An Open Source MSDL/C-BML Interface to VR-Forces

Dr. J. Mark Pullen, Mohammad Ababneh, Lisa Nicklas, Michael Connor and Alexandre Barreto
C4I Center
George Mason University
Fairfax, VA 22030, USA
+1 703 993 3682
{mpullen, mababneh, nicklas, mconnor, adebarro}@c4i.gmu.edu

Keywords:
SISO Military Scenario Definition Language, C-BML, MSDL, VR-Forces, Web Services, JC3I EDM, BML C2 GUI

SISO Military Scenario Definition Language (MSDL) and Coalition Battle Management Language (C-BML) continue to progress as a means to interoperate Command and Control (C2) systems with simulation systems. However, the ability to use these powerful interoperation technologies is limited to national military groups who possess a C2 or simulation system and choose to invest in an MSDL/C-BML interface. In order to expand the experimental capability for C2-Simulation technologies, both in our laboratory and in the larger community, the GMU C4I Center is undertaking development of an open source interface to the MÅK VR-Forces computer-generated forces simulation, which is commercially available, to join our other free, open-source BML software. We plan to compare the result with other systems in our testbed, such as US Army OneSAF and Saab WISE. The open source interface will begin as a student project; if successful we will seek funding to create a complete implementation of MSDL/C-BML. This paper describes the basic design and initial implementation.

1. Overview

This paper presents an open-source interface between BML in general (MSDL and C-BML in particular) with MÅK’s product VR-Forces through its Remote Control Application Programming Interface (API). So far as we are aware, this is the first trial to bring together open-source BML software with industrial-strength commercial software.

2. Introduction

The work presented in this paper is based on standards technologies developed in the SISO community. The core of these standards is MSDL and C-BML, which will be discussed in the next section. Our work builds on other enabling technologies developed by GMU’s C4I center that are available as open source software and also on the commercial simulation software VR-Forces from MÅK. We believe that integrating BML tools and standards with VR-Forces will synergize both technologies by expanding the reac of the openly available BML standards and tools to VR-Forces. It also will allow VR-Forces to be used by the wider community that is interested in standardized and open source tools and technologies. Among those users are NATO and other coalition nations and users.

2.1 BML and C-BML

Battle Management Language (BML) and its various proposed dialects, among them Coalition Battle Management Language (C-BML) supplemented by the Military Scenario Definition Language (MSDL), are intended to facilitate interoperation among command and control (C2) and modeling and simulation (M&S) systems by providing a common, agreed-to format for the exchange of information such as orders and reports [1]. In the predominant model used today, this is accomplished by providing a repository service that the participating systems can use to post and retrieve messages expressed in BML/C-BML. The service is implemented as middleware, essential to the operation of BML, and can be either centralized or distributed. Recent service implementations have focused on use of the 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][8].

2.2 SBML

Experience 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 database representation used to implement the BML Web Services. 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 the BML WS to store a representation of BML in a well-structured relational database, accessed via the Structured Query Language (SQL). This stability implies an opportunity for a re-usable system component: a Scripting Engine, driven by a BML Schema and a Mapping File. The resulting Scripted BML Server (SBMLServer) accepts BML push and pull transactions and processes them according to a script (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 that database can be implemented and tested rapidly
- the ability to change the service rapidly reduces cost and facilitates prototyping
- the scripting language provides a concise definition of BML-to-data model mappings that facilitates review and interchange needed for collaboration and standardization

The heart of SBML is a scripting engine, introduced in [2], that implements a BML WS by converting BML data into a database representation and also retrieving from the database and generating BML as output. It could implement any XML-based BML and any SQL-realized underlying data model. Current SBML scripts implement the Joint Command, Control and Consultation Information Exchange Data Model (JC3IEDM). In the following description, any logically consistent and complete data model could replace JC3IEDM.

Version 2 of SBML implements a publish/subscribe capability [5], using the Java Message Service (JMS) as implemented by JBoss in open source (see http://www.jboss.com). Version 2 also implements the XML Path Language (XPath) (see http://www.w3.org/TR/xpath), wherever a relative path in the XML input is required. The BML/JC3IEDM conversion process is accomplished under control of the scripting language, as described in [3].

