Supporting Coalition Battle Management Language Experiments with Scripted Web Services

Dr. J. Mark Pullen
Douglas Corner
Center of Excellence in C4I
George Mason University
4400 University Drive
Fairfax, VA 22030, USA
Voice: (1) 703-993-3682
Fax: (1) 703-993-1706

mpullen@c4i.gmu.edu / dcorner@c4i.gmu.edu

Dr. Kevin Heffner
Command & Control Group
Product & Technologies
CAE Inc.
Montreal, Canada
Voice : (1) 514-341-6780 ext. 4486
Fax: (1) 514-734-5698

kevin.heffner@cae.com

ABSTRACT

Recent developments in Command and Control (C2) to Modeling and Simulation (M&S) interoperability have shown excellent potential to enable a range of coalition C2 and M&S systems to interoperate for training, mission rehearsal, and decision support. Achieving these capabilities is one of the major coalition challenges for the next few years. The NATO Modeling & Simulation Group has sponsored a Technical Activity, MSG-048, which is exploring recent advances in Command and Control-Simulation interoperability. The technology opportunity lies in Battle Management Language (BML), which supports effective communication among the C2 and simulation systems involved. The result promises to enable significant advances in simulation support for mission rehearsal, training, and mission planning for military operations. MSG-048 is in the fourth and final year of an experimentation program, intended to identify and validate the applicability of BML in support of coalition activities. This paper reports on a technical development, which has proved useful in enabling the rapid development of net-centric infrastructure for these experiments. The concept of scripted languages, recognized to be of high value in computing systems, enables an approach whereby the underlying functions of the BML Web Service are implemented in a scripting engine. This in turn facilitates a concise representation of the necessary mapping between BML and its supporting data model, resulting in an ability to implement changes in BML and the data model rapidly and accurately. The paper describes how the scripted services have been implemented and used to increase the effectiveness of the MSG-048 experimentation program. The result was that new capabilities and changes to existing services, requested by the experimentation group, could be made available rapidly and had fewer problems in implementation. The scripted services supported a six-nation interoperability demonstration in 2008 and are on track to support a full week of BML experimentation in 2009.

1.0 INTRODUCTION

The capability for ready interoperation of Command and Control (C2) systems with Modeling & Simulation systems long has been recognized as an important enabler of coalition operations. Battle Management Language (BML) is being developed as a generic way to accomplish the needed interoperability in the network-centric environment [1-3]. Recent years have seen demonstrations of a highly promising capability, based on the Command and Control Lexical Grammar (C2LG) and the Joint Command, Control and Consultation Information Exchange Data Model (JC3IEDM) [4-11]. Of particular interest in the coalition environment are the Simulation Interoperability Standards Organization (SISO) Coalition Battle Management Language (C-BML) [12] and the NATO Modelling and Simulation Group Technical Activity 048 (MSG-048) [13-16]. SISO C-BML was chartered to develop an industry standard for BML in the coalition environment. MSG-048 was chartered to evaluate and demonstrate the effectiveness of BML in that same environment. The synergy between the two is evident and has evolved to a mode of operation where MSG-048 evaluates national BML technologies in experimental interoperation while SISO C-BML is able to adjust its standards development on their demonstrated effectiveness.

This paper provides an introduction to a new technical approach to implementing the middleware repository used in the MSG-048 experiments and proposed for the SISO C-BML standards, which has evolved from the approach used in JBML [4]. The Scripted BML Web Service (SBML) produces the same database from the same BML inputs and produces the same BML outputs, but the scripts it uses require less time and expense to develop and maintain than the approach used in JBML. In following sections we first describe the layered approach to BML being pursued in SISO C-BML; then describe the motivation and design of SBML; we end with a description of the way these services are being used in the ongoing work of MSG-048.

2.0 BML CONCEPT MAP

The BML provides for a digitized representation of military documents (e.g. orders, requests, reports, plans) for use by C2 and simulation systems, with the underlying goal of increasing the efficiency and empowering the warfighter in a net-centric environment where the automated processing of these documents will be key to achieving, maintaining and exploiting an information superiority.

Some of the likely benefits of BML-enabled information flow include more efficient and timely mission planning capabilities, enhanced decision support systems and more realistic, comprehensive mission rehearsal scenarios. Also, BML will facilitate the direct stimulation of C2 systems by simulations for the purposes of operator training.

Figure 1 presents a layered concept map that represents BML as a means of conveying military – documents, in the form of BML expressions, to and from the C2 and simulation systems. In an effort to clarify the BML “system boundaries,” the following paragraphs describe the relationships between the concepts that relate BML to the BML consumer/producers and the constituent information elements of the reference data model from which the BML expressions are composed.

