

Battle Management Language – Command and Control Graphical user Interface (BMLC2GUI)

Mohammad Ababneh, Dr. J. Mark Pullen

C4I Center

George Mason University

Fairfax, VA 22030, USA

+1 703 993 3682

{ mababneh, mpullen }@c4i.gmu.edu

Keywords:

Command and Control, Simulation, BML, SBML, Web Services, GUI

Battle Management Language (BML) and related technologies under development for the past few years have been shown to provide a promising capability for command and control to simulation interoperability. BML enables a robust "System of Systems" or a "Coalition of Systems". However, the "last mile problem" for BML development is to have a user interface that represents information flowing to/from C2 and simulation systems. We have developed an open-source tool for this purpose: the BMLC2GUI. Inspired by Fraunhofer FKIE's C2 Lexical Grammar GUI. The main purpose of the BMLC2GUI is to provide an easy-to-use and open-source 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. BMLC2GUI is an application, developed in Java that generates an interface using other open-source tools: Xcentric's JAXFront, which generates a GUI interface at run-time for any XML document based on its schema, and BBN's OpenMap, which is used to display geospatial data and control features residing in the BML document (orders or reports) on the map. The BMLC2GUI provides easy and efficient means for the end user to edit validate and push BML orders to the Scripted BML (SBML) Server Web Services and other means to pull and view BML reports from the services. The BMLC2GUI also can subscribe to the SBML subscription service so that the GUI will be updated whenever a new report is push by another client. This paper introduces the BMLC2GUI and explains how it works with the SBML Server to provide convenient capabilities to BML developers and users.

1. Overview

The BML client depicted in Figure 2 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 GUI as described in [1], we have developed the "Battle Management Language Command and Control Graphical User Interface" (BMLC2GUI) as part of the open source tools associated with SBML. The BMLC2GUI is an open-source user interface tool that represents information flowing to/from C2 and simulation systems in text and image formats.

The main purpose of the BMLC2GUI 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. BMLC2GUI is a Java application that generates an interface using other open-source tools: Xcentric's JAXFront and BBN's OpenMap. Figure 1 shows a screen shot of the GUI.

The BMLC2GUI 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. The BMLC2GUI also can subscribe to the SBML subscription service so that the GUI will be updated whenever a new report is pushed by another client under 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. BMLC2GUI is using the OpenMap's implementation of military standard MilStd2525b for unit and object symbol representation [2].

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.

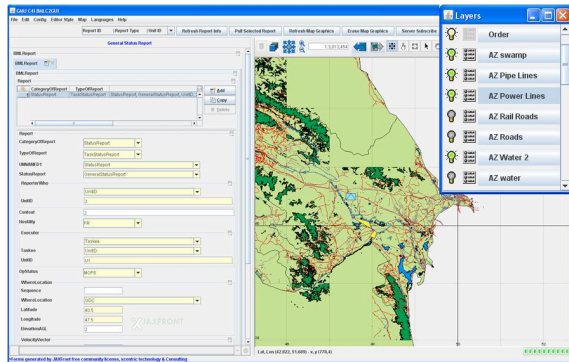


Figure 1: The BMLC2GUI

2. Background

The BMLC2GUI was developed to enhance the use and implementation of BML and related technologies. In this section we provide some background information about these.

2.1 BML

Battle Management Language (BML) and its various proposed extensions are intended to facilitate interoperability 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. This is 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, essential to the operation of BML, and can be either centralized or distributed. Recent 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 [3].

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 implies an opportunity for a re-usable system component: a Scripting Engine, driven by a BML Schema and a Mapping File that 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 [4], 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. Reference [5] describes the second generation of SBML.

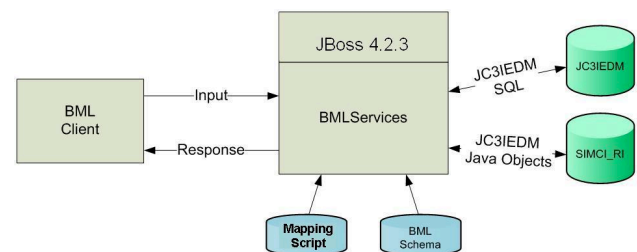


Figure 2: SBML Configuration

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 [6], passes java objects through Red Hat's Hibernate persistence service, which performs the actual database interface function.

