Publish/Subscribe and Translation Performance in the WISE-SBML Server for MSDL and C-BML

Dr. J. Mark Pullen
Douglas Corner
C4I Center
George Mason University
Fairfax, VA 22030, USA
+1 703 993 3682
mpullen@c4i.gmu.edu
dcorner@c4i.gmu.edu

Magnus Grönkvist
Saab AB
Järfälla, Sweden
+46 734 375 277
magnus.gronkvist@saabgroup.com

Keywords:
Schema Translation, Performance, Command and Control - Simulation Interoperability

The Coalition Battle Management Language (C-BML) as currently implemented requires that interoperating Command and Control (C2) and Simulation systems implement an interface to a Web service, using a clearly defined schema, to propagate Orders and Reports to participating systems. GMU and Saab have been collaborating on a robust parsing publish/subscribe server that can translate among semantically-equivalent schemata. This capability was used for the NATO MSG-085 final demonstration and served well, enabling interoperation of clients that were built for three different schemata (the server is capable of a fourth, C-BML "full", but no client had been programmed to use it). This paper reports on our experience in the MSG-085 final demonstration and our work since then to improve the performance/throughput of both the translation and publish/subscribe functions of the server.

1. Overview

Battle Management Language (BML) and its various proposed extensions are intended to facilitate interoperation among command and control (C2) and simulation systems (C2SIM) by providing a common, agreed-to format for the exchange of information such as orders and reports. In recent implementation, this format has been accomplished by providing a repository service that the participating systems can use to post and retrieve messages expressed in BML. The service is implemented as middleware that is essential to the operation of BML and can be either centralized or distributed. Recent implementations have focused on use of Extensible Markup Language (XML) along with Web service (WS) technology, a choice that is consistent with the Network Centric Operations strategy currently being adopted by the US Department of Defense and its coalition allies [1].

SISO has a two-part standards effort supporting BML. The Military Scenario Definition Language (MSDL) standard [2] was approved in 2008. It is intended to reduce scenario development time and cost by enabling creation of a separable simulation independent military scenario format, focusing on real-world military scenario aspects, using the industry standard data model definition eXtensible Markup Language (XML) as input to initialize C2 and simulation systems. The Coalition BML (C-BML) standard [3] provides the tasking and reporting aspects of C2-Sim. It was balloted in 2012 and approved as a SISO standard in 2014. The approach has generally followed the Lexical Grammar approach introduced by Schade and Hieb [4,5]. Several recent studies and implementations have addressed effective combination of MSDL and C-BML [6-9]. Informing the standardization process have been multiple projects under various US DoD sponsors and an ongoing sequence of experimental BML configurations that were developed and demonstrated by the members of NATO MSG-048 and MSG-085 [10-16]. The experience gained in these activities has proved critical to shaping the MSDL and C-BML standards and implementing infrastructure, such as the translation service described in this paper, and also in demonstrating the potential applicability of BML.

References [15] and [16] describe the C2-Sim environment developed for NATO Technical Activity MSG-048, *Coalition Battle Management Language*. This activity included six national C2 systems and 5 national simulations, a scale of interoperation not previously attempted. Reference [17] describes developmental work for NATO Technical Activity MSG-085, *Standardization for C2-Simulation Interoperation,* leading to an experimental operational environment where multiple national C2 and Simulation systems can interoperate using MSDL and C-BML (see Figure 1). This paper provides a description of results obtained in performance tuning of the WISE-SBML server, using the Widely Integrated Systems Environment (WISE) made available by Saab AB to MSG-085 through the GMU C4I Center as an extension of the Scripted BML Server (SBMLServer) [17-23].

Continuing progress of C-BML, both in NATO MSG-085 and in the SISO standards process, has produced a situation where various clients have implemented different generations of schemata, which, while semantically equivalent or nearly so, cannot interoperate directly because of schema mismatch. Retrofitting these clients would be a drain on scarce resources. In our work to implement a robust C-BML/MSDL server using the WISE platform, we have incorporated the ability, first enabled in the Scripted BML Server (SBML), to translate among semantically equivalent schemata. The resulting capability was used by NATO MSG-085 to enable interoperation of all participating groups in the MSG-085 final demonstration. This paper explains the concept schema translation works and also issues encountered in its implementation, which includes an MSDL aggregation server.

