16th CCRTS
"Collective C2 in Multinational Civil-Military Operations"

Title of Paper:

An Open Source Graphical User Interface Surrogate C2 System for Battle Management Language Experimentation

Topics:

Topic 6: Experimentation, Metrics, and Analysis
Topic 8: Architectures, Technologies, and Tools
Topic 10: C2, Management, and Governance in Civil-Military Operations

Authors:

Mohammad Ababneh [STUDENT] and J. Mark Pullen

C4I Center
George Mason University
4400 University Dr.
Fairfax, VA 22030
mpullen@c4i.gmu.edu

Abstract

Battle Management Language (BML) has been under development since 2003. BML shows a promising capability for command and control to simulation interoperability, enabling a robust system of systems for training, mission rehearsal, and course of action analysis. An important issue in BML development is the ability to inspect and modify XML-encoded information flowing among C2 and simulation systems. We have developed an open-source tool for this purpose, called the BMLC2GUI. Our work was inspired by Fraunhofer FKIE's C2 Lexical Grammar GUI. The BMLC2GUI provides an easy to use, open-source graphical user interface for BML users and developers. It can serve not only as a development tool but also as a surrogate for C2 system input/output.

The BMLC2GUI was developed as a Java application. It uses other open-source tools to generate a user interface. JAXFront, from Xcentric, generates forms at run-time, based on the schema of an XML document. OpenMap, from BBN, is used to display geospatial data and control features residing in the BML document (Orders and Reports) on the map. The BMLC2GUI is able to act as data source or sink and provide an examination capability for development testing by pushing and pulling XML files from the supporting Web service and also by subscribing to a publish/subscribe interface. Moreover, it has the facilities necessary to stand in for a minimal C2 system that can be used when developing simulations and interfacing them to BML.

1. Introduction

This paper describes a developmental supporting technology for interoperation of command and control (C2) systems with simulation systems. The general technology area on which we report is Battle Management Language (BML), which aims to provide an unambiguous information exchange for such interoperation [1-3]. C2 electronic documents implemented in BML are Orders and Reports; also, in the future Requests (documents that provide tasks to be done, but without the force of command; for example, Request for Fire). The supporting technology is the BML Command & Control Graphical User Interface (BMLC2GUI), which provides a tool for development of BML interfaces by inspection/modification of data in BML messages encoded in the Extensible Markup Language (XML) and also can stand in for a minimal C2 system to generate orders and receive reports. We refer to the BMLC2GUI at some places in this document as "the GUI."

1.1 BML and BML experiments

The NATO Modeling and Simulation Group (MSG) technical activity 048 (MSG-048) was chartered to evaluate the potential of BML for coalition C2-simulation interoperability. Coalition operations have a need for interoperability that is even greater than that of national Service and Joint operations. Because coalitions must function under greater complexity due to significant differences among doctrine and human language barriers, the agility to train and rehearse rapidly

before the actual operation is highly important [4]. MSG-048 adopted a SOA approach [5]; its first major demonstration employed the WS development for JBML. Our organization developed SBML to support subsequent demonstrations and experimentation by MSG-048. The scope of the final experimentation conducted by MSG-048 [6-8] is evident in Figure 1. The potential of this approach to extend to multinational civil-military operations, based on the availability of C2 systems and simulations that work across the civil-military boundary, is evident.

