, the overall process and application is independent of vehicles and can be applied to any system-of-systems.

Prior to development of an in-browser SysML authoring tool, a user’s first task to use the framework was to author a SysML model in a thick-client tool, specifically MagicDraw® [5]. The associated MDXML file is parsed via Python script to generate the data objects needed to populate the FACT database. With the SysML model architecture properly mapped to FACT’s data standard, and surrogate models linked to the design parameters and performance metrics, the FACT tool is ready to perform tradespace analyses on the system of interest. The web-based tool provides several capabilities for the collaborative management and exploration of system designs and their tradespaces.

A complete discussion of FACT’s capabilities is available in [8]. FACT is under continued development, but to date can offer the following capabilities:

- Create/Manage various requirement sets to assist in trading requirement values and weightings
- Create/Manage different instances of systems, allowing real-time comparison against requirement sets
- Conduct on-the-fly “what-if” scenarios for system designs with real-time feedback on performance, cost, etc. metrics of interest
- Conduct Monte Carlo simulations: vary system parameters (bottom-up) to generate probability distributions against achieving requirements; can also build scatterplots with top-down filtering to identify the feasible tradespace of alternatives
- Qualitative mapping of system attributes to requirements to determine a relative score/importance for each system parameter
- Compare system solutions against each other with real-time feedback as design parameters and requirements are modified by multiple users
- Generate qualitative comparison scores for components and entire systems based on requirement weightings and system performance

2.4. SysML

SysML is a general purpose graphical modeling language for MBSE applications that supports the specification, design, analysis and verification of a broad range of systems [2]. It is a subset and extension of the Object Management Group’s Unified Modeling Language™ (UML), the industry standard for modeling software-intensive
systems, which gives systems engineers the ability to represent system requirements, structure, behavior and properties using a formal diagram syntax. It includes nine diagram types, seven of which are directly borrowed or modified UML 2 diagrams, and two of which, requirement and parametric, are added to support requirements engineering and performance analysis, respectively. The diagrams can be thought of as windows into the model that offer different perspectives from which to view the system. SysML is not a methodology or tool for model development and execution; it simply provides a standardized set of semantics and notation for creating graphical system models.

SysML models consist of an interconnected set of model elements, or instances of SysML metaclasses, whose relationships to one another are visualized via symbols on the various diagram types. A model element must conform to the properties, constraints and relationships allowed by its metaclass, ensuring model consistency. Beyond using SysML to define systems, many commercial SysML tools allow users to connect constraints to external models to produce a more comprehensive and executable model. The authors have found that executing external models can, however, be somewhat cumbersome and inefficient; experience with such tools has affected the means by which certain concepts are implemented in the web-based SysML authoring tool.

3. Development of the Web-Based SysML Authoring Tool

The authors adopted an incremental approach to ensure successful development of the web-based SysML authoring tool. The initial goal was to provide the most common authoring capabilities to meet immediate needs. A primary overarching goal of the development, however, was to provide an interface with the adopted SysML XMI specification. Users could choose to enter or leave the web-based authoring tool for a thick-client at any point in time using a provided interface via this specification. Lastly, the authors wanted to ensure that the initial set of authoring capabilities was sufficient to utilize the tradespace analysis capabilities offered by FACT so that it was in some sense immediately executable at the completion of the SysML model.

3.1. Phase I Proof-of-Concept Implementation Goals

For the first iteration the authors identified a limited set of authoring capabilities, listed in Table 1. The primary driver for this set was determined as the minimal capabilities required to define a system work breakdown structure/organization and the associated set of parametric relationships. From the authors’ experience, defining these concepts are by far the most prevalent and important.

Table 1. Summary of proof-of-concept authoring capabilities for the web-based authoring tool

<table>
<thead>
<tr>
<th>SysML Diagram Type:</th>
<th>Block Definition (BDD)</th>
<th>Parametric (PAR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authoring Capabilities:</td>
<td>Create/Modify/Delete $&lt;block&gt;$</td>
<td>Create new $&lt;constraint&gt;$</td>
</tr>
<tr>
<td></td>
<td>Delete existing $&lt;block&gt;$</td>
<td>Add/Modify/Remove inputs and outputs which map to value properties</td>
</tr>
<tr>
<td></td>
<td>Define/Modify part properties (directed aggregation relationship)</td>
<td>Define equation or provide metadata for executable code</td>
</tr>
<tr>
<td></td>
<td>Create/Modify/Delete value property</td>
<td></td>
</tr>
</tbody>
</table>

3.2. Authoring Tool Requirements

The author’s leveraged stakeholder guidance [13] for the development of FACT to assist in initiating the development of this web-based SysML authoring tool. This guidance included the following requirements:

- Be assessable using a standard web-browser with zero client footprint
- Utilize open source software to the greatest extent possible
- Maintain data in a centralized repository to improve accessibility
- Utilize existing data standards
- Provide real-time collaboration among users
From this set of high-level requirements, the authors developed a set of derived requirements. These five high-level requirements mapped to this set of lower-level ones, generalized for brevity:

- Utilize open source Javascript libraries that work in browsers ranging from Internet Explorer 7 to Chrome
- Avoid technologies such as Flash or Silverlight that can require client installation, if possible
- Identify or define a data schema for storing SysML models in a Document Oriented Database such as MongoDB
- Utilize Extensible Messaging and Presence Protocol (XMPP) or similar technology to provide real-time collaboration

3.3. Implementation

In order to support a wide range of web browsers (specifically older though still commonly used ones such as Internet Explorer (IE) 7 and 8), certain technologies such as scalable vector graphics (SVG) could not be used directly. To provide a desktop-like experience in a web browser, the authors identified a collection of Javascript libraries. The most highly leveraged libraries include JointJS, Raphaël, and jQuery [14,15,16]. Each of these is open source and used widely by other web applications. JointJS provides a library to build UML diagram elements as well as make each component movable on the page. The authors extended the JointJS library to include necessary SysML diagram elements. Raphaël abstracts the creation of visual elements and will create either vector markup language (VML) or SVG files depending on the browser being used by the client; for example in IE7 VML is generated while if using Google Chrome an SVG is generated. JointJS utilizes Raphaël to draw the appropriate diagram elements. Finally jQuery is actually a collection of libraries that greatly simplifies how to modify the Document Object Model (DOM) (i.e. the web page) as a user interacts with it. Selecting these technologies ensured that just about any web browser in use today could be used to access the authoring tool.

For the collaboration piece, experience with FACT and other web applications led the authors to select a solution which could utilize either an XMPP server or a standard asynchronous server-side polling service [17]. Both solutions allow a user to receive updates automatically and instantaneously from a server as a result of other users’ interactions with a model. XMPP is the technology utilized by most chat clients and Google Docs® to provide real-time user experiences. An important note about both of these approaches is that the primary load for handling the distribution of messages to simultaneous users remains on the server, not with the client, a critical design choice for increasing accessibility and user experience.

3.4. Data Schema

Since SysML is a graphically based standard, prior to release of the XMI specification the most appropriate means to store the data would have been in an SVG or similar format. Even with an XMI specification, storing an entire model as a single XML document poses challenges. Making incremental changes to either format is not straightforward, therefore the authors determined the best approach was to define a data schema for storing a SysML model in a database. A schema was defined using JavaScript Object Notation (JSON) which lends itself well to being stored in the NoSQL database MongoDB [18,19]. The authors are beginning the process of releasing this standard for consideration by the broader community of interest to be eventually published. Table 2 summarizes the mapping of SysML concepts to how they are represented in the JSON data schema.

Table 2. Mapping of the SysML concept to an object notation data schema

<table>
<thead>
<tr>
<th>SysML Concept Name</th>
<th>Data Schema Concept Name</th>
<th>Attributes (for object types)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block</td>
<td>system - a type of object</td>
<td>• _id &lt;br&gt; • name &lt;br&gt; • wbs_id &lt;br&gt; • description &lt;br&gt; • parent (see Aggregation below)</td>
<td>A &lt;&lt;block&gt;&gt; in SysML is the most basic concept which is used as the primary stereotype for most SysML concepts (including many of those listed in the rows below). The generic &lt;&lt;block&gt;&gt; stereotype is used for systems and houses the associated part properties / subsystems (parts), value properties, constraints, and requirements. The hierarchy of Systems and Parts provides the WBS for a vehicle.</td>
</tr>
</tbody>
</table>
### Aggregation / Part Property
- **parent** – an attribute of a system
- **n/a**

This concept defines a relationship between two systems, where one system is the parent and one is the child. A system can have multiple children but only 0 or 1 parents. This is a stronger enforcement than SysML itself.

### Value Property
- **system_attribute** – a type of object
- **id**
- **name**
- **system (_id of system)**
- **i/o (input or output?)**
- **default_value**
- **limits (min/max)**

* if i/o = True

An atomic property that has an associated unit and can either be defined as a user input or set by a constraint’s output.

