Proceedings of the 2015 Winter Simulation Conference
L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, eds.

MULTIFORMALISM, MULTIRESOLUTION, MULTISCALE MODELING

Fatma Dandashi
Vinay Lakshminarayan
Nancy Schult

MITRE Corp.
7515 Colshire Blvd
McLean, VA 22102, USA

ABSTRACT

The modeling and simulation (M&S) landscape for systems engineers is complex. This complexity is due to a variety of factors including the diversity of system components over a range of domains which are modeled at multiple scales using varied modeling formalisms. There is a need to chain system models developed using varying formalisms, resolutions or scales. An ideal Multiformalism, Multiresolution, Multiscale Modeling (M4) environment would provide a singular (or small set of) modeling formalisms from which to derive representations (models) across formalism, resolution and scale. This paper describes a research activity to translate Systems Modeling Language (SysML) models to the Joint Communications Simulation System (JCSS). A deliverable from this activity will aid the Department of Defense (DoD) in the design and analysis of the Joint Information Environment (JIE).

1 INTRODUCTION

The modeling and simulation (M&S) landscape for systems engineers is complex. This complexity is due to a variety of factors, including the diversity of system components, in particular the presence of hardware, software, and cognitive elements. This implies the use of models for a range of domains, some of which can be highly domain-specific (e.g., in software, hydraulics, wireless networking). In addition, modeling solutions must support Systems Engineering (SE) analyses at multiple scales. Currently, separate models are expressed in different modeling formalisms, scales, and tools with proprietary model representation formats; separate models for system components that represent different domains (e.g., software, wireless networking) are not interconnected and the analysis is not holistic over the models' scales and domains. As model content is communicated from one modeler using one formalism to another modeler using another formalism, interrelationship information may get lost or misrepresented. There is a need to "chain" system models developed using varying formalisms, resolutions or scales. An ideal Multiformalism, Multiresolution, Multiscale Modeling (M4) environment would provide a singular (or small set of) modeling formalisms from which to derive representations (models) across formalism, resolution and scale.

This paper provides an interim report on a MITRE internal research activity where a Joint Communications Simulation System (JCSS) profile was developed to model network devices and other network-related modeling elements and relationships, customized for the Department of Defense (DoD) domain. A *profile* in this context is a tailoring of the Object Management Group (OMG)'s Systems Modeling Language (SysML) to the specific domain of network modeling. Several profiles are published as specifications by OMG (OMG 2015), for example the profile for Software Radio Components (SDRP, 2007). Applying the JCSS profile, a SysML model is exported as an Extensible Markup Language (XML) Metadata Interchange (XMI) (XMI 2015) file into a JCSS-formatted XML file that can be directly imported into JCSS. The idea is to enable the import of a SysML model to JCSS in order to quickly populate a JCSS model with operational facilities, network devices, links and other data, for further analysis. The deliverable from this research activity will aid DoD in the design and analysis of the Joint Information Environment (JIE). The ability to import and export model information from one formalism (SysML) to another (JCSS) and vice versa supports two Defense Information Systems Agency (DISA) needs:

1. An ability to reverse engineer an existing JCSS model to generate an equivalent SysML for the purpose of producing documentation/ or communication across a diverse group of stakeholders.
2. An ability to generate network models (in JCSS) from SysML models for the purpose of conducting further network performance analysis and simulation.

The long term goal of this MITRE research is to produce a modeling environment that supports the broadest range of Systems Engineering technical processes such as requirements analysis, design, implementation, and testing. Part of this goal is to have discrete event simulation tools import and export model representations using an industry standard (to be developed), similar to the manner that SysML tool vendors have adopted the SysML XMI specification as the de facto standard for the exchange of SysML models. With such a standard in place, development of a general purpose algorithm to import a SysML model using the semiformal SysML machine readable specification (SysML 2014), and export that model into discrete event simulation formalism/standard would be straightforward.

1.1 Related Work

A number of publications exist in the literature that describe research, methods and techniques to transform a SysML model to another formalism for conducting further M&S activities (see Kapos et al 2014b, Batarseh and McGinnis 2012, McGinnis and Ustun 2009). The research is needed due to the use of SysML as an SE modeling language (see briefings at various Model-Based SE workshops (MBSE Wiki 2015), combined with the need to construct and run simulations on various aspects of the SysML model that the SysML language does not support. Several reasons for this exist, the first being that SysML was designed to be a general purpose SE modeling language to document and analyze existing and proposed systems. Consequently, SysML was not based on an M&S formalism such as Discrete Event System Specification (DEVS) (Zeigler et al 2000) and was not intended to replace M&S modeling formalisms. SysML does not support time-flow mechanisms, random number generation, statistics collection, or verification and validation beyond modeling time aspects using sequence diagrams for example, static modeling of state machines, and including verification relationships that tie use cases to the relevant requirements statements. To simulate, another formalism and other tools are needed. Published research explores relating SysML to the Foundational subset for executable UML (fUML) language (fUML 2013), Colored Petri Nets (Wang and Dagli 2008), or DEVS (Kapos et al 2014a, Wang and Zhao 2012, Wang et al 2014). Related research in multi-paradigm modeling is discussed in (Mosterman and Vangheluwe 2004).