Translating among schemata necessarily involves completely parsing each document input to the server and then publishing an output for each schema that is used by some participant. This great increase in workload places performance of software and hardware at a premium. The remainder of this paper provides a condensed history of SBMLServer and Saab's WISE system, a summary of how WISE is used to enable an operationally-focused server (WISE-SBML), and a discussion of how we tested and tuned the performance of WISE-SBML. Parts of this paper are extracted from [23].

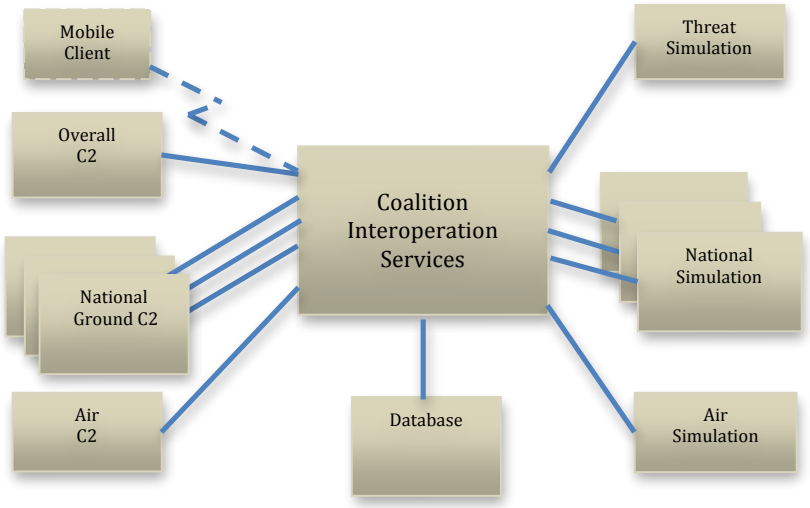

Figure 1: C2SIM Coalition Services Architecture

2. Background: Scripted BML Server

The George Mason University C4I Center, under management of US Army PM OneSAF and in close cooperation with MITRE and QinetiQ personnel, has developed a set of services that provide infrastructure to support implementation of MSDL/C-BML in MSG-085 C2 and simulation systems. The top-level architecture of a C2-simulation coalition using these services is shown in Figure 1. These implementations are available at http://c4i.gmu.edu/OpenBML as open source software.

Experience to date in development of BML indicates that the language will continue to grow and change. This is likely to be true of both the BML itself and of the underlying data representation used to implement the scripted server capability. However, it also has become clear that some aspects of BML middleware are likely to remain the same for a considerable time: namely, the XML input structure and the need for a repository server to store a representation of BML in a well-structured database. Initially, this implied to us an opportunity for a re-usable system component: a scripted server that can convert between a relational database and XML documents based on a set of mapping files and XML Schema files. The scripted server introduced in [17] and now named "SBMLServer," accepts *push* and *pull* transactions (BML/MSDL XML documents) and processes them according to a script (or mapping file, also written in XML). While the scripted approach may have lower performance when compared to hard-coded implementations, it has several advantages:

- new BML constructs can be implemented and tested rapidly
- changes to the data model that underlies the database can be implemented and tested rapidly
- the ability to change the service rapidly reduces cost and facilitates prototyping

- the script provides a concise definition of BML-to-data model mappings that facilitates review and interchange needed for collaboration and standardization

Since its initial use in NATO MSG-048 [18], SBMLServer has been enhanced considerably by:

- Supporting XML namespaces: XML tagnames can be qualified by addition of a "namespace" prefix. This allows tagnames from different sources to work together safely.
- Schema validation: the server is designed with an option to confirm that each document received conforms to the schema, which identifies a likely source of incompatibilities, at the cost of slowing the service.
- Logging/replay: the server writes a file showing every transaction it receives, with time stamps. The server is then capable of replaying this file to recreate the original sequence of Orders and Reports at original time intervals.
- Multithreading for performance: server throughput can be improved by processing messages in parallel.
- RESTful Web service interface: initially, SBMLServer supported only the traditional Web service protocol SOAP, which is intended to support remote procedure calls. However, the need in BML is for transfer of documents, which can be achieved more efficiently via Representational State Transfer (REST) protocol, reducing overhead and facilitating C++ implementation. MSG-085 adopted the RESTful approach for its final demonstration.
- Aggregating MSDL inputs: When multiple systems participate in a coalition, it is necessary to merge their MSDL scenario files. Some parts of the merge process consist simply of concatenation, but other parts require functions such as the largest of a group or the total count. The MSDL scenario is the element that binds together the components to be used for a particular exercise.
- Schema translation: because SBMLServer parses the BML input documents and stores their XML elements in a runtime database, it is possible to generate a version of the document translated to comply with a different, semantically similar schema.

3. SBML in the WISE Integration Environment

Saab Corporation is in the business of providing software for military command and control. They have been active in the Swedish delegation to NATO MSG-085 and have offered use of their Widely Integrated Systems Environment (WISE) for experimentation support. In 2012, discussions between the GMU C4I Center and Saab concluded that the general approach used in SBML could be productively re-implemented in WISE. WISE supports a robust, high-performance information switching capability with a graphic setup editor that provides and improves upon the advantages associated with the scripted approach of SBMLserver. This capability enables fundamental research at GMU, prototyping a new generation server that is expected ultimately to transition to operational military use as described in [20].

Figure 2 shows the architecture of the WISE-SBML server. The "BMS" system in Figure 2 represents the 9LandBMS or other WISE-interfaced C2 system. WISE appears to SBML as an in-memory, non-persistent database. This approach enables a great improvement in performance over the previous SBMLServer and is suitable for deployment in the high-performance cloud computing environment.

Integrating a new capability into WISE requires creating a software interface element and then using the WISE graphic editor to configure information mappings among various internal databases. These configuration elements must be maintained as changes to the schema occur. It is noteworthy that the second step in particular can be achieved more quickly than developing an SBML Version 2 script. It also is noteworthy that the WISE architecture is well suited to operation in a cloud.

The WISE-based Web service accepts XML inputs through a REST interface and publishes one or more XML documents (the original plus translations) through a STOMP interface. Therefore, to build a server based on WISE, the GMU team had to complete two important steps:

- Build a WISE driver, shown in green on the figure, for each major information flow to be interfaced: C-BML/MSDL Web service (one for each schema version); publish-subscribe service; persistent recording interface; and the 9LandBMS WISE interface, adapted for C-BML/MSDL.
- Use the WISE graphic editor to specify information flows within the WISE data repository.

The second of these steps represents one of the powers of the WISE approach and requires only drag-and-drop in the WISE Connectivity Designer. However, a requirement remains to implement WISE driver to be programmed in conjunction with the specifics of the application (in the case at hand, C- BML and MSDL) for XML documents, both incoming from the REST interface and outgoing to the STOMP interface for publication. The driver generates an XML document consistent with the schema under which the output is being published.