2.3 SBML Publish/Subscribe

Version 2 of SBML implements a publish/subscribe capability [7], 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 which is described in [5].

SBML runs under JBoss 4.2.3, which provides the JBoss Messaging or JBossMQ. JBossMQ is an implementation Java JMS 1.1 [8]. 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 [9] 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 therefore could 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 3.

The BMLC2GUI implements the SBMLSubscriber Client application to connect to the subscription service of the SBMLServer. This client connectivity enables the GUI to listen to any BML activity such as reports being generated. The main usage of the Subscriber in the GUI is to pull new BML reports of any type and display them immediately on the screen in the editing/viewing panel in addition to extracting

the geospatial information and display it in the map panel.

This usage provides an indirect advantage of our work and proves that our tool can also be used as a C2 system. Some modifications to the code can make it a more advanced C2 system.

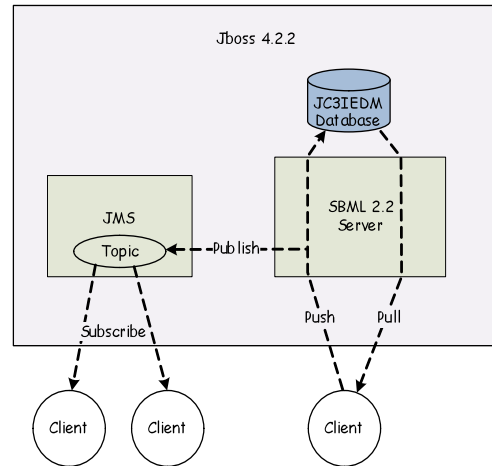


Figure 3: Publish/Subscribe Architecture for SBML

2.4 C2LG GUI vs. BMLC2GUI

Command and Control Lexical Grammar (C2LG) is a formal grammar supporting and enforcing BML [10] [11]. It enables the generation of standard representation of BML orders, reports or requests. The C2LG GUI was developed to make it easier for the user to formulate correct C2LG messages [1].

The BMLC2GUI is different from the C2LG GUI in two aspects. The first one is that it is making use of two comprehensive Open-Source software packages and integrates them. This integration produces a tool that is supporting all the functionality of both of them. The Use of the Community version of JAXFront gave us a comprehensive XML customizable editor and OpenMap gave us a fully functional Mapping application. The second difference is that The BMLC2GUI is Open-Source by itself. This makes it possible for the BML community to add to the software and customize it for the most appropriate use. Off-course the use of open-source software had accelerated the development time and reduced the cost.

3. Editing BML Documents Using JaxFront

The BML C2 GUI uses the Free Community version (Open-Source) of JaxFront as a major component for editing BML documents. JaxFront is a technology

developed by Switzerland's Xcentric Technology & Consulting GmbH. It is "a technology to generate graphical user interfaces on multiple channels (Java Swing, HTML, PDF) on the basis of an XML schema". See web site <http://www.jaxfront.org>. The user can dynamically generate GUIs that allow the user to edit XML data without being exposed to the underlying XML technology. JAXFront is both an Open-Source and a commercial product. It has some requirements so it can be used as open-source in any project. The most important requirement is that the project has also to be open-source and publicly available. Figure 4 shows the JAXFront's architecture.

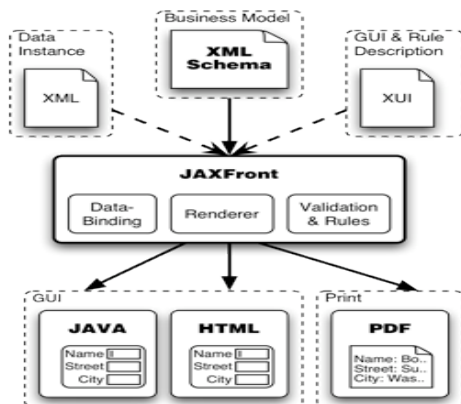


Figure 4: JAXFront architecture.