SBML runs under JBoss 4.2.3, which provides the JBoss Messaging or JBossMQ. JBossMQ is an implementation of JMS 1.1 [6]. It provides both point-to-point messaging between two entities (JMS Queues) and a subscription based distribution mechanism (JMS Topics) for publishing messages to multiple subscribers. JMS provides reliable delivery of messages for all subscribers to a particular topic.

SBML version 2.3 provides a set of preconfigured JBossMQ Topics, which are used for the distribution of incoming orders and periodic reports to any interested subscribers. As BML messages are received they are processed by the appropriate script and written to the database. The successful completion of the transaction is an indication that there were no errors in incoming data and that the message can be forwarded to subscribers. There is an XPath [7] statement associated with each Topic, which serves as a filter to determine whether each received message should be written to that Topic. If application of the XPath statement to the message results in non-null result, the message is written to that Topic. A particular BML message may match more than one XPath statement and if so will be transmitted to more than one Topic. A client then might receive the same message more than once. The publish/subscribe architecture is depicted in Figure 2.

2.2.1 SBML Publish/Subscribe

Reference [3] describes the second generation of SBML.

Figure 1: SBML Configuration

The current SBML implementation and scripts support two JC3IEDM database interfaces, as shown in Figure 2: one is a direct SQL interface, used with a MySQL database server. The other, SIMCI_RI [4], passes Java objects through Red Hat’s Hibernate persistence service, which performs the actual database interface function.

Figure 2: Publish/Subscribe Architecture for SBML
2.3 MSDL

The Military Scenario Definition Language (MSDL) is intended to provide a standard mechanism for loading Military Scenarios independent of the application generating or using the scenario [9]. Standard MSDL is defined utilizing an XML schema thus enabling exchange of all or part of scenarios among Command and Control (C2) planning applications, simulations, and scenario development applications. XML based scenario representations can readily be checked for conformance against the standard’s schema.

The scope of MSDL is bounded by the situation, defined at one instant in time, combined with the course of action about to be taken in context to that situation. The intent is for MSDL to include that information which is either core or common to the situation and course of action (COA) of a military scenario. Definition of COA falls under the scope of the Coalition Battle Management Language PDG. The MSDL and C-BML Product Development Groups are collaborating on common elements of these two languages to ensure the two standards are compatible.

3. BML C2 GUI

The BML client depicted in Figures 1 and 2 above may be relatively complex to develop, requiring a variety of tools and capabilities such as: an XML editor, operating system command line knowledge, and Java programming experience. The difficulty is greater because the developer doesn’t have the opportunity to see the actual geospatial information in the BML documents on a representative map. Inspired by the Fraunhofer FKIE’s C2 Lexical Grammar (C2LG) GUI [10][11], we have developed the “Battle Management Language Command and Control Graphical User Interface” (BML C2 GUI) as one of the open source tools associated with SBML [12][13]. The BML C2 GUI is an open-source user interface tool that displays information flowing to/from C2 and simulation systems in text and image formats such as MSDL and C-BML.

The BML C2 GUI is different from the C2LG GUI in two aspects. The first one is that it integrates two comprehensive Open-Source software packages, JAXFront and OpenMap. Using the Community version of JAXFront results in a comprehensive XML customizable editor and OpenMap gave us a fully functional Mapping application. The second difference is that the BML C2 GUI is open source software. This makes it possible for the BML community to add to the software and customize it for the most appropriate use. Thus there is potential for accelerated the development time and reduced cost via re-use.

The main purpose of the BML C2 GUI is to provide an easy-to-use graphical user interface to BML users and developers that can serve as a surrogate input/output GUI or alternately to monitor (and if necessary revise) BML documents flowing to/from BML client systems [10]. BML C2 GUI is a Java application that generates an interface using other open-source tools: Xcentric's JAXFront and BBN's OpenMap. Figure 3 shows a screen shot of the GUI.

Figure 3: The BML C2 GUI

The BML C2 GUI provides an easy and efficient alternative for the end user to edit, validate, and push BML orders to the SBML Web Services and also to pull and view BML reports from the services. It also can subscribe to the SBML subscription service so that the GUI will be updated whenever a new report is pushed by another client with the SBML publish/subscribe capabilities. The map will depict geospatial information from the BML document the user is editing or revising and display the correct symbols representing the objects or units in addition to all the mapping capabilities supported by OpenMap. BML C2 GUI uses the OpenMap implementation of military standard MIL-STD-2525B for unit and object symbol representation [14].