2.1 Lexicon, Syntax and Semantics

The three layers presented in figure 1 can be mapped roughly to three linguistic concepts: lexicon, syntax and semantics. The first layer represents the lexicon upon which BML is based and contains the set of terminals or words. The second layer – the current focus of SISO C-BML – deals with syntax which allows for the creation of non-terminal symbols (aka BML expressions) based on production rules that dictate how the lexical elements may be combined [20]. The third layer focuses on semantics - the meaning of the BML expression as intended or interpreted by the BML producer/consumer - which greatly depends on the specific context, available knowledge and the knowledge representation.

As semantic structure cannot always easily be derived from syntactic structure, and in an effort to reduce the ambiguity of the interpretation of BML expressions, the Command & Control Lexical Grammar (C2LG) [20] proposes a formal grammar that includes a set of semantic labels (or thematic roles) and an unambiguous mapping between these labels and the constituent elements composing the expressions. Indeed, as shown in the right of the figure, the grammar represents the principal BML link between the upper *application layer* and the *reference data model*.

Achieving semantic interoperability between C2 systems and simulation systems ultimately is required in order to access the advanced capabilities described above. BML will provide the basis for semantic interoperability by allowing these systems to accurately exchange meaning. Ensuring syntactic interoperability - a prerequisite to semantic interoperability - implies virtually eliminating ambiguity at both the lexical and syntactical/structural levels. This is clearly one of the primary requirements of BML, but achieving the ultimate goal of semantic interoperability will likely require the development of *domain ontologies* to further ensure the accurate and unambiguous exchange of meaning.

2.2 BML Reference Data Model

BML expressions are composed of elements (most of which are) derived from the underlying reference data model the JC3IEDM, which contains a comprehensive set of data elements and business rules that allow for the detailed representation of military documents. However, for the purposes of BML, logical groupings of JC3IEDM data elements (also known as *business objects* or *transactionals*) can be created to facilitate the formation of BML expressions. Some examples of research in the area of JC3IEDM-based frameworks (see https://trac.fkie.fgan.de/JC3DEV) that facilitate the creation and processing of such transactionals include the OMG SOPES effort [21], the MIP-MDA IDA effort [22] and the ODU VMASC Model-Based Data Engineering [23]. Similarly, as describe in the following sections, SBML also deals with JC3IEDM transactionals.

For the purposes of discussion, the *representation* captures the sum of the information elements required to interpret the BML expression.

Although the JC3IEDM provides the vocabulary and data reference data model for BML, extensions to this model will likely be required - as indicated in figure 1. These extensions may include additional data elements, business rules and composite data elements. Such additional elements may or may not be accepted by the MIP as part of the JC3IEDM, but they will be required in systems that are to exchange BML expressions.

2.3 Battle Management Language and BML Usage

BML itself is comprised essentially of the reference data model extensions (e.g. to the JC3IEDM), the mappings of transactionals onto the JC3IEDM, and the business rules or production rules that constitute the grammar and allow for the creation and validation of BML expressions. Conceptually, the grammar specifies how the information elements may be accessed and interpreted in a meaningful manner by the BML producer/consumers.

Strictly speaking, doctrine is specific to a service or to joint, combined or coalition forces. Coalition BML or C-BML is intended to support coalition operations, but ultimately, in addition to NATO doctrine, must support specific doctrines from coalition countries. Therefore the BML grammar is required to construct expressions that comply with all of the supported doctrines; thus it must be based on common doctrinal constructs rather than be based on any one specific doctrine.

For the purposes of discussion, *representation*, *grammar* and *doctrine* are depicted as abstract entities, and they are mapped to information exchange constructs (e.g. schemas) as follows: doctrine is expressed as (application) policy, grammar is comprised of BML business rules (also known as *production rules*) and representation is constrained by the reference data model business rules. In reality the three are mutally dependent, in a way that is similar to the relationship among the three sides of the BML triangle from reference [24] (e.g. doctrine, protocols, representation).