3.1 JAXFront's XUI Editor

In addition to the open-source distribution libraries, JAXFront introduces the "XUI Editor". The XML User Interface (XUI) editor is used to produce XUI definitions (.xui files). The XUI file is a definition and description of the user interface in XML. The use of the XUI files gives us more control and customization capability over the user interface.

The BMLC2GUI is able to generate the user interface even without specifying an XUI definition causing a default view. Assigning a null value to the xuiUrl variable in section 6 will produce this result. We have used the XUI editor to create a "Tab" view for all BML documents. XUI editor is also available as a community edition that can be downloaded and installed to generate more customizable views. Appendix 1 shows the XUI file used to customize the view of a BML General Status Report.

4. Using OpenMap for Geospatial Information Representation

OpenMap is an Open Source JavaBeans based Geospatial development tool. OpenMap can quickly

be used to build applications and applets that access data from legacy databases and applications. It provides various capabilities to allow users to see and manipulate geospatial information. The BMLC2GUI is using the latest version of OpenMap 4.6.5, released March 5, 2009. See <http://openmap.bbn.com>.

OpenMap supports various map data files and this release of the BMLC2GUI is using ESRI shape files. Our GUI uses OpenMap to display multiple data layers on the map in addition to drawing BML objects, units and control measures.

5. BMLC2GUI Design and Development Goals

The development of the BMLC2GUI was inspired by the FGAN C2LG. Unlike C2LG, the BMLC2GUI had considered the following goals:

- Open-Source: The BMLC2GUI is using two marvelous Open-Source tools; JaxFront and OpenMap and is Open-Source itself. Being Open-Source will make it much easier for the BML community to adapt and customize the tool for the desired usage.
- Ease of use: The BMLClient depicted in Figure 1 represents a relatively professional and experienced client with XML tools like an XML Editor and Operating System command knowledge because the current SBMLClient package distribution supports only OS command mode. The GUI hides these details from the user.

6. BMLC2GUI Development

The core of the BMLC2GUI is the use of JAXFront Open-Source libraries to build a new and a customized type of XML Document Object Module (DOM) that can be rendered as Java Swing objects. The DOM Builder takes the following parameters to generate a JAXFront document:

- The XML document: The W3C XML [12]
- The Schema: The W3C XSD [13]
- The XUI if available
- The root node

BML orders and reports are treated without much differentiation as documents. The only difference is in the parameters used to build the document especially the XUI file, which controls the view of the document. JAXFront's DOMBuilder uses xerces DOM as part of its effort to build the renderable XML document.

```
currentDom = DOMBuilder.  
getInstance(). build("default-
```

```
context", xsdUrl, xmlUrl,
xuiUrl, root);
```

The other important JAXFront's component used by the GUI is the EditorPanel; which is a Java swing panel that hosts all the swing components generated by the DOMBuilder. The BMLC2GUI uses a Java swing JSplitPane that displays the JAXFront's Editor Panel in its left component. The OpenMap's MapPanel is displayed in the right component.

After the successful rendering of the XML document, we become interested in the extracting the geospatial information (Latitude, Longitude coordinates representing positions or dimensions of objects) on the map. We use the JAXFront's document "getRootType" method to parse the document into a string and then populate an array with the document's elements and values.

```
bmlString =
currentDom.getRootType().
getDisplayValue();
```

The MapHandler is the class in OpenMap responsible for managing the layers of geospatial data that need to be displayed on the MapPanel. The BMLC2GUI implements a RouteLayer class for drawing BML's geospatial data. The RoutLayer's constructor accepts the array representing the document and the Lat-Lon coordinates are then extracted from the array based on the criteria that they will always be preceded by a "GDC" tag and transferred to a temporary array. Based on the type of the object and count of Lat-Lon pairs the RouteLayer draws the right shape which is a:

- Point: if there is only one pair of Lat-Lon coordinates
- Polygon: if the number of pairs is greater than 1 and the last pair was the same as the first pair.
- Line: if the number of pairs is greater than 1 and the last pair was not the same as the first pair.