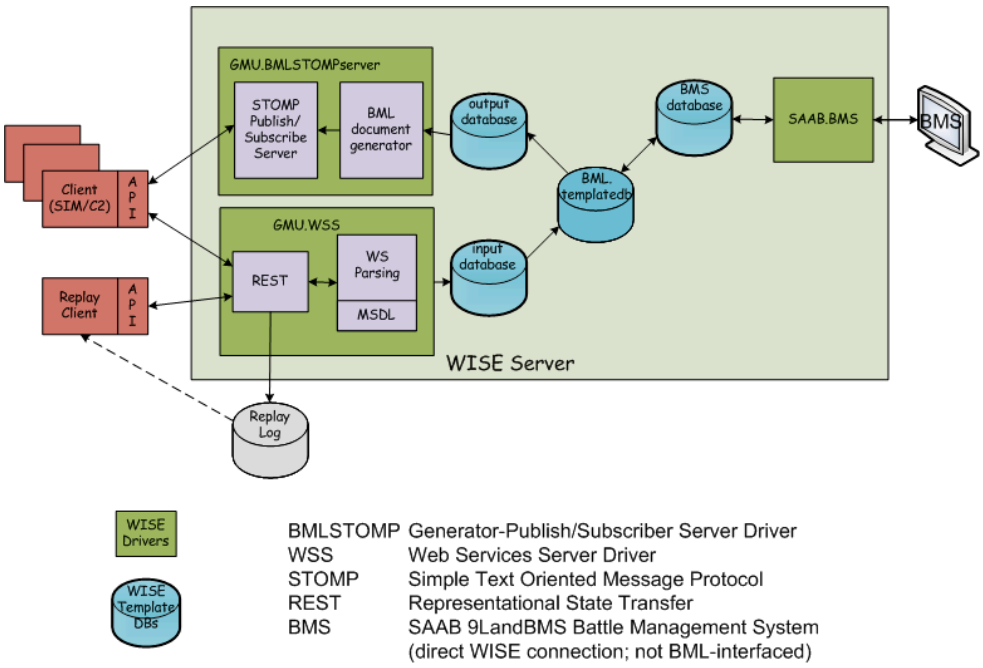

Figure 2. WISE/SBML Architecture

This takes place as follows, as shown in figure 3:

Accepting incoming XML from REST:

Parsing: using the open source DOM parser, the interface extracts each data element from the input XML file to an internal data structure.

Building: the internal data structure is pushed into the WISE database.

Producing outgoing XML through STOMP:

Receiving: a matching internal data structure is extracted from the database.

Generating:XML output is generated in accordance with the appropriate schema, using the internal data structure.

The WISE driver software generated for this purpose, written in C++, is available as open source at http://c4i.gmu.edu/OpenBML. Also available are the client software and replay client. WISE-SBML preserves all features of SBMLServer with the exception of the obsolete SOAP interface. It does not feature a persistent database; however, the trace of all inputs is captured in the replay log and can be used to restore the internal database to any desired checkpoint for restart. References [21] and [22] provide more information on WISE-SBML, including its use in a distributed server system for MSG-085.

4. Performance Tuning

At the MSG-085 Final Demonstration in December 2013 it was verified that the server had the intended functionality. Some performance and stability issues were however encountered. In the time up until the next demonstration at ITEC in May 2014 these issues were resolved.

In brief, performance issues were centered on:

- Memory management: small memory leaks are a common bug when using C++ incorrectly. As the server over time processes a large amount of messages and does many operations on strings (parsing and generating XML), these small memory leaks will add up over time. Because the test-cases during development use small numbers of messages to test functionality, such memory-leaks can go undetected for a long time. It is advisable to test any server-grade software from the beginning with large amounts of data to catch the errors early on. Also, choosing where to allocate memory in very frequently used functions, i.e. either on the stack or the heap, will have great impact on performance. There is however no silver bullet here that will work optimally for all cases, especially since today's compilers automatically optimize the code they produce. In addition, processors with different architectures and virtual machines execute the code differently. Testing performance for different solutions with large amounts of data is recommended.

- String concatenations: the string-class in the C++ Standard Template Library offers many ways to concatenate strings together, which is done a lot when generating XML documents. The different ways to concatenate strings affect performance greatly, mainly due to how memory is managed during the operation. If one knows roughly how large the total output XML document will be, good performance can be achieved by declaring a string on the stack, then calling reserve() on it giving the estimated total size and then use append() on it to concatenate the other strings. Using nested calls of the + operator should be avoided (ie string1 += string2 + string3) as performance decreases the more nested calls are used. It is better to use string1.append(string2).append(string3). Some instances of this problem remain in the server used for tests reported below.
- For clients that consume data at different rates or in bursts: the output from the server supports multiple connected clients, and it is important to handle situations where clients might consume data at different rates. Some clients might do internal processing on the data from the server that will cause them to be unable to keep up with reading what the server sends. This must not be allowed to cause issues for the server but, depending on the application, other faster-reading clients should still be able to receive data as fast as possible. This makes it necessary for the server to keep separate message output-queues for all connected clients and separate threads for each client that handle the sending of data.

In conclusion, with regard to performance and stability: it is important to test from the beginning with large amounts of data to catch errors early on. Also, be sure to optimize performance where it is needed most by finding out where the bottlenecks are; these are not always obvious and might differ among different platforms.