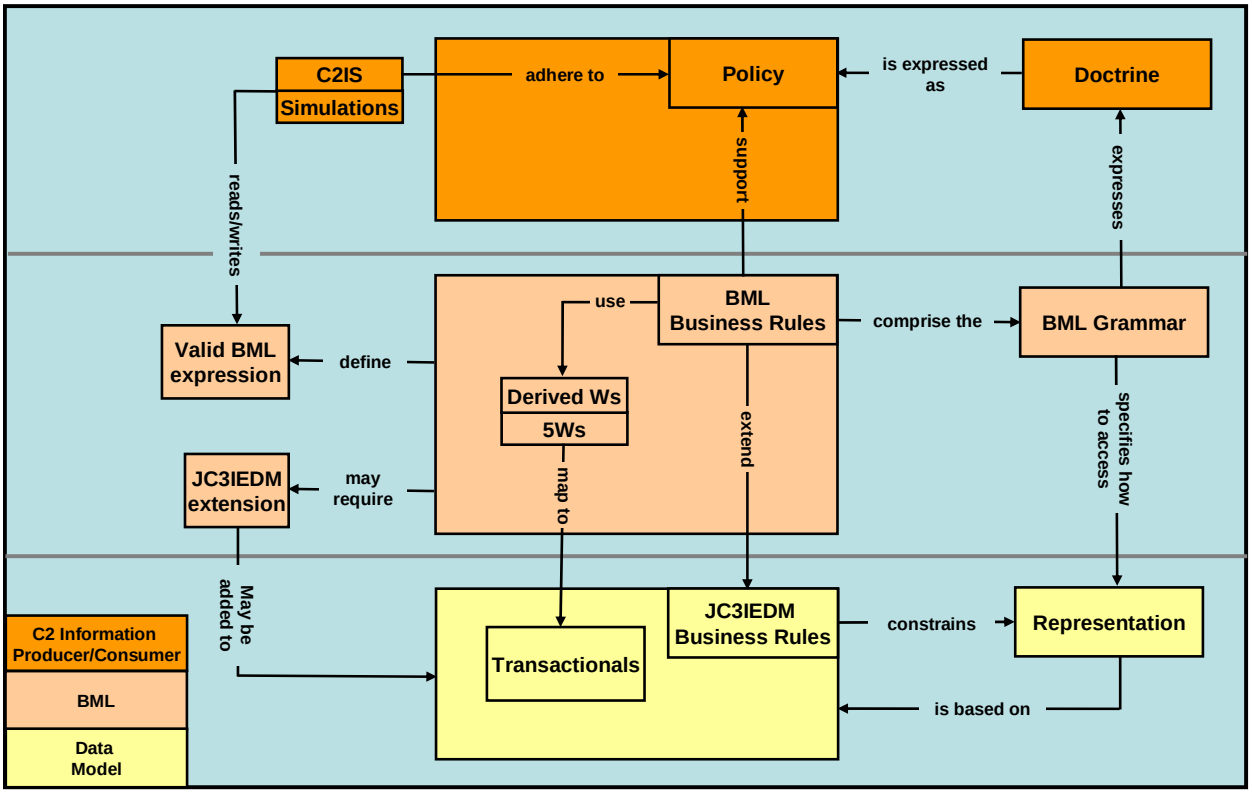

Figure 1: BML Concept Map

3.0 BML NETWORK-CENTRIC ARCHITECTURE

Figure 2 shows a general overview of the approach used in BML today, originally described in [3]. In principle, BML could be used for C2-C2 and Simulation-Simulation linkage. Also a capability for C2 to Robotic Systems has been considered, since the unambiguous nature of BML would fill a key requirement in that domain. However, work to date has focused on C2-Simulation, where there is a large and evident need that BML shows great promise of filling. The implementation of the BML middleware repository for orders and reports via Web services makes this architecture an excellent fit for the latest approach to Network-Centric systems, the Service Oriented Architecture (SOA). It also provides an excellent development environment, because various participating C2 and simulation software systems need not be operational simultaneously; each system can push and pull XML documents to and from the repository for testing. With an instance of the service available via Internet, this approach has been demonstrated by MSG-048 to greatly facilitate BML implementation by coalition teams working over a wide span of time zones and software technologies.

An additional strong synergy between BML and SOA is the issue of namespaces. SOA development of distributed systems proceeds quite rapidly, but can only occur when the namespaces for participating systems provide a solid basis for interfacing. BML is an ideal SOA application because the language it defines for orders and reports serves as a specification for that namespace.