The RouteLayer also extracts the UnitID value and invokes the SBML Service "ListWho" to get the rest of unit information in order to collect the 14 characters used by MILSTD2525b in order to draw the correct symbol at the right position. If the unit information is unavailable for any reason then a default symbol is drawn.

When subscribed to the SBMLServer's Publish-Subscribe, the BMLC2GUI receives the JMS topic

message; which includes the BML XML document. The tool uses XPATH to parse the document to obtain the report type and then calls the "openReportWS" method that displays the report in the Editor Panel, extracts the geospatial data and displays the corresponding graphical object(s) at the correct position. When multiple reports are arriving consequently and they are all related to one BML activity like unit position, the tool can serve as a command and control (C2) system. Figure 5 shows a unit moving.

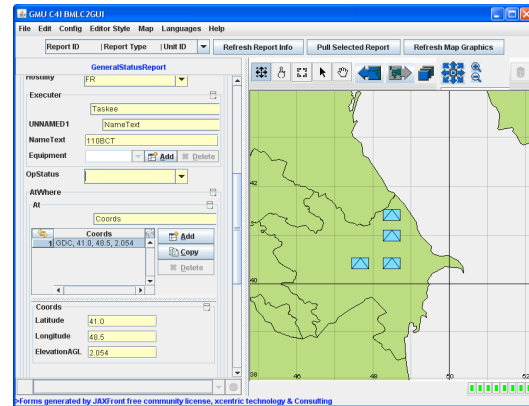


Figure 5: A Unit Movement

7. BMLC2GUI Functionality

In this section, we provide a brief description of the main functionality of the BMLC2GUI.

7.1 Editing a BML Document

The BML C2 GUI can edit any type of BML document. The user has the capability to create a new order or a report in the GUI. The user has the flexibility of changing, deleting, validating, serializing the document and when ready he can push the document to the web services.

7.2 Serialization of a BML Document

The BML C2 GUI provides the user with the capability to see the XML source of any document he is editing. This feature is very useful, especially to the experienced and advanced users.

7.3 Validation of a BML Document

The BML C2 GUI provides the user with the capability to validate the BML document against its schema. This serves as a validation option before sending the document to any web service in order to guarantee well conformance with the schema. When the validation option is selected, all the possible validation problems can be displayed in the GUI's

status bar and also a red box can be drawn around the text area having the problem.

7.4 Pulling a BML Document

The BML C2 GUI provides the user with the capability to pull reports from the SBMLServer through two approaches:

a. By activating the “SBMLSubscriber” listener application, which listens to any coming new report and automatically detects its type and displays it in the editor area. It will also extract any geospatial data from the report and display it on the map.

b. By selecting the desired report from a Report Information list of the latest reports added from the web service. The report info includes: “Report ID”, “Report Type” and “Object ID”. The user can get the latest reports by pressing the “Refresh Report Info” and after selecting the report he can click “Pull Selected Report” to open it.

7.5 Pushing a BML Document

The BMLC2GUI provides the capability to create, edit, validate (optional) and push any type of BML document, especially Orders. In the community of BML users, it is highly desirable to have a tool that provides an elegant and simple user interface for creating and editing BML documents. The GUI provides the capability for validation also as mentioned in section 7.3. For some considerations, mostly in an environment of experienced users or at testing stages, validation might be skipped and the tool is flexible towards this.

When the user decides that he/she is ready to push the order, the GUI provides him with a simple user interface to do so with a click of a button. Off-course this interface is a shell to the call of the SBMLClient application with parameters representing the target of this push operation (Server name or IP address and domain name).

7.6 Retrieving Latest Reports

The BMLC2GUI provides the user with the capability to see a list of reports consisting of the Report ID, Type and Object/Unit ID. The user has the capability to refresh this list manually in addition to the automatic update while the subscriber is running. The user can select any report from the list and get a view of that report: document view and geospatial information representation.