1.2 Joint Communications Simulation System (JCSS)

JCSS (JCSS 2015) is a modeling and simulation desktop application for the DoD. The latest release of JCSS is version 14.0 and is based on OPNET Modeler 17.5 PL6 (Riverbed 2015). JCSS v14.0 provides an integrated ability to analyze communication networks and a simulation capability with databases and underlying models so that studies can be consistent throughout the Unified Combatant Commands, Services, and others within the Command, Control, Communications, and Computer Systems (C4) community. By simulating communications effects of existing or planned networks that support warfighter operations, JCSS helps to quantify risks and identify C4 deficiencies prior to execution or change implementation. Results from JCSS simulations and analyses also provide support for prudent acquisition decisions.

1.3 Systems Modeling Language (SysML)

The SysML modeling language is used to document and analyze existing and proposed systems by an increasingly large number of systems engineers supporting DoD. SysML is a general-purpose graphical modeling language for specifying, analyzing, designing, and verifying complex systems that may include hardware, software, information, personnel, procedures, and facilities. In particular, the language provides graphical representations with a semantic foundation for modeling system requirements, behavior, structure, and parametrics, which is used to integrate with other engineering analysis models. SysML represents a subset of the Unified Modeling Language (UML) with extensions needed to satisfy the requirements of the *UML for Systems Engineering* call for proposals. SysML leverages XMI to exchange modeling data between tools, and is also intended to be compatible with the evolving ISO 10303-233 (ISO 10303-233:2012) systems engineering data interchange standard. SysML v1.0 was adopted by the OMG as a standard in July 2006. The current release is SysML v1.4 which was adopted by OMG March 2014 (SysML 2014).

2 RESEARCH PROCESS

This research activity started by developing a simple network communications model using JCSS; generating an XML file from that model, and using the XML file to reverse engineer an equivalent SysML model. This step was part of the learning process for team members who were unfamiliar with JCSS in order to learn about the elements and relationships defined in JCSS (i.e., discover the underlying metamodel).

Using Python script (Python 2015), a translator was developed to parse the XMI file and generate an XML file that can be directly imported into JCSS. The translator identifies instances of stereotyped JCSS profile elements present in the XMI and builds corresponding XML elements suitable for import into JCSS. Figure 1 identifies the major steps in this process. A system designer builds a representation of the desired network using SysML and the JCSS profile, shown at the bottom left. After the design is complete, the SysML model is exported to an XMI format. The JCSS XML translation of the system design is imported into the JCSS tool.

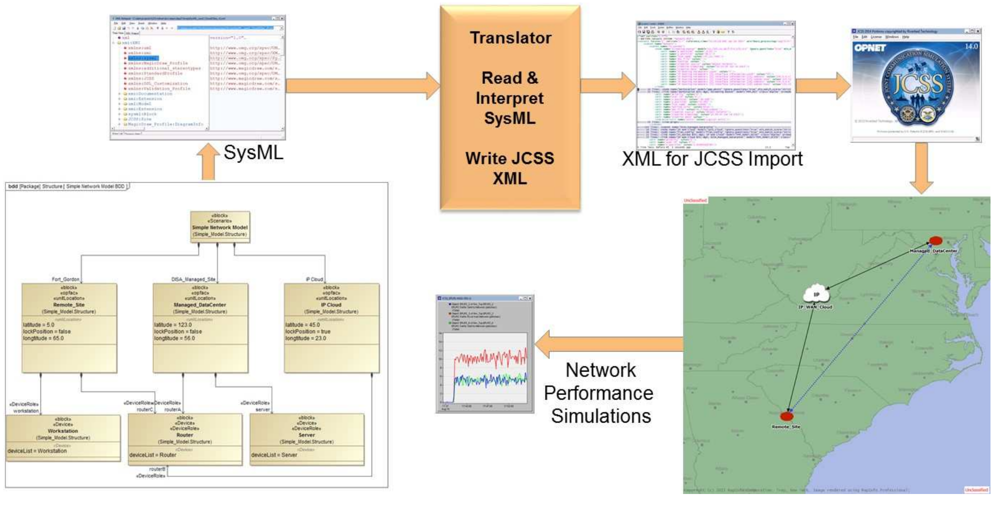


Figure 1: Model formalism transformation process.

3 DETAILED APPROACH

Our approach was to create a simple model in JCSS as a starting point (see Figure 2). The model consisted of two operational facilities: a Managed_DataCenter and RemoteSite, communicating over an IP-cloud infrastructure.