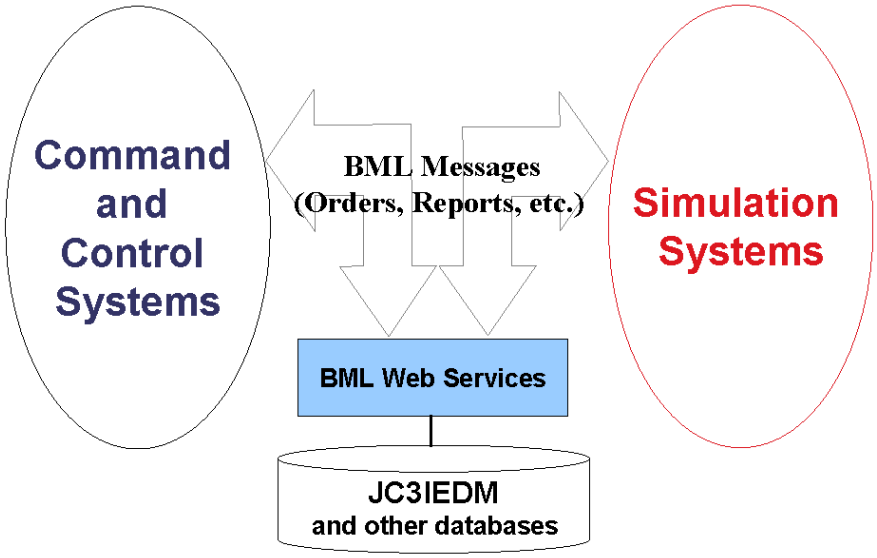

Figure 2: BML Top-Level Architecture

4.0 MOTIVATION FOR SCRIPTED SERVICES

The interpreted implementation of the BML WS, described in [18, 19], is shown in Figure 3. The essential difference between this and previous BML WS is that the server code no longer implements the

transformation mapping from BML to JC3IEDM directly in Java. Instead, the server consists of an interpreter, while the logic of the required transformation and database transactions is contained in an external script. The BML transactions to/from the client, the BML schema, and the server script all are coded in XML; however, they contain very different information.

- The BML transactions provide for Push and Pull of BML Orders and Reports. In effect, the BML WS is acting as a repository for this information.
- The BML schema defines all legitimate BML inputs.
- The server script describes how each possible BML transaction type will be carried out, in terms of its inputs, outputs, transformations, and get/put interactions with the JC3IEDM database.

Close inspection of Figure 3 will reveal that the service implements two different JC3IEM data storage paradigms. The one discussed below is a typical Structured Query Language (SQL) interface to the open source MySQL database server. The other interfaces to the SIMCI JC3IEDM Java Reference Implementation (RI) described in [6].

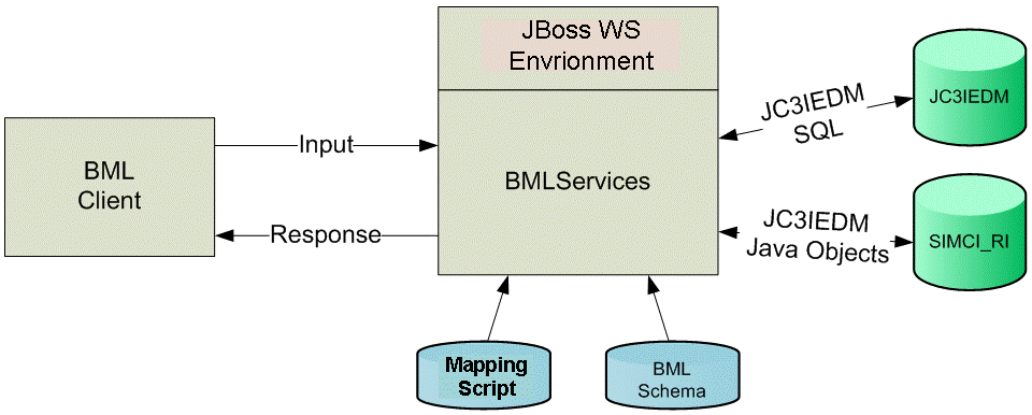

Figure 3: Interpreted BML Service

5.0 IMPLEMENTATION OF SCRIPTED SERVICES

BML as a language is rooted in the Command and Control Lexical Grammar (C2LG) described in [7, 11, 20]. For example, a Task takes the form:

*OB → Verb Tasker Taskee (Affected|Action) Where Start-When (End-When) Why Label (Mod)**

The values for Verb are taken from the JC3IEDM table "action-task-category-code". Tasker represents the unit or individual that assigns the task, and Taskee is the unit that has to execute it. Start-When and End-When express when the task has to start and has to be finished, respectively. End-When is optional as indicated by the brackets. Why denotes a purpose for the assignment. Label is a unique identifier for the task: it is used to refer to the task in other expressions. Mod (modifier) is a wild card.

5.1 BML Namespace

The XML tags associated with the C2LG elements have names derived from the traditional military description of Order information: "Who, What, When, Where, and Why." However, close inspection of the elements of the grammar will reveal that a finer distinction is necessary in order to produce an unambiguous task definition. For example, the BML Task defines three types of Who: *TaskerWho*, *TaskeeWho*, and *AffectedWho*, in order to clarify precisely the role of each in the task.