8. SBML Services

The BMLC2GUI uses the following Web Services maintained by SBMLServer through the SBMLClient application.

a. CallListWho

The SBML command “callListWho” is used by the GUI to fetch all the necessary information about a unit given its UnitID in order to compose the MILSTD2525b key (String characters) that enables the tool to draw the correct unit symbol in its desired position.

```
<callListWho>
<UnitID>3</UnitID>
</callListWho>
```

b. GetLatestReportIDs

The SBML command “GetLatestReportIDs” is used by the GUI to build a list of report information: Report ID, Type and Object/Unit ID. This command is used every time the user wishes to refresh the list of Latest reports.

```
<GetLatestReportIDs>
</GetLatestReportIDs>
```

c. ReportPull

The SBML command “ReportPull” is used by the GUI to pull a report from the SBML Web Service so that it can be viewed or edited and its geospatial information be extracted and illustrated on the map.

```
<ReportPull>
<ReportID>410</ReportID>
</ReportPull>
```

9. BMLC2GUI Configuration

The BMLC2GUI makes use of the same JAXFront editing capabilities to configure its environment variables that otherwise need to be hard-coded; such as the SBML Server name or IP address, the domain name and the location of the folder containing most of the tool components. A schema has been developed first that describes these variables. Using the schema the tool edits the XML file containing the current tool configuration. After any editing and saving these changes, the configuration is activated immediately. Appendix 1 shows a sample configuration XML file and its schema. Figure 6 shows the BMLC2GUI Configuration screen.

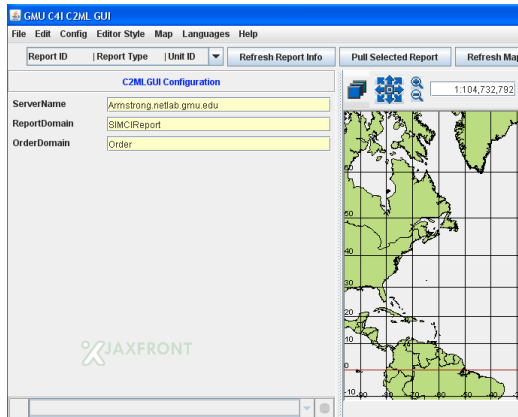


Figure 6: BMLC2GUI Configuration

9.1 Map Configuration

The BMLC2GUI is using OpenMap as its mapping tool. So, it is capable of doing anything possible in OpenMap including support to various types of data formats. This release uses ESRI Shape files (.shp). The user can add/ remove any shape data file by editing the BMLC2GUI.properties file that exists in the package's folder. This mechanism of configuration avoids the hard-coding of such information and gives the user the flexibility to construct the layers he/she is interested in or are available. Appendix 2 displays the first few configuration pages of a working configuration.

10. Conclusions

The BMLC2GUI provides an easy-to-use and a comprehensive tool for the BML's end-user. It reliefs the user from the use of a separate XML editor and knowledge of platform specific instructions to run BML pull and push commands. It provides the user with the capability of seeing the geospatial information in the documents that he is editing or viewing. It enables the user to validate and serialize the documents even without being connected to the services at the editor level instead of waiting until the execution results appear. And finally being open-source makes it less expensive to own and operate such a tool and at the same time makes it possible for customization and enhancements by the BML community.

11. Future Work

When we started developing the BMLC2GUI, it was not in our intension to use it as a C2 tool. Maybe because the SBML web services and the Publish-Subscribe capabilities were still under development. At this time, and after the maturity level that SBML and its related technologies had reached, we feel that the BMLC2GUI can be used as a C2 tool with some

improvements to the code so that better performance and robustness can be achieved when used for this purpose. The tool was not tested rigorously in this side because this feature came as an indirect added value.