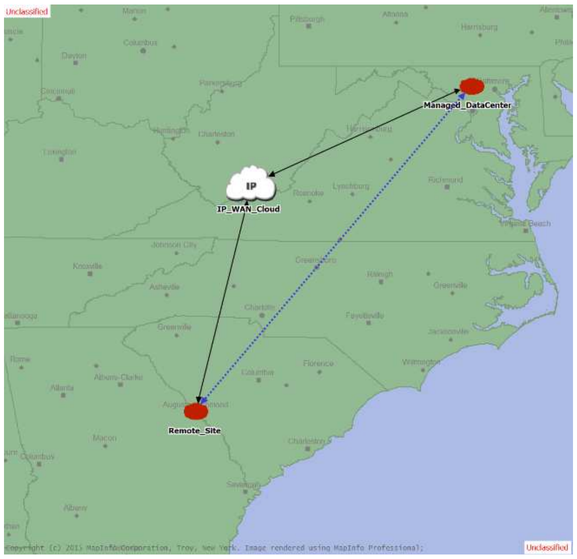
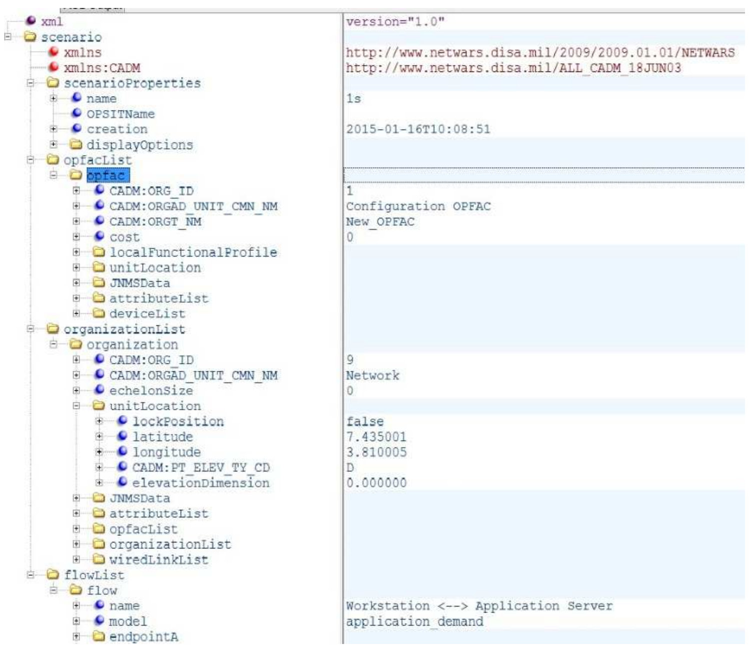


Figure 2: JCSS model.

By generating the corresponding XML from this model (see Figure 3), we identified the hierarchy for elements that are defined and stored in a JCSS model.



```
xml                                   version="1.0"
scenario
  xmlns                               http://www.netwars.disa.mil/2009/2009.01.01/NETWARS
  xmlns:CADM                          http://www.netwars.disa.mil/ALL_CADM_18JUN03
  scenarioProperties
    name                              1s
    OPSITName
    creation                          2015-01-16T10:08:51
    displayOptions
  opfacList
    opfac
      CADM:ORG_ID                     1
      CADM:ORGAD_UNIT_CMN_NM          Configuration OPFAC
      CADM:ORGT_NM                    New_OPFAC
      cost                            0
      localFunctionalProfile
      unitLocation
      JNMSData
      attributeList
      deviceList
  organizationList
    organization
      CADM:ORG_ID                     9
      CADM:ORGAD_UNIT_CMN_NM          Network
      echelonSize                     0
      unitLocation
        lockPosition                  false
        latitude                      7.435001
        longitude                     3.810005
        CADM:PT_ELEV_TY_CD            D
        elevationDimension            0.000000
      JNMSData
      attributeList
      opfacList
      organizationList
      wiredLinkList
  flowList
    flow
      name                            Workstation <--> Application Server
      model                           application_demand
      endpointA
```

Figure 3: XML file generated from JCSS model.

The top level element that owns the rest of the model elements is called Scenario. Other elements of note are Organization with requisite attributes for longitude, latitude, etc. Wired links are defined to represent connectivity between organizations and through the network infrastructure, while an Information Exchange Requirement (IER) represents the logical interface for which information and data is specified in the model.

3.1 SysML Profile

As a first profiling step, a subset of the JCSS elements defined in the JCSS XML Schema Definitions (XSD) were chosen: *Scenario, Organization, OpFac, Device, unitLocation, Location, Flow* and *WiredLink*. JCSS defined elements have a predefined set of attributes (e.g., classification level associated with *wiredLink*). Subsequently, a profile for a simplified JCSS metamodel, containing this set of elements and a subset of their attributes was developed using No Magic's MagicDraw (No Magic 2015) (see Figure 4).

In addition, some enumerated data types for classification levels, models of routers, servers and workstations were included in the JCSS profile. These elements and data types do not represent the complete list of JCSS defined elements nor do the enumerated types include a complete listing of values that are predefined in the JCSS tool, but this small number of elements is a sufficient representative set for the purposes of the first step.