Figure 4 gives the BML schema for a Task, including the WhoType for *TaskeeWho*. (The latest posted schema can be found at http://netlab.gmu.edu/JBML.) In this figure, the influence of the C2LG is readily apparent. Figure 5 gives an example of the BML input, involving *TaskeeWho*. Figure 6 gives an example of the script which defines the mappings for TaskeeWho Business Object (BO) domain entity.

5.2 Scripted BML Server Design

The principal innovation in Scripted BML is the software architecture for an interpreted WS; the scripting language itself is not a significant innovation, but is a necessary part of the interpreted WS. We summarize it here to indicate its simple nature and scope, which are directly linked to the simplicity of the interpreter and also to the fact that a developer who knows the data model needs no expertise in Java programming. More detail regarding the scripting language is available in [18].

- The scripting engine works by stepping though the BML input file using the Document Object Model (DOM) parser. In due course, the entire BML file is processed.
- A BO defined in a script is invoked when its name is found at the root node in the BML input data file during the parsing process. It can then *call* other BOs as needed, to complete processing the BML input.
- The basic data element for the script is a character string. The interpreter maintains a table of *workingVariables* (WVs) which are associated with strings or aggregates of strings. The scope of these is local to the node where the BO is invoked, unless the WV is declared to be global.
- A *literalValue* is available for assignment and comparison to the value of a WV. The value associated with a *businessObjectTag* at the XML node being parsed also can be assigned or compared to the value of a WV.
- The supported aggregates are *Row*, a collection of named scalars associated with a row in a database table, and *List*, a multi-valued working variable, *i.e.* an array of strings. When a script is invoked on a *List*, each element is associated with an iterator (iteration index) value; all elements are processed in sequence.
- The script can *call* a sub-script, much like a C function or Java method, and a sub-script can have parameter WVs. When the *call* is performed at a point in the input file containing multiple instances of a BML tag, it is automatically repeated for each instance. A WV defined within a sub-script has scope local to the sub-script. The sub-script can *return* a result or *abort* if it detects an error.
- Processing elements in the script are *tableQuery*, which describes a database transaction; *call*; *conditional*, which supports an if-then-else construct based on relational operators; *assign*, which replicates the value associated with a WV; *abort*; and *BusinessObjectReturn*, which produces a string in the XML output produced by the interpreter before returning to the calling script.
- It is noteworthy that the scripting engine is very general in nature; it could be used to create an SQL-based repository for any XML-based domain language.

```
<xsd:complexType name="GroundTaskType">
  <xsd:sequence>
    <xsd:element name="TaskeeWho"
                 type="WhoType">
    </xsd:element>
    <xsd:element name="What"
                 type="WhatType">
    </xsd:element>
    <xsd:element name="Where"
                 type="WhereType">
    </xsd:element>
    <xsd:element name="StartWhen"
                 type="WhenType">
    </xsd:element>
    <xsd:element name="EndWhen"
                 type="WhenType"
                 minOccurs="0">
    </xsd:element>
    <xsd:element name="AffectedWho"
                 type="WhoType"
                 minOccurs="0">
    </xsd:element>
    <xsd:element>name="Why"
                 type="WhyType"
                 minOccurs="0">
    </xsd:element>
    <xsd:element name="TaskControlMeasures"

  type="TaskControlMeasuresType"
                 minOccurs="0">
    </xsd:element>
    <xsd:element name="Transport"
                 type="TransportType"
                 minOccurs="0">
    </xsd:element>
    <xsd:element name="Label"
                 type="LabelType"/>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
. . .

<xsd:complexType name="WhoType">
  <xsd:choice>
    <xsd:element name="Equipment"
                 type="EquipmentType"
                 maxOccurs="unbounded"/>
    <xsd:element name="OrgName"
                 type="xsd:string"/>
    <xsd:element ref="CompositeWho"/>
  </xsd:choice>
</xsd:complexType>
```

Figure 4: BML Schema for *TaskeeWho*

```
<!-- Fragment of <OrderPush> -->
  <Task>
    <GroundTask>
      <TaskeeWho>
        <UnitID>UIE9 FA</UnitID>
      </TaskeeWho>
      . . .
```