References

- [1] Pullen, J. *et al.*, "An Expanded C2-Simulation Experimental Environment Based on BML," IEEE Spring Simulation Interoperability Workshop, Orlando, FL, 2010
- [2] US Department of Defense, "Interface Standard Common War fighting Symbology MIL-STD-2525B w/Change 1", July 2005
- [3] Carey, S., M. Kleiner, M. Hieb, and R. Brown, "Standardizing Battle Management Language – A Vital Move Towards the Army Transformation," IEEE Fall Simulation Interoperability Workshop, Orlando, FL, 2001
- [4] 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
- [5] Pullen, J., D. Corner and S. Singapogu, "Scripted Battle Management Language Web Service Version 2," IEEE Fall 2009 Simulation Interoperability Workshop, Orlando, FL, 2009
- [6] Levine, S., L. Topor, T. Troccola, and J. Pullen, "A Practical Example of the Integration of Simulations, Battle Command, and Modern Technology," IEEE European Simulation Interoperability Workshop, Istanbul, Turkey, 2009
- [7] Corner, D., J. Pullen, S. Singapogu and B. Bulusu, "Adding Publish/Subscribe to the Scripted Battle Management Language Web Service," IEEE Spring 2010 Simulation Interoperability Workshop, Orlando FL, 2010
- [8] Sun Microsystems, "JAVA Message Service 1.1" April 12, 2002.
- [9] World Wide Web Consortium, "XPath Language Version 1.0", November 16, 1999
- [10] Schade, U. and M. Hieb, "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, 2006
- [11] Schade, U., and M. Hieb. "Battle Management Language: A Grammar for Specifying Reports," IEEE Spring Simulation Interoperability Workshop, 2007, Norfolk, VA

[12] World Wide Web Consortium, “Extensible Markup Language (XML)”, <http://www.w3.org/standards/techs/xml>.

[13] World Wide Web Consortium, “XML Schema: Formal Description”, <http://www.w3.org/standards/xml/schema>.

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 the Information Technology & Engineering School of George Mason University and member of the staff of its C4I Center. He is also a major at the Royal Jordanian Air Force. He is the lead software developer on the BMLC2GUI.

Appendix 1: A Sample General Status Report XUI File

```
<?xml version="1.0" encoding="UTF-8"?>
<?jaxfront version=2.61;time=2010-03-30
15:51:20.328;xsd=IBMLReports.xsd;xui=file:/C:/C2MLGUI/XUIView/GeneralStatusReportView.xui;nsMapping=%KEYhttp://www.jaxfront.com/xui%VAL%KEYhttp://www.w3.org/2001/XMLSchema%VALxsd%KEYhttp://www.w3.org/2001/XMLSchema-instance%VALxsi?>
<XUI xsi:schemaLocation="http://www.jaxfront.com/xui
jar:file:/C:/Program%20Files/JAXFront-2.6/lib/jaxfront-core.jar!/xui.xsd"
xmlns="http://www.jaxfront.com/xui" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <component xpath="/BMLREPORT">
    <style>
      <treeEntry>
        <occurrence>
          <visibility>
            <never/>
          </visibility>
        </occurrence>
      </treeEntry>
    </style>
  </component>
  <component
xpath="/BMLREPORT/Report/UNNAMED1/StatusReport/PositionStatusReport/AtWhere/bml:ControlFeature">
    <style>
      <mode visible="false">
        <message/>
      </mode>
    </style>
  </component>
  <component
xpath="/BMLREPORT/Report[1]/UNNAMED1/StatusReport/GeneralStatusReport/AtWhere/bml:ControlFeature">
    <style>
      <mode visible="false">
        <message/>
      </mode>
      <treeEntry>
        <occurrence>
          <visibility>
            <never/>
          </visibility>
        </occurrence>
      </treeEntry>
    </style>
  </component>
</XUI>
```

Appendix 2: A Sample BMLC2GUI Configuration File and Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<?jaxfront version=2.50;time=2010-06-30
09:47:21.921;xui=C2MLGUIConfigView.xui;xsd=C2MLGUIConfig.xsd?>
<SBMLServer xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C2MLGUIConfig.xsd">
    <ServerName>129.174.95.15</ServerName>
    <ReportDomain>SIMCIReport</ReportDomain>
    <OrderDomain>Order</OrderDomain>
</SBMLServer>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
    <xs:element name="SBMLServer">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="ServerName"/>
                <xs:element ref="ReportDomain"/>
                <xs:element ref="OrderDomain"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="ServerName" type="xs:NCName"/>
    <xs:element name="ReportDomain" type="xs:NCName"/>
    <xs:element name="OrderDomain" type="xs:NCName"/>
</xs:schema>
```

Appendix 3: A Sample OpenMap Layer Configuration (c2ml.property)