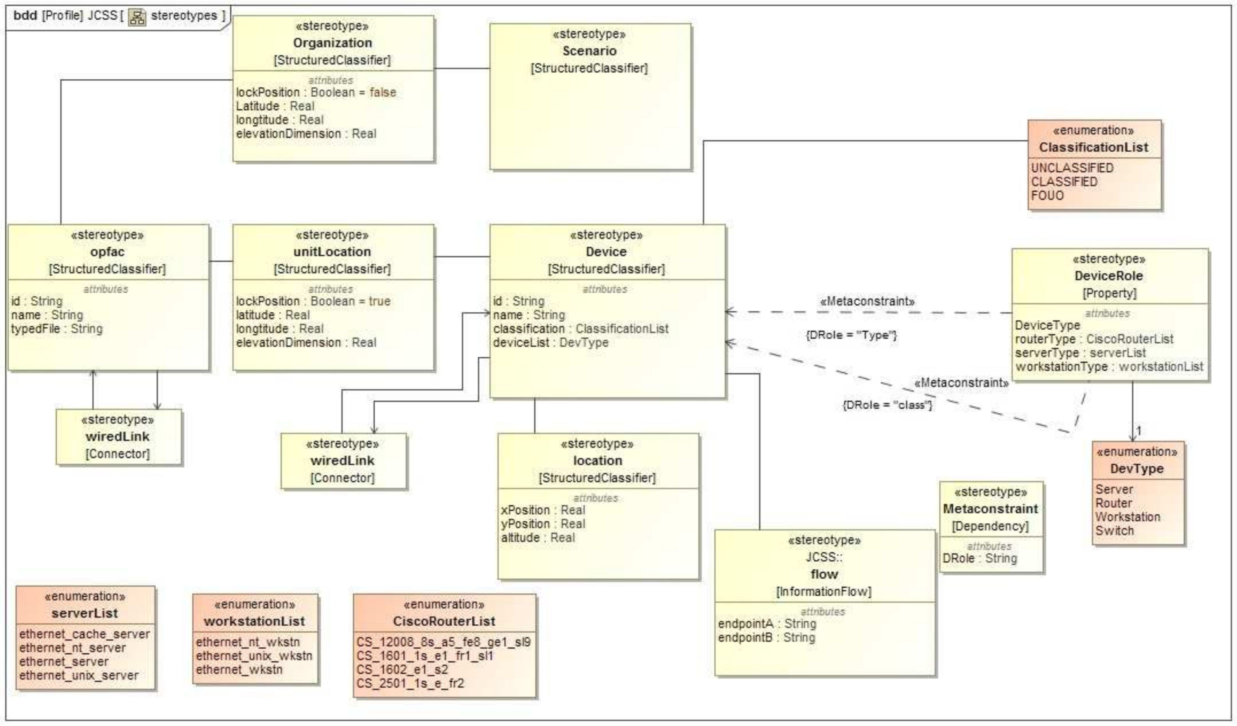

bdd [Profile] JCSS [stereotypes]
«stereotype» Organization [StructuredClassifier]
attributes
lockPosition : Boolean = false
Latitude : Real
longtitude : Real
elevationDimension : Real
«stereotype» Scenario [StructuredClassifier]
«enumeration» ClassificationList
UNCLASSIFIED
CLASSIFIED
FOUO
«stereotype» opfac [StructuredClassifier]
attributes
id : String
name : String
typedFile : String
«stereotype» unitLocation [StructuredClassifier]
attributes
lockPosition : Boolean = true
latitude : Real
longtitude : Real
elevationDimension : Real
«stereotype» Device [StructuredClassifier]
attributes
id : String
name : String
classification : ClassificationList
deviceList : DevType
«Metaconstraint»
{DRole = "Type"}
«Metaconstraint»
{DRole = "class"}
«stereotype» DeviceRole [Property]
attributes
DeviceType
routerType : CiscoRouterList
serverType : serverList
workstationType : workstationList
1
«stereotype» wiredLink [Connector]
«stereotype» wiredLink [Connector]
«stereotype» location [StructuredClassifier]
attributes
xPosition : Real
yPosition : Real
altitude : Real
«stereotype» Metaconstraint [Dependency]
attributes
DRole : String
«enumeration» DevType
Server
Router
Workstation
Switch
«stereotype» JCSS:: flow [InformationFlow]
attributes
endpointA : String
endpointB : String
«enumeration» serverList
ethernet_cache_server
ethernet_nt_server
ethernet_server
ethernet_unix_server
«enumeration» workstationList
ethernet_nt_wkstn
ethernet_unix_wkstn
ethernet_wkstn
«enumeration» CiscoRouterList
CS_12008_8s_a5_fe8_ge1_sl9
CS_1601_1s_e1_fr1_sl1
CS_1602_e1_s2
CS_2501_1s_e_fr2

Figure 4: JCSS profiled elements.