Planned future enhancements include XML parsing. The server now uses the Xerces XML parser's DOM-mode. This means that the entire XML-document is parsed and its data stored in memory, and afterwards one queries for the attribute-values one is interested in. This is perhaps not the ideal way for the BML-server to operate, as it is always interested in every attribute in the received messages. An approach using a SAX-parser could be more efficient in this case. Such a parser goes through the XML-document once from start to finish and outputs attribute values as it parses. This would remove the store to memory and query step now needed with the DOM-parser.

5. Performance Measurements

As shown in Figure 1, the server sits in the center of this environment and forwards orders from C2 system to simulation systems and reports from the simulated environment back to the C2 system for display. In order for the overall system to present a realistic picture of the simulated battle in as close to real time as is possible it is important that the server forward messages (BML documents) as quickly as possible. The WISE-SBML server performs translations among BML dialects, which adds to the processing load in the server.

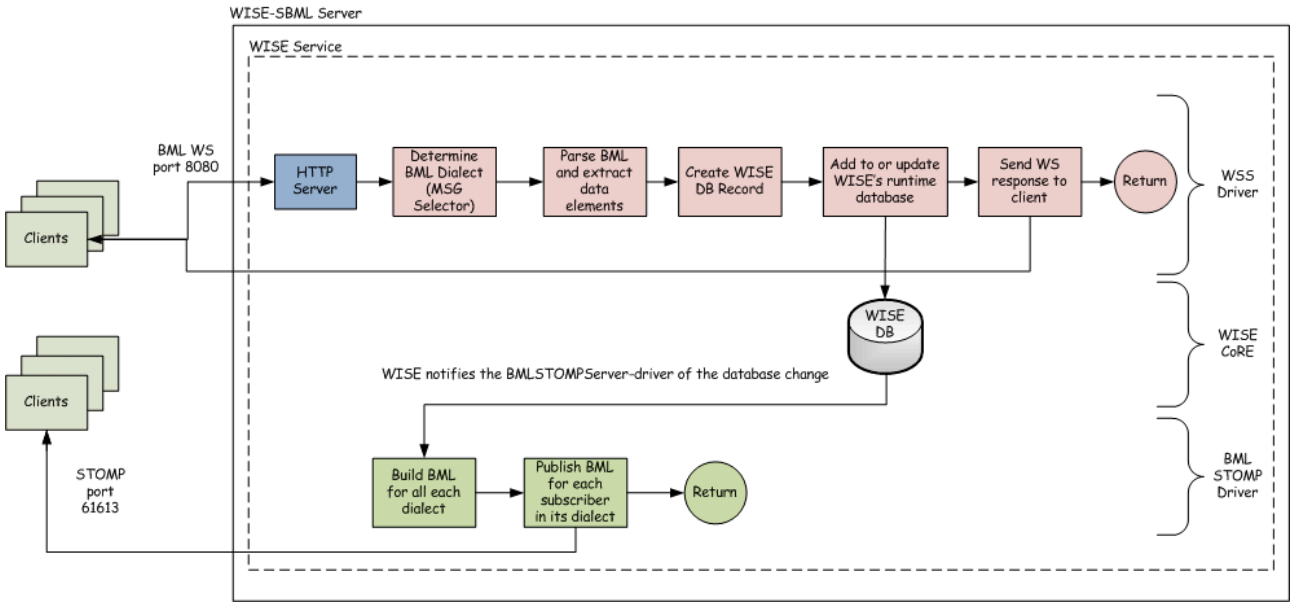

Figure 3. Data Flow in WISE-SBML Server

We have evaluated the throughput of the WISE-SBML server for some representative numbers of clients. The basic technique for this evaluation was to measure the time to process a single BML document and return a completion to the submitter. Multiple transactions were then submitted sequentially, at rate limited only by the server's throughput, and the responses averaged. The computing environment used was that of the VPN used for MSG-085 testing. The Web Service client and STOMP client were run on separate Linux virtual machines allocated 2GB memory with 1 processor core each and the WISE-SBML server was run on Windows7 VMWare virtual machine allocated 3GB memory with 2 processor cores. All of these virtual machines were run under VMWare VSphere on an Intel Xeon E5530 2.4 GHz.