```
# *****
# Properties file for BMLC2GUI
# Desc   : The purpose of this file is to remove the hard coding of Shape
#         Layers inside the Java Program. The Layers related to any
#         part of the world can be added in this file without modifying
#         the program code.
# Author : Mohammad Ababneh - GMU C4I Center
# Date   : 12/28/2009
# *****

# #####
# These properties define the starting projection of the map.
# These properties are listed in com.bbn.openmap.Environment.java,
# and affect the initialization of the application.
# #####

# Latitude and longitude in decimal degrees
#c2ml.Latitude=41.5f
#c2ml.Longitude=-71f
#Scale: zoom level (1:scale)
#c2ml.Scale=10000000f

# *****
# Layers to be loaded and shown on the map
# *****

c2ml.components=menuBar fileMenu helpMenu
menuBar.class=com.bbn.openmap.gui.MenuBar
fileMenu.class=com.bbn.openmap.gui.FileMenu
helpMenu.class=com.bbn.openmap.gui.DefaultHelpMenu

# graticule and world should be the last two
# graticule world

c2ml.layers= AZswamp AZpipelines AZpowerlines AZrailroads AZroads AZwater2
AZwater AZlandmarks AZtracks AZtrees AZboundaries AF_settlements
AF_airport_airfeilds AF_health_facilities AF_lakes AF_district_boundary
AF_international_boundary Af_Rivers AF_irrigated_areas AF_landcover
AF_provincial_boundary AF_watersheds AF_river_region graticule world

# *****
# Graticule Layer
# *****
graticule.class=com.bbn.openmap.layer.GraticuleLayer
graticule.prettyName=Graticule

# *****
# World Political Map Layer
# *****
world.class=com.bbn.openmap.layer.shape.ShapeLayer
world.prettyName=Political Solid
world.shapeFile=data/azer/world_adm0.shp
world.spatialIndex=data/azer/vmap_area_thin.ssx
world.lineColor=000000
world.fillColor=BDDE83
```

```
# *****
# Afghanistan watersheds
# *****
AF_watersheds.class=com.bbn.openmap.layer.shape.ShapeLayer
AF_watersheds.prettyName=AF watersheds
AF_watersheds.shapeFile=C://BMLC2GUI//Afghanistan//watersheds//watershed.shp
AF_watersheds.spatialIndex=data/azer/vmap_area_thin.ssx
AF_watersheds.lineColor=007FFF
AF_watersheds.fillColor=007FFF

# *****
# Afaghanistan settlements
# *****
AF_settlements.class=com.bbn.openmap.layer.shape.ShapeLayer
AF_settlements.prettyName=AF settlements
AF_settlements.shapeFile=C://BMLC2GUI//Afghanistan//settlements//07_03_settlements.shp
AF_settlements.spatialIndex=data/azer/vmap_area_thin.ssx
AF_settlements.lineColor=777777
AF_settlements.fillColor=ffbde83

# *****
# Azerbaijan tracks Layer
# *****
AZtracks.class=com.bbn.openmap.layer.shape.ShapeLayer
AZtracks.prettyName=AZ tracks
AZtracks.shapeFile=C://BMLC2GUI//azer//TrackL.shp
AZtracks.spatialIndex=data/azer/vmap_area_thin.ssx
AZtracks.lineColor=777777
AZtracks.fillColor=ffbde83

# *****
# Azerbaijan trees Layer
# *****
AZtrees.class=com.bbn.openmap.layer.shape.ShapeLayer
AZtrees.prettyName=AZ trees
AZtrees.shapeFile=C://BMLC2GUI//azer//TreesA.shp
AZtrees.spatialIndex=data/azer/vmap_area_thin.ssx
AZtrees.lineColor=000000
AZtrees.fillColor=009040
```