3.2 JCSS Profile Application

Next we applied the profile to create a network model using SysML. A Block Definition Diagram (BDD) was developed in SysML, using the profiled elements for scenario, *OpFac*, etc. (see Figure 5). A SysML BDD is the declarative part of a model and represents the bill of materials, that is, all system components needed to model a system and their composition (decomposition) relationships.

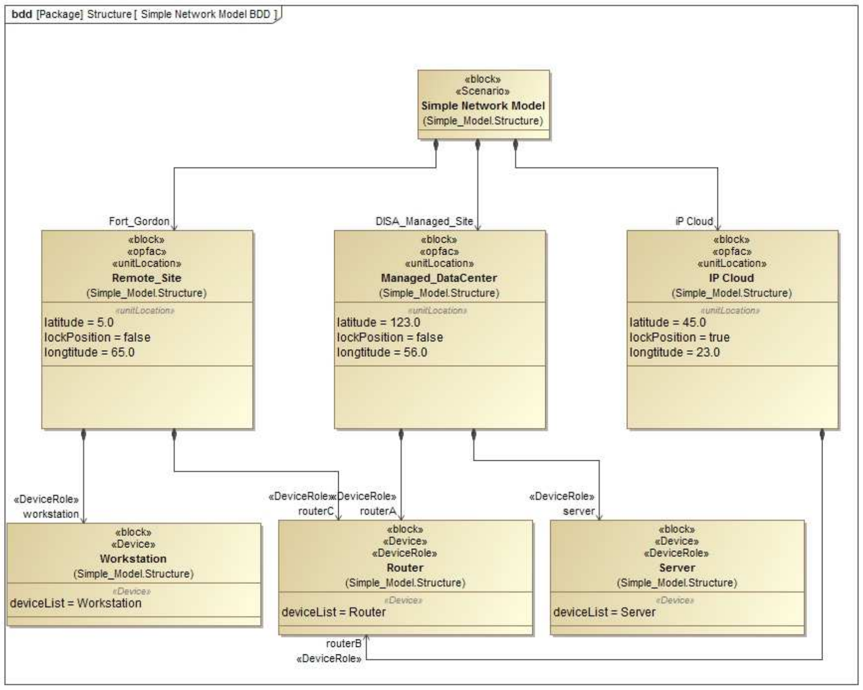

bdd [Package] Structure [Simple Network Model BDD]
«block» «Scenario» Simple Network Model (Simple_Model.Structure)
Fort_Gordon
DISA_Managed_Site
IP Cloud
«block» «opfac» «unitLocation» Remote_Site (Simple_Model.Structure)
«unitLocation»
latitude = 5.0
lockPosition = false
longtitude = 65.0
«block» «opfac» «unitLocation» Managed_DataCenter (Simple_Model.Structure)
«unitLocation»
latitude = 123.0
lockPosition = false
longtitude = 56.0
«block» «opfac» «unitLocation» IP Cloud (Simple_Model.Structure)
«unitLocation»
latitude = 45.0
lockPosition = true
longtitude = 23.0
«DeviceRole» workstation
«DeviceRole» routerC
«DeviceRole» routerA
«DeviceRole» server
«block» «Device» Workstation (Simple_Model.Structure)
«Device»
deviceList = Workstation
«block» «Device» «DeviceRole» Router (Simple_Model.Structure)
«Device»
deviceList = Router
«block» «Device» «DeviceRole» Server (Simple_Model.Structure)
«Device»
deviceList = Server
routerB «DeviceRole»

Figure 5: BDD for the network model.

Figure 6 is the corresponding Internal Block diagram (IBD). It shows how the system parts connect to each other and how they might pass data and other types of flows among them. This ibd was also constructed through an application of the JCSS profile, using the stereotype *wiredLink*. Following the development of this model, the SysML model information was exported as an XMI file.

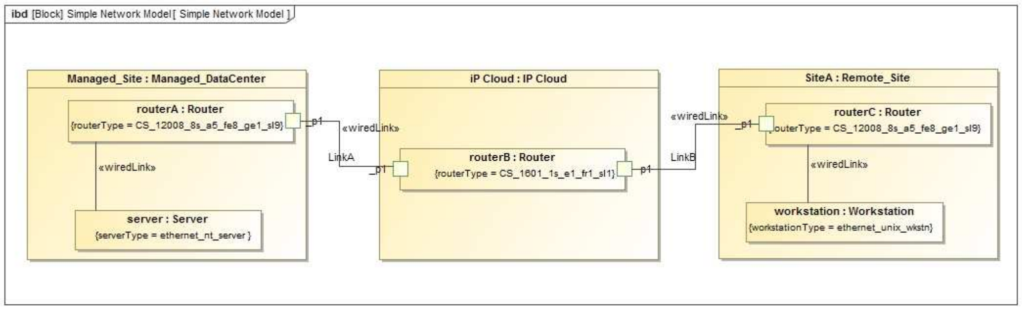

ibd [Block] Simple Network Model [Simple Network Model]
Managed_Site : Managed_DataCenter
routerA : Router
{routerType = CS_12008_8s_a5_fe8_ge1_sl9}
p1
«wiredLink»
server : Server
{serverType = ethernet_nt_server}
iP Cloud : IP Cloud
«wiredLink»
LinkA
p1
routerB : Router
{routerType = CS_1601_1s_e1_fr1_sl1}
p1
LinkB
«wiredLink»
SiteA : Remote_Site
p1
routerC : Router
{routerType = CS_12008_8s_a5_fe8_ge1_sl9}
«wiredLink»
workstation : Workstation
{workstationType = ethernet_unix_wkstn}

Figure 6: IBD for the network model.