### Constraint
- **constraint** – a type of object
- **id**
- **name**
- **system Prototype**
- **inputs (list of system_attribute _ids)**
- **outputs (list of system_attribute _ids)**
- **package**
- **module**
- **function_name**
- **function**

* if pointer to code
** if storing algebraic relationship in object directly

A parametric relationship that associates inputs and outputs (sets of system attributes). To increase flexibility, constraints can simply contain pointers to underlying Python code so that any model complexity can be executed or they can contain algebraic relationships in the object itself.

### Requirement
- **requirement** – a type of object
- **id**
- **system_attribute importance**
- **description**
- **score_fxns (pairings of threshold/object values and weights)**

In SysML a <<requirement>> provides limited information (mostly human-readable text only), and therefore has been expanded and strengthened. The information added by this JSON data standard can be represented in SysML using additional constructs. A requirement is associated with a system_attribute such that it can be verified using logic within the tool.

### Instance
- **prototype_instance** – a type of object which defines a set of system instances
- **id**
- **name**
- **system Prototype**
- **description**

A <<instance>> is an instantiation of a <<block>>. Similarly, a prototype_instance is an instantiation of a system-of-systems and a system_instance is an instantiation of a system. The system Prototype and system concepts are the metadata while the instances provide specific data such as values for system_attributes.

### System Instance
- **system_instance** – a type of object which adheres to a definition of a system
- **id**
- **system (_id of system)**
- **name**
- **description**
- **values (list providing single value for each system_attribute)**

A <<instance>> is an instantiation of a <<block>>. Similarly, a prototype_instance is an instantiation of a system-of-systems and a system_instance is an instantiation of a system. The system Prototype and system concepts are the metadata while the instances provide specific data such as values for system_attributes.

### 3.5. The User Interface

Moving from a desktop-client to a web interface does require modifications to how a user interacts with the authoring tool. Using JavaScript libraries like JointJS and jQuery has allowed for similar interactions and views to be created like those used in the thick-client tools. Figure 1 provides two example screenshots. In the lower foreground is a cropped view of a large BDD. For each <<block>> a popup panel provides the list of value properties and editing capabilities. In the upper background an example PAR diagram is provided. Similar popup panels to those seen in the BDD allow for editing each <<constraint>> individually. Future work will allow for a user to aggregate PAR diagrams so that if an input to one <<constraint>> is an output from another, that upstream constraint be viewed at the same time.
4. Conclusion and Future Work

The web-based SysML authoring tool discussed in this paper challenges the current thick-client desktop application paradigm implemented by all existing tools. Utilizing a single-user paradigm greatly limits the development processes a design team can choose to implement because of the limitations in configuration management, simultaneous authoring, and final usefulness of the model via direct execution. Although many of the existing commercial products have provided extensions to the authoring tools, the authors argue that moving to a web-based authoring tool provides a superior means to enable collaboration and integrate with modeling and simulation frameworks such as FACT. This paper has explained the need for a real-time collaborative SysML authoring tool in an interactive web environment and outlined the methodology and considerations required to develop and implement such a tool.

As an initial proof of concept, the web-based tool allows for authoring of block definition and parametric diagrams. Specifically, the creation/modification/deletion of blocks, value properties, aggregation relationships, constraints, and ports is supported. An equation builder is currently in development to accompany the constraint authoring capability so that the SysML tool itself, isolated from any M&S framework such as FACT, could still execute simple parametric relationships. The authors intend to continue the effort by first focusing on requirements (with the stronger enforcement of meaning) and moving onto activity and internal block diagrams.

Finally, an important feature of the tool, while still in development, is the ability to import and export of SysML
models in the SysML XMI specification format [7]. By supporting this standard, models created or edited using the in-browser tool can then also be exported for use with other authoring tools. Care will be taken in handling concepts not yet supported by the authoring tool in hopes of attracting a user-base for feedback toward application improvement. With the eventual offering of a full-feature in-browser SysML authoring tool, MBSE design teams will have an opportunity to explore more agile and rapid development cycles that previously were only available to software development teams.

Acknowledgements

The authors wish to thank the Marine Corps Systems Command for funding principal development of the SysML authoring tool as well as the Framework for Assessing Cost and Technology used as a demonstration platform. Additionally, the authors thank Tommer Ender, Jennifer Delockery, Brian Taylor and John Schaberg for their assistance in developing the in-browser SysML authoring capability and authoring of this paper.

References

11. Assistant Secretary of Defense (Networks and Information Integration) and DoD Chief Information Officer, DoD Information Assurance Certification and Accreditation Process (DIACAP), Department of Defense Instruction 8510.01, 2007.
12. Assistant Secretary of Defense (Networks and Information Integration) and Chief Information Officer, Clarifying Guidance Regarding Open Source Software (OSS), Department of Defense Memorandum, 2009.