Figure 5: BML Input for *TaskeeWho*

```
<!-- Fragment of code from WhatWhenPush -->
<call>
  <boName>TaskeeWhoPush</boName>
  <anchorTag>TaskeeWho</anchorTag>
  <parameter>
    <workingVariable>task_act_id
    </workingVariable>
  </parameter>
</call>
. . .
<BusinessObjectTransaction>
    <transactionName>TaskeeWhoPush
    </transactionName>
    <parameter>task_act_id</parameter>
    <tableQuery>
      <!—    implements:    0    GET    unit
        formal_abbrd_name_txt   =   TaskerWho
        result <- unit_id -->
      <databaseTable>unit</databaseTable>
      <queryAction>GET</queryAction>
      <resultName>unit_id</resultName>
      <columnReference>
        <columnName>formal_abbrd_name_txt
        </columnName>
        <businessObjectTag>UnitID
        </businessObjectTag>
      </columnReference>
    </tableQuery>
    . . .
```

Figure 6: Interpreter Script for *TaskeeWho*

6.0 MSG-048 EXPERIMENTS USING SCRIPTED SERVICES

Beginning in 2006, MSG-048 has presented progressively more complex and meaningful experimental demonstrations yearly. In 2006, a basic proof of principle was conducted by the US and France. In 2007, C2 and simulation software from six nations was integrated for a demonstration of Orders only [15]. By 2008, MSG-048 had progressed to an ability to exchange both Orders and Reports in BML, allowing C2 systems to provide orders for the simulations and show the results of those orders on the C2 systems' own screens, with no human intermediary, as shown in Figure 7. Moreover, the 2008 configuration included air operations and enabled various national systems to interoperate on a "mix and match" basis, including multiple combinations, as described further in [19]. All of this was supported by the Scripted BML Web Services:

- Netherlands ISIS C2 plus Germany C2LG GUI with Netherlands POLLUX simulation
- Norway NORTaC C2 with France SCIPIO simulation
- NATO ICC air C2 plus Germany C2LG GUI with US JSAF simulation (both operated by UK)
- Netherlands ISIS C2 plus Germany C2LG GUI with France SCIPIO simulation
- Norway NORTaC C2 with Netherlands POLLUX simulation

The charter for MSG-048 expires in 2009; this final year's goal is to build a set of user experiments around an expanded set of software that will include, in addition to the 2009 systems, a simulation from Spain (SIMBAD), a C2 system from the USA (MCS), and a simulation from the USA (OneSAF), and a C2 system (BattleView) and UAV simulation from Canada. All of these components will be supported by SBML, operating with a distributed JC3IEDM database as shown in Figure 8. Two sets of military SMEs will be engaged in the experimentation; in addition to the supporting SMEs who have worked with MSG-048 to create the capability, there will be a new, unbiased group of SMEs to evaluate the utility of BML in the coalition context, thus fulfilling MSG-048's charter to validate the utility of BML for NATO.

7.0 CONCLUSIONS AND FUTURE WORK

Work on the Scripted BML WS continues. The scripting language has been refined, with the result that the current generation of scripts needs about half as many lines of code as the one described in [17]. Our current effort is focused on adding a publish/subscribe capability. This is needed to avoid the polling style of operation that has been required for our BML WS to date. In order to do this we will change from the Axis WS environment to JBoss, a different open source support platform that implements publish/subscribe via the Java Messaging Service (JMS). This approach is used in the SIMCI RI described in [6]. The scripted BML server will be posted as open source software on website http://netlab.gmu.edu/SBML as it becomes available in stable form.

Based on results described above, we conclude that scripted operation, inplemented in a Web service, provides a powerful infrastructure for the continuing development of BML. A further value of the scripting language is the ability to define precisely a mapping from BML to JC3IEDM. SBML represents a powerful means by which to test and experiment within the SISO C-BML as the standard evolves. Its highly concise, XML-formatted scripting offers a paradigm for an executable specification of the BML-to-JC3IEDM mappings that can result in a clear standard that is readily implementable.