The hierarchical, geographical, network connectivity, and information flow relationships between the system objects can be visualized in the JCSS GUI. Some iterations over the JCSS model may be needed to revise the design and correct any problems that may become apparent in these relationships. The design iterations should result in a functional network instance whose performance can be evaluated using techniques provided by the JCSS software. The performance evaluations could suggest the need for additional revisions to the design to achieve a desired system performance.

3.3 XMI to JCSS XML Translator

As described in Section 1, this research activity to enable evaluation of network performance of SysML models started by building a simple test harness network in JCSS, and generating an XML that JCSS is able to import. This XML sample was reverse engineered to build a SysML representation of the test harness network. Also resulting from the reverse engineering activity was an understanding of the JCSS XML structure, including the location of XML tags, necessary attributes, and their child tags. This knowledge was encoded into a text file and is used by the translation script. We anticipate that in the future, XSD files provided with the JCSS software can be used for this purpose. The text file representation of the JCSS XML structure is described in the following paragraphs.

We developed a table to describe the structure of a JCSS XML file. Each record in the table has 6 fields, as shown in Table 1. Each record in the table describes one XML tag required in the JCSS XML. The tag name is identified in the ChildTag field. The ParentTag field specifies the name of the parent XML tag that includes the ChildTag. The remaining fields specify additional properties of the ChildTag, including the name of a Python method to compute a value if needed.

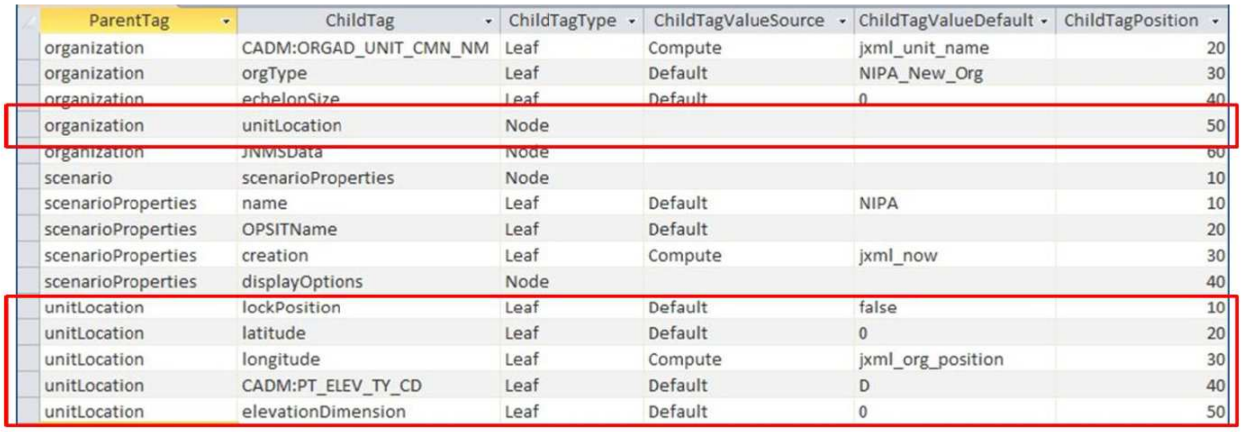
Table 1: Text fields describing JCSS XML structure.

Field Name	Description	Example
ParentTag	Name of an XML tag in the JCSS XML schema	scenarioProperties
ChildTag	Name of an XML sub-tag required for the ParentTag	name
ChildTagType	Type of this sub-tag, either "Node" or "Leaf." "Node" tags MUST have associated TagName records	Leaf
ChildTagValueSource	Method to determine value of this child tag or default	Default
ChildTagValueDefault	Default value of this child tag, or name of a Python method/recipe that can compute the value	NIPA
ChildTagPosition	Position of this child tag relative to its siblings	10

An example is highlighted in two (2) boxes in Figure 7. The record identified in the first box describes the "unitLocation" tag, as indicated in the ChildTag field. The ChildTagType is "Node", indicating that this tag is a node in the XML tag tree that has its own child tags. The second box identifies children of the "unitLocation" tag, i.e., those records that have a ParentTag field value of "unitLocation." These records all have a ChildTagType value of "Leaf", indicating that these tags have no additional children in the XML tag tree. "Leaf" tags either have a "Default" value, such as "0," or a "Compute" value, which refers to a Python method to be called within the Python script file.