There are two significant components to overall delay in the server: the input Web Service (including parsing the input document), and the output STOMP publication (including formatting the output document). On the input side, BML transactions are submitted to the server via a RESTFUL Web Service (WS) interface. This is a stateless interface in that each transaction proceeds as follows:

1. The client requests a TCP connection with the server (on port 8080) and records the start time.
2. The server accepts the connection.
3. The client submits the BML document
4. The server parses the BML document and returns an "OK" response or error message to the client.
5. The client closes the TCP connection and records the end time.

The elapsed time from open to close of connection provides an accurate measurement of the processing time as observed by the submitter. Table 1 shows measured throughput in our test configuration (where each translated output is counted as a transaction); in every case throughput is over 150 transactions per second. Tests where performed with up to 4 STOMP clients, each of which was configured for a different schema in order to test with full translation workload.

Table 1. Measured Throughput of WISE-SBML

Test Number	Configuration (WSC: web service client; SC: STOMP client)	Average Processing Time ms	Standard Deviation Processing Time ms	Total Server Throughput Rate tps
1	1 WSC No SC	6.6	1.6	153
2	4 WSC No SC	18.2	5.8	220
3	1 WSC 1 SC	6.5	1.8	154
4	4 WSC 1 SC	19.4	7.9	206
5	1 WSC 4 SC	6.6	1.7	153
6	4 WSC 4 SC	21.6	10.0	185

On the output side, the server generates one XML document per input transaction for each BML dialect configured, which complies with the schema for that dialect. Each dialect is then published to a separate subscription channel. To measure output performance, we instrumented a STOMP client to collect throughput in transactions per second. We confirmed that the server publishes STOMP messages at the same rate as they are introduced on the WS interface.

6. Conclusions

Work in C2-simulation interoperation, using emerging SISO standards MSDL and C-BML, continues to make progress as described in this and companion papers. Practical implementation by MSG-085 team members is leading to understanding of how military operations can be supported effectively by this technology. National implementations in both C2 and simulation systems, coupled with supporting open source server software, make the feasibility of this approach clear. However, a natural aspect of this flowering development is the existence of multiple, related schemata among the various C2-Sim clients. Availability of translating services supported by a robust, high-performance platform can serve as an enabler to gain critical experience through expanded application and interoperation of various national systems, as exemplified in MSG-085.

References

[1] Sudnikovich, W., J. Pullen, M. Kleiner, and S. Carey, "Extensible Battle Management Language as a Transformation Enabler," in *SIMULATION,* 80:669-680, 2004

[2] Simulation Interoperability Standards Organization, *Standard for: Military Scenario Definition Language (MSDL)* http://www.sisostds.org/DigitalLibrary.aspx?Command=Core_Download&EntryId=30830

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

[4] Schade, U. and Hieb, M., "Formalizing Battle Management Language: A Grammar for Specifying Orders," *2006 Spring Simulation Interoperability Workshop*, IEEE Spring Simulation Interoperability Workshop, Huntsville, AL, 2006

[5] Hieb, M. and U. Schade, "Formalizing Command Intent Through Development of a Command and Control Grammar," 12th International Command and Control Research and Technology Symposium, Newport, RI, 2007

[6] Pullen, J., D. Corner, R. Wittman, A. Brook, O. Mevassvik, and A. Alstad, "Technical and Operational Issues in Combining MSDL and C-BML Standards for C2-Simulation Interoperation in MSG-085," NATO Modeling and Simulation Symposium, Stockholm, Sweden, October 2012

[7] Remmersmann, T., U. Schade, L. Khimeche, and B. Gautreau, "Lessons Recognized: How to Combine BML and MSDL," IEEE Spring Simulation Interoperability Workshop, Orlando, FL, 2012

[8] Heffner, K. C. Blais and K. Gupton, "Strategies for Alignment and Convergence of C-BML and MSDL," IEEE Fall 2012 Simulation Interoperability Workshop, Orlando, FY, 2012

[9] Pullen, J., D. Corner and R. Wittman, "Next Steps in MSDL and C-BML Alignment for Convergence, IEEE Spring 2013 Simulation Interoperability Workshop, San Diego, CA, 2013