Editing and issuing a BML order or report is very easy using the GUI, requiring only data field entry and selection of items from drop down lists, which are populated automatically from enumerations in the associated schema. The GUI can accommodate changes in BML schemas easily because all of the GUI generation happens at run time. Furthermore, the GUI satisfies the need of experienced users by providing a serialization method of the BML document enabling XML manual edit, validate, save and push capabilities. The BML C2 GUI has been enhanced to support viewing, editing, pushing and pulling of MSDL scenario files.
3.1 BML C2 GUI MSDL Enhancement

The MSDL schema was implemented in the BML C2 GUI, providing a run-time Java Swing interface for MSDL documents. Figure 6 illustrates a sample MSDL document in the editor part of the GUI and the corresponding Geospatial information on the right side. The geospatial capability included the representation of all geospatial information that can be found in a BML document. In the MSDL case, it includes the Organizations and Equipment information as in figure 4 and the Environment information as in figure 5.

4. MÄK VR-Forces

VT MÄK’s VR-Forces is a “powerful and flexible simulation environment for scenario generation. It has all the necessary features for use as a tactical leadership trainer, threat generator, behavior model test bed, or Computer Generated Forces (CGF) application” [17]. (See http://mak.com.) Figure 6 illustrates a typical VR-Forces configuration.

Some useful features of VR-Forces are:

- includes a C++ toolkit to extend or embed VR-Forces in another computer application
- can be used as a distributed simulation engine with remote GUI control
- can aggregate unit and entity modeling
- supports standard simulation protocols such as HLA and DIS
- supports various kinds of terrain, including streaming terrain
- supports GUI-based entity and parameter editing

4.1 Simple Scenario Editing

“VR-Forces Computer Generated Forces provides an intuitive GUI that allows to build scenarios by positioning forces, creating routes and waypoints, and assigning tasks or plans with a simple point and click” [17]. The basic outline can be drawn on a 2D tactical map, and then switched to the 3D scenario editing mode to accurately position entities within a complex urban environment. An XR mode is available to provide a big picture to understand the scenario, without losing the 3D perspective. Figure 7 shows a sample VR-Forces layout.

4.2 Powerful Simulation Engine

“VR-Forces comes with simulation models for a wide variety of battlefield entities and weapon systems. During scenario execution, VR-Forces vehicles and human entities interact with the terrain, follow roads, move in convoys, avoid obstacles, communicate over simulated radios, detect and engage enemy forces, and calculate damage. Through multi-resolution modeling, VR-Forces can switch between aggregate and entity level movement models ‘on-the-fly’ based on scenario events such as sensor detection or area of interest” [17].
5. MSDL/C-BML and VR-Forces Integration

One of our purposes in our BML work is to accommodate the growing demand for OpenBML use of C-BML standards and technologies by the Modeling and Simulation and Command and Control communities by providing open source tools such as SBMLServer and BML C2 GUI. We are now considering expanding our open source software to include and interface between MSDL/C-BML and the widely used, comprehensive and complex MÄK VR-Forces.

We are considering an open-source interface to reflect BML operations in VR-Forces. Here, we highlight the high level steps that must be taken to integrate C-BML Light with VR-Forces. Our goal is the arrangement shown in Figure 8.

5.1 Bridging MSDL/C-BML and VR-Forces

The new component we are planning is the open source Bridging Application to be created by the GMU C4I Center. This software will be written in Java interfaced with C++ so that it can use a set of classes in the VR-Forces API to interact with the VR-Forces simulator from the BML C2 GUI. As shown in figure 8, the bridging software will be a client of a C-BML server such as SBMLServer. Orders and MSDL are received via subscription to the RESTful version of SBMLServer (subscription to the SOAP is not natively available to C++ clients) either from a standalone Java application, or directly attached to the BML C2 GUI. The bridging software also is a client of the C-BML server so that it can provide reports, for example Position Status Reports. To get reports out of the VR-Forces simulator, we will create an interface message from VR-Forces that will report on the state repository of all entities at a set interval (see section 5.2 below).
MSDL information input to VR-Forces will be achieved in the same way as C-BML inputs. Preparation of MSDL instance documents for input to aggregated coalition MSDL, as described in [18], will require that the Bridging Application receive the state information as described in the next section.