Two Python scripts were built to perform the translation of SysML XMI to JCSS XML:

- jcss_xml_build.py
- jxml_callbacks.py



ParentTag	ChildTag	ChildTagType	ChildTagValueSource	ChildTagValueDefault	ChildTagPosition
organization	CADM:ORGAD_UNIT_CMN_NM	Leaf	Compute	jxml_unit_name	20
organization	orgType	Leaf	Default	NIPA_New_Org	30
organization	echelonSize	Leaf	Default	0	40
organization	unitLocation	Node			50
organization	JNMSData	Node			60
scenario	scenarioProperties	Node			10
scenarioProperties	name	Leaf	Default	NIPA	10
scenarioProperties	OPSITName	Leaf	Default		20
scenarioProperties	creation	Leaf	Compute	jxml_now	30
scenarioProperties	displayOptions	Node			40
unitLocation	lockPosition	Leaf	Default	false	10
unitLocation	latitude	Leaf	Default	0	20
unitLocation	longitude	Leaf	Compute	jxml_org_position	30
unitLocation	CADM:PT_ELEV_TY_CD	Leaf	Default	D	40
unitLocation	elevationDimension	Leaf	Default	0	50

Figure 7: Description of some example JCSS XML tags.

The main Python script, jcss_xml_build.py, reads and parses the XMI export from SysML and generates a JCSS XML file describing the organizations, *OpFac*s, devices, wired links, and IERs. The Python script builds the JCSS elements based on the stereotype instances found in the XMI and maps them to the appropriate JCSS XML tags. The second python script, jxml_callbacks.py, contains the library of routines developed to compute values for JCSS XML tags. Figure 8 shows the steps required to build the JCSS XML after an XMI stereotype instance is mapped to its corresponding JCSS representation.

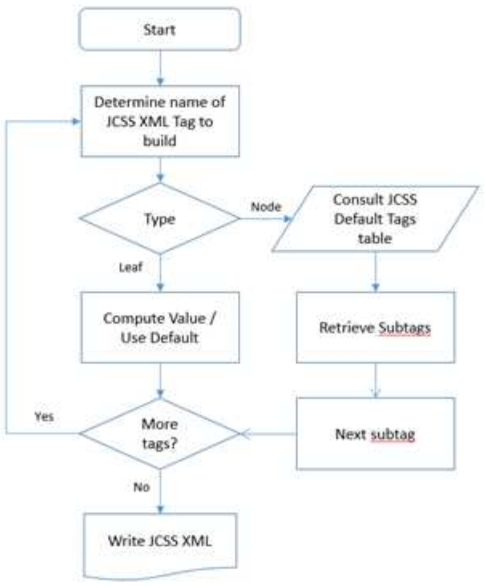

Start
Determine name of JCSS XML Tag to build
Type
Node
Consult JCSS Default Tags table
Leaf
Compute Value / Use Default
Retrieve Subtags
Yes
More tags?
Next subtag
No
Write JCSS XML

Figure 8: Parsing and transformation algorithm.

4 SUMMARY AND FUTURE WORK

4.1 Summary

There is a need to chain system models developed using varying formalisms, resolutions or scales. An ideal M4 environment would provide a singular (or small set of) modeling formalisms from which to derive representations (models) across formalism, resolution and scale. To support such an environment, we examined the application of a generic Systems Engineering (SE) modeling language's formalism as a common specification for discrete event simulations of networks. In this paper, we described an approach to enable the import of a SysML model to JCSS in order to quickly populate a JCSS model with OpFacs, network devices, links and other data. To complete this work, we plan to add more elements to the profile and test the translator and process with large realistic SysML models provided by DISA.

4.2 Future Work

While the JCSS profile developed in MagicDraw may be reusable in any SysML modeling tool, we recently learned that due to changing DoD modeling needs, the JCSS schema may be changing soon. In principle, the return on investment in devising unique tool solutions is low. In the short term, we may continue to develop this SysML to JCSS transformation profile and algorithm. The goal is to support DISA in the design and analysis of the Joint Information Environment (JIE 2015). The ability to import and export model information from one formalism (SysML) to another (JCSS) and vice versa supports two DISA needs:

1. An ability to reverse engineer an existing JCSS model to generate an equivalent SysML for the purposes of producing documentation/ or communication across a diverse group of stakeholders.
3. An ability to generate network models (in JCSS) from SysML models for the purpose of conducting further performance analysis and simulation.

The long term goal of this MITRE research is to produce a modeling environment that supports the broadest range of Model-based Systems Engineering activities.

ACKNOWLEDGMENTS

This work is supported through MITRE Independent Research and Development. The views expressed are those of the authors and do not reflect official positions of the MITRE Corporation or the U.S. Government. The authors thank Glenn Roberts and Ernie Page for their vision and enthusiastic support for the research.

REFERENCES

Batarseh, O, McGinnis, L.F., 2012. "System Modeling In SysML And System Analysis In Arena." In *Proceedings of the 2012 Winter Simulation Conference* 1 – 12.