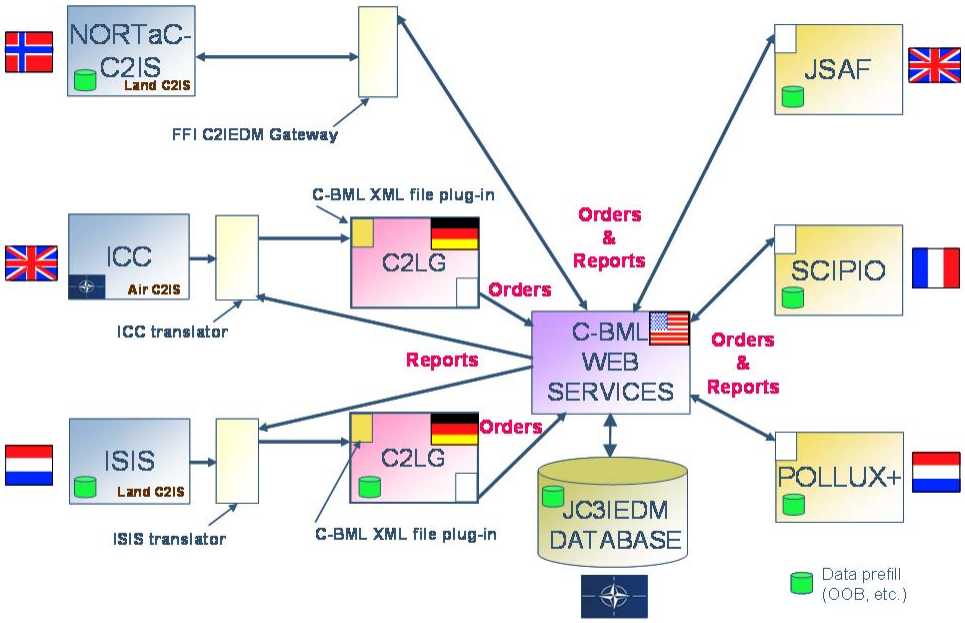

Figure 7: MSG-048 Demonstration Configuration 2008

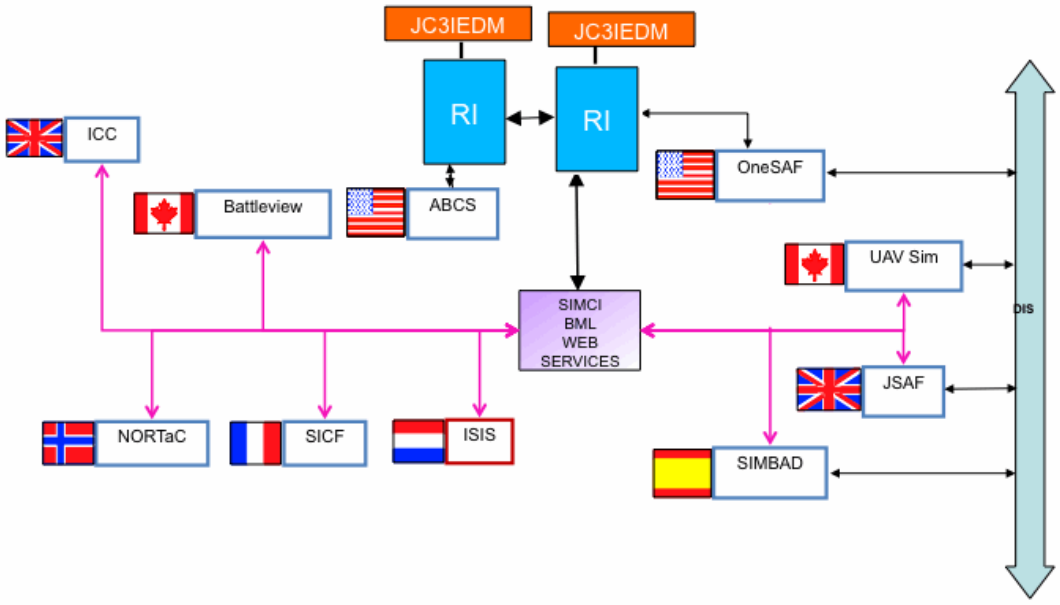

Figure 8: MSG-048 2009 Experimentation Configuration

8.0 REFERENCES

[1] Sudnikovich, W., Pullen, J. Kleiner, M. & Carey, S. (2004) *Extensible Battle Management Language as a Transformation Enabler,* SIMULATION, Vol. 80, pp. 669-680

[2] Morse, K., Brunton, R., Pullen, J., McAndrews, P. Tolk, A. & Muguira, J. (2004) *An Architecture for Web Services Based Interest Management in Real Time Distributed Simulation,* 8th IEEE International Symposium on Distributed Simulation and Real-Time Applications, Budapest, Hungary.

[3] Sudnikovich, W., Ritchie, A., Hieb, M., Pullen, J., de Champs, P. & Khimeche, L. (2006) *NATO Exploratory Team – 016 Integration Lessons Learned for C2IEDM and C-BML*, IEEE 2006 Spring Simulation Interoperability Workshop, San Diego, CA, USA.

[4] Levine, S., Pullen, J., Hieb, M., Pandolfo, C., Blais, C., Roberts, J. & Kearly, J. (2007) *Joint Battle Management Language (JBML) Phase 1 Development and Demonstration Results*, IEEE Fall Simulation Interoperability Workshop, Orlando, FL, USA.

[5] Pullen, J., Makineni, K. & McAndrews, P. (2007) *A Grammar-Based Web Service Enabling Multi-Domain Distributed Interoperation of Command and Control and Simulation Systems*, 11th IEEE International Symposium on Distributed Simulation and Real-Time Applications, Chania, Greece.