[10] Perme, D., M. Hieb, J. Pullen, W. Sudnikovich, and A. Tolk, "Integrating Air and Ground Operations within a Common Battle Management Language," IEEE Fall Simulation Interoperability Workshop, Orlando FL, 2005

[11] Sudnikovich, W., A. Ritchie, P. de Champs, M. Hieb, and J. Pullen, "NATO Exploratory Team – 016 Integration Lessons Learned for C2IEDM and C-BML," IEEE Spring Simulation Interoperability Workshop, San Diego CA, 2006

[12] Hieb, M., S. Mackay, M. Powers, M. Kleiner, and J. Pullen, "The Environment in Network Centric Operations: A Framework for Command and Control," 12th International Command and Control Research and Technology Symposium, Newport, RI, 2007

[13] Galvin, K., W. Sudnikovich, P. deChamps, M. Hieb, J. Pullen, and L. Khimeche, "Delivering C2 to M&S Interoperability for NATO - Demonstrating Coalition Battle Management Language (C-BML) and the Way Ahead," IEEE Fall Simulation Interoperability Workshop, September 2006

[14] Pullen, J., M. Hieb, S. Levine, A. Tolk, and C. Blais, "Joint Battle Management Language (JBML) - US Contribution to the C-BML PDG and NATO MSG-048 TA," IEEE European Simulation Interoperability Workshop, June 2007

[15] de Reus, N., R. de Krom, O. Mevassvik, A. Alstad, U. Schade and M. Frey, "BML-enabling national C2 systems for coupling to Simulation,", IEEE Spring Simulation Interoperability Workshop, Newport, RI, 2008

[16] Gustavsson, P., M.R. Hieb, M. Groenkvist, V. Kamath, Jakob Blomberg, and Joakim Wemmergard. "BLACK-CACTUS – Towards an Agile Joint/Coalition Embedded C2 Training Environment," IEEE Spring Simulation Interoperability Workshop, Providence, RI, 2008

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

[18] Pullen, J. and K. Heffner, "Supporting Coalition Battle Management Language Experiments with Scripted Web Services", NATO 2009 Modeling and Simulation Symposium, Brussels, Belgium, 2009

[19] McAndrews, P., L. Nicklas and J. Pullen, "A Web-Based Coordination System for MSDL/C-BML Coalitions," IEEE Spring Simulation Interoperability Workshop, San Diego, CA, 2012

[20] Pullen, J., D. Corner, P. Gustavsson, and M. Grönkvist, "Incorporating C2---Simulation Interoperability Services Into an Operational C2 System," International Command and Control Research and Technology Symposium 2013, Alexandria, VA

[21] Pullen, J., D. Corner, R. Wittman, A. Brook, P. Gustavsson, U. Schade and T. Remmersmann, "Multi-Schema and Multi-Server Advances for C2-Simulation Interoperation in MSG-085," NATO Modeling and Simulation Symposium 2013 (submitted)

[22] Pullen, J., D. Corner, T. Remmersmann and I. Trautwein, "Linked Heterogenous BML Servers in NATO MSG-085," IEEE Fall 2013 Simulation Interoperability Workshop, Orlando, FL, 2013

[23] Pullen, J., D. Corner, P. Gustavsson and R. Witttman, "Order and Report Schema Translation in WISE-SBML Server," IEEE Fall 2013 Simulation Interoperability Workshop, San Diego, CA, 2013

Author Biographies

DR. J. MARK PULLEN is Professor of Computer Science at George Mason University, where he serves as Director of the C4I Center and also heads the Center's Networking and Simulation Laboratory. He is a Fellow of the IEEE, Fellow of the ACM and licensed Professional Engineer. He has served as Principal Investigator of the XBML, and JBML, IBML and SBML projects.

DOUGLAS CORNER is a member of the staff of the George Mason University C4I Center. He is the lead software developer on the SBML scripting engine.

MAGNUS GRÖNKVIST works for Saab AB, Security and Defence Solutions, Combat Systems and C4I Solutions. He is a senior systems engineer within the area of C4I-systems and interoperability.