FMI 2015, Functional Mock-up Interface, https://www.fmi-standard.org/

fUML 2013. *Semantics of a Foundational Subset for Executable UML Models (fUML)* v1.1, Accessed May 5. http://www.omg.org/spec/FUML/1.1/

ISO 10303-233:2012. Industrial Automation Systems and Integration -- Product Data Representation and Exchange -- Part 233: Application protocol: Systems engineering, Accessed May 5. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=55257

JCSS 2015. *Joint Communication Simulation System*, Accessed May 5. http://www.disa.mil/Mission-Support/Enterprise-Engineering/JCSS

JIE 2015. *Joint Information Environment,* Accessed May 5. http://www.disa.mil/about/our-work/jie

Kapos, G-D, Dalakas, V., Nikolaidou, M., Anagnostopoulos, D., 2014a. "An Integrated Framework for Automated Simulation of SysML Models using DEVS." *Simulation* 90 (6): 717 – 744.

Kapos, G-D; Dalakas, V., Tsadimas, A. ; Nikolaidou, M. and Anagnostopoulos, D., 2014b. "Model-based System Engineering using SysML: Deriving Executable Simulation Models with QVT." *Systems Conference (SysCon), 8th Annual IEEE* 531 – 538.

MBSE Wiki, 2015. Accessed May 5. http://www.omgwiki.org/MBSE/doku.php

McGinnis, L. Ustun, V., 2009. “A Simple Example of SysML-Driven Simulation.” In *Proceedings of the 2009 Winter Simulation Conference* 1703 – 1710.

MITRE 2012. Abdul-Rauf, S. Houser, D., Lakshminarayan, V., “Network Integration and Performance Analysis (NIPA) – Joint Communications Simulation System (JCSS) Interface - Assessment, Installation, Use, & Technical Reference, Version 0.1.” MTR 120612, The MITRE Corporation.

Mosterman, P. J., Vangheluwe, H., 2004. “Computer Automated Multi-paradigm Modeling: An Introduction.” *Simulation* 80: 433 – 450.

No Magic 2015. MagicDraw UML Modeling Tool, Accessed May 5. http://www.nomagic.com/products/magicdraw.html

OMG 2015. UML Profiles: Accessed May 5. http://www.omg.org/mda/specs.htm

Python 2015. Python Programming Language, Accessed May 5. https://www.python.org/

Riverbed 2015. OPNET Modeler, Accessed May 5. http://www.riverbed.com/products/performance-management-control/opnet.html

SDRP 2007. *PIM and PSM for Software Radio Components (SDRP)*, Accessed May 5. http://www.omg.org/spec/SDRP/

SysML 2014. *Systems Modeling Language*, Accessed May 5. http://omgsysml.org/index.htm

Wang, R., Dagli, CH 2008. “An Executable System Architecture Approach to Discrete Events System Modeling using SysML in Conjunction with Colored Petri Net,” In Proceedings of the *2nd Annual IEEE Systems Conference* 1-8.

Wang, W., Zhao, B., 2012. “Research on Construction Method of DoDAF View Based on DEVS and SysML,” *World Automation Congress* (WAC), 1 – 4.

Wang, Z., Hongyue, H. E., Wang, Q., 2014. “Executable Architecture Modeling and Simulation Based on fUML,” *19th ICCRTS*, http://www.dodccrp-test.org/s/2014-091-yfn5.pdf

XMI 2015. *XML Metadata Interchange*, Accessed May 5. http://www.omg.org/spec/XMI/

Zeigler, B.P., Praehofer, H., and Kim, T., 2000. *Theory of Modeling and Simulation*. Academic Press, 2nd edition.

AUTHOR BIOGRAPHIES

FATMA DANDASHI works at the MITRE Corporation where she applies knowledge and expertise in systems engineering, information technology, operational concepts, and enterprise modernization to address various US Government’s critical needs. In this role, Dr. Dandashi supports DoD on standards development for MBSE and architecture-based analysis modeling languages (UML, SysML, BPMN) and frameworks UPDM. Dr. Dandashi is an International Council on Systems Engineering (INCOSE) Certified Systems Engineering Professional (CSEP). She holds a Ph.D. in Information Technology from George Mason University. Her email address is dandashi@mitre.org

VINAY LAKSHMINARAYAN works at MITRE's Center for Connected Government, in their Systems Engineering Tech Center. He has supported several DoD initiatives, including the development of JCSS software distributed by DISA. His research interests include the application of Model Based Systems Engineering practices to network management. He received his Master of Science degree in Computer Science & Telecommunications from the University of Missouri – Kansas City. His email address is vinay@mitre.org.

NANCY SCHULT is a Division Staff within the Infrastructure and Networking Technical Center at MITRE. Her primary technical interests are in computer networks and analysis, with a focus on analyzing the performance of different networks and protocols, using analytic, modeling & simulation (M&S), and emulation techniques. She holds a Ph.D. in Computer Science from the University of Virginia. Her email address is nschult@mitre.org.