[6] Levine, S., Topor, L., Troccola, T. & Pullen, J. (2008) *A Practical Example of the Integration of Simulations, Battle Command, and Modern Technology*, IEEE 2008 Fall Simulation Interoperability Workshop, Orlando, FL, USA.

[7] Schade, U. & Hieb, M. (2006). *Development of Formal Grammars to Support Coalition Command and Control: A Battle Management Language for Orders, Requests, and Reports*, 11th International Command and Control Research and Technology Symposium, Cambridge, UK.

[8] Bernard, F., Khimeche, L., Pullen, J, Hieb, M., & Powers, M. (2006). *Battle Management Language Transformations*. NATO 2006 Modelling & Simulation Symposium, Rome, Italy.

[9] Hügelmeyer, P., Schade, U. & Zöller, T. (2007). *Application of BML to inter-agent communication in the ITSimBw simulation environment*. 2007 Winter Simulation Conference, Washington, DC, USA.

[10] Gustavsson, P., Hieb, M., Groenkvist, M., Kamath, V., Blomberg, J. Wemmergard, J. (2008). *BLACK-CACTUS – Towards an Agile Joint/Coalition Embedded C2 Training Environment*. IEEE 2008 Spring Simulation Interoperability Workshop, Providence, RI, USA.

[11] Schade, U. & Hieb, M. (2007). *A Linguistics Basis for Multi-Agency Coordination*. 12th International Command and Control Research and Technology Symposium, Newport, RI, USA.

[12] Blais, C., Galvin, K. & Hieb, M. (2005) *Coalition Battle Management Language (C-BML) Study Group Report*, IEEE 2005 Fall Simulation Interoperability Workshop, Orlando FL, USA.

[13] Pullen, J., Hieb, M., Khimeche, L., Powers, M. & Galvin, K. (2007) *Evaluating the Proposed Coalition Battle Management Language Standard as a Basis for Enhanced C2 to M&S Interoperability*, NATO 2007 Modelling and Simulation Symposium, Prague, Czech Republic.

[14] de Reus, N., de Krom, P., Mevassvik, O., Alstad, A., Schade, U. & Frey, M. (2008) *BML-enabling national C2 systems for coupling to Simulation*. IEEE 2008 Spring Simulation Interoperability Workshop, Newport, RI, USA.

[15] Pullen, J. *et al.* (2008). *NATO MSG-048 Coalition Battle Management Initial Demonstration Lessons Learned and Follow-on Plans*. IEEE 2008 European Simulation Interoperability Workshop, Edinburgh, UK.

[16] Pullen, J., Hieb, M., Schade, U., Kruger, K., Frey, M. & Orichel T. (2008). Enabling the MSG-048 Multinational Demonstration 2007 with the Command and Control Lexical Grammar and JBML Web Services. NATO 2008 Modelling & Simulation Symposium, Vancouver, CA.

[17] Pullen, J., Corner, D. & Singapogu, S. (2009) *Scripted Battle Management Language Web Service Version 1.0 Operation and Mapping Description Language*. IEEE 2009 Spring Simulation Interoperability Workshop, San Diego, CA, USA.

[18] Pullen, J., Corner, D., Singapogu, S. & McAndrews, P. (2009). *Interpreted Web Services as a Tool for Development of Command and Control-Simulation Interoperability*. 13th IEEE International Symposium on Distributed Simulation and Real-Time Applications, Singapore.

[19] Pullen, J. *et al.*(2009). *Adding Reports to Coalition Battle Management Language for NATO MSG-048*. IEEE 2009 European Simulation Interoperability Workshop, Istanbul, Turkey.

[20] Schade, U., Hieb, M. & Frey M. (2009) Command and Control Lexical Grammar (C2LG) Specification. *Draft*

[21] Shared Operational Picture Exchange Services (SOPES) (2009) *Information Exchange Data Model (IEDM) Specification, Object Management Group Ver 0.96*

[22] Loaiza, F., Wartik, S. (2008) Normative Interaction Specifications for C2: A Comprehensive Type of Rule - Models for Use in the Model Driven Architecture Framework, Institute for Defense Analyses

[23] Tolk, A., Diallo, S., (2005) *Model-Based Data Engineering for Web Services*, IEEE Internet Computing, vol. 9, no. 4, pp. 65-70

[24] Tolk, A., Blais, C., (2005) *Taxonomies, Ontologies, and Battle Management Languages – Recommendations for the C_BML Study Group*, IEEE 2005 Simulation Interoperability Workshop