5.2 Obtaining C-BML Reports from VR-Forces

The above discussion concerns about sending a BML order from the BML Server through the CBML-VRForces Bridge Application to a unit in VRForces simulation scenario. The other direction of integration clearly involves getting information from the VRForces scenario back to BML Server and client. As illustrated in figure 10 above, the unit’s entity state in the VRForces scenario will be sent back to the Bridging Application, which is, in turn, subscribed with the SBML Publish/Subscribe. Mapping will take place at the Bridging Application to convert VRForces entity state to standard position status BML report understood and supported by the SBMLServer and client. Once a new report is detected by any subscribed client (for example, the BML C2 GUI), the report will be rendered and presented to the client.

To develop this task, periodically, the Bridging Application asks DtExerciseConn about the state of his entities in simulation environment. DtExerciseConn is the class responsible for control all aspect in the simulation, as the scenarios files, time control and other simulations functions.

The information that DtExerciseConn provides is a list of reflection attributes of an entity (like positions in geocentric format, speed, acceleration, conditions of entity, etc.). This will allow the Bridging Application to extract information needed to create C-BML Reports.

5.3 Integration Plan

The integration will follow these phases of work:

1. The planned first version of the Bridge Application Software will interface with VR-Forces and submit a C-BML Light order to VR-Forces in the task format that it expects. This will require that entities referenced by the order be pre-loaded into VR-Forces using their GUI system. This version will subscribe to a RESTful version of SBMLServer. The Bridging Application will support only those C-BML actions matching VR-Forces tasks.

2. The planned second version of the Bridge Application Software will submit entities to VR-Forces from the MSDL received from SBMLServer via subscription. This will require mapping MSDL information to the VR-Forces entity format.

3. A third version of the Bridge Application Software will create an interface message between itself and VR-Forces. This interface message will report on the state of all entities currently being simulated, at a configurable time interval. This will require adding to the Bridging Application the ability to use the message information to create a C-BML Light position status report. The Bridge Application Software also will become a client of SBMLServer so that it can push these C-BML Light reports.

Figures 9 and 10 illustrate a VR-Forces scenario of a unit that has a task to move along a route. It is using the MIL-STD-2525B symbols in figure 9, while using a 3D display in figure 10. The goal of our work is to create an interface between C-BML/MSDL and such a scenario in VR-Forces, so that a two-way interaction can happen between the two environments. A BML open source end-user or client can send BML orders, tasks and reports to VR-Forces through the developed interface. The unit in VR-Forces will receive the BML order or task and execute it in a simulation environment, off-course after having it translated from BML format to a format that it can understand through the Bridging Application.
6. Conclusions

In this work we have presented a design for an interface between our open-source BML technology and the commercial simulation software VR-Forces. This interface will enable BML orders and tasks to be executed in the VR-Forces simulation environment. By integrating BML tools and standards with VR-Forces, we expect to synergize open source and commercial technologies by producing an open source C-BML/MSDL interface to VR-Forces. As a result, VR-Forces will be usable by the community investigating C2-simulation interoperation.

References


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 has served as Principal Investigator of the XBML and JBML projects.

MOHAMMAD ABABNEH is a PhD student at George Mason University pursuing a degree in Information Technology from its Engineering School and a member of the staff of its C4I Center. He is also a Lt.Col. at the Royal Jordanian Air Force. He is the lead software developer on the BML C2 GUI.

LISA NICKLAS is a member of the staff of the George Mason University C4I Center. She is the lead software developer on the SBML scripting engine.

MICHAEL CONNOR is a member of the staff of the George Mason University C4I Center. He is a Master of Science candidate in the GMU Department of Computer Science.

ALEXANDRE BARRETO is a doctoral student at the Instituto Tecnológico De Aeronáutica (ITA), São José dos Campos, Brazil and a Major in the Air Force of Brazil. He is a visiting researcher at the GMU C4I Center for 2012.