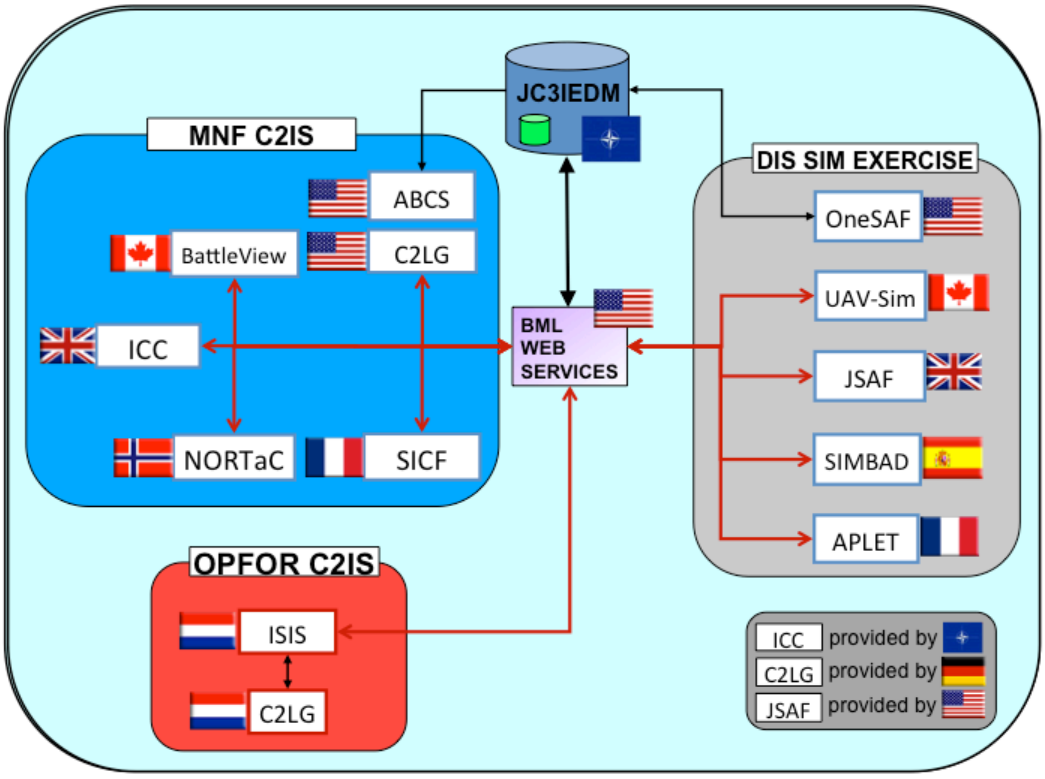

Figure 1. MSG-048 Experimentation Architecture

During MSG-048 experimetnation, the Command and Control Lexical Grammar graphical user interface (C2LG) system element in Figure 1, developed by Fraunhofer FKIE, proved a valuable system element in that it supported review and, where necessary, modification of BML orders in the experimental environment. In consultation with Fraunhofer FKIE, we determined that an open-source interface with similar functionality would provide a valuable capability for the BML community as it engages in developing, prototyping, and experimentation.

1.2 BMLC2GUI

Accordingly, we have developed the BMLC2GUI, based on inspiration from Fraunhofer FKIE's C2LG GUI with additional functionality described in this paper, as part of the open source tools associated with our Scripted BML (SBML) server project [9]. The GUI 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 GUI is to provide an easy-to-use graphical user interface to BML users and developers that can serve as a surrogate input/output C2 GUI or alternately to monitor (and, if necessary, revise) BML documents flowing to/from BML client systems. The GUI is a Java application that generates an interface using other open-source tools: Xcentric's JAXFront and BBN's OpenMap. Figure 2 shows a screen capture of the GUI.

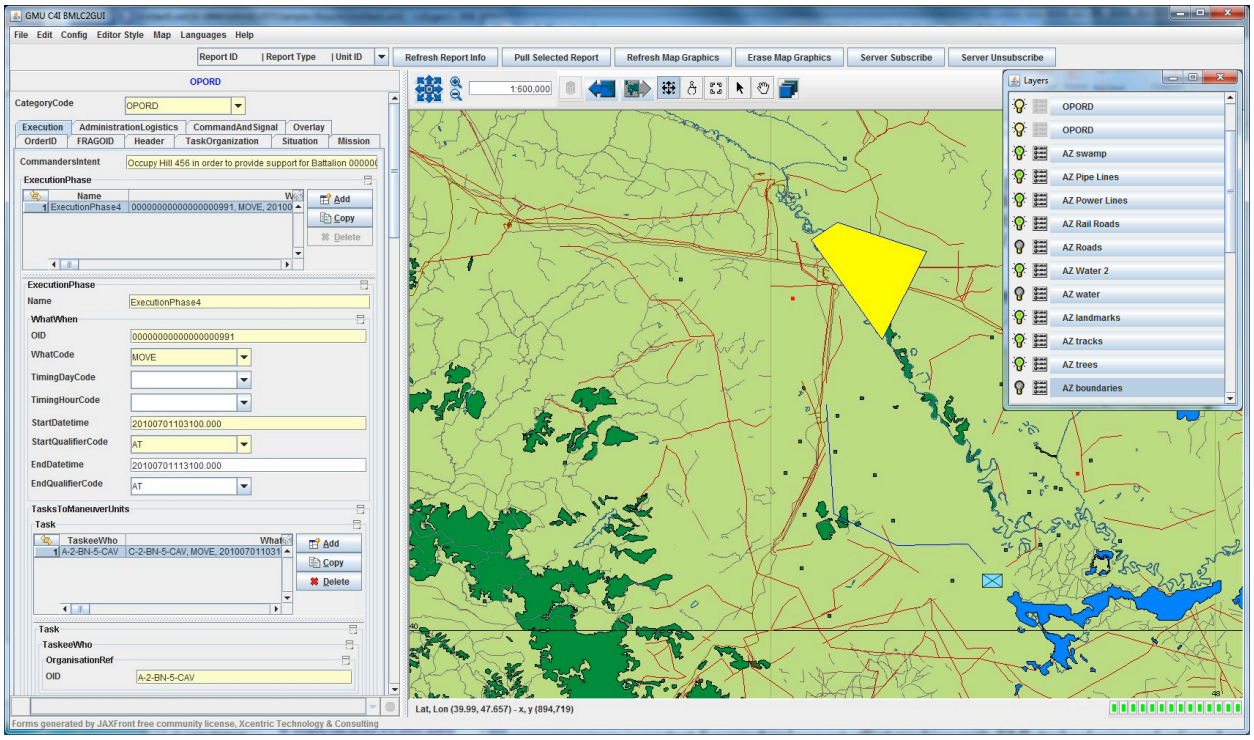

Figure 2. The BMLC2GUI

The 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. The GUI also can subscribe to the SBML subscription service so that the GUI will be updated whenever a new report is published. The map will depict geospatial information from the BML document the user is editing or revising, displaying the correct symbols representing the objects or units in addition to all the mapping capabilities supported by OpenMap. BMLC2GUI uses the OpenMap implementation of military standard MilStd2525b for unit and object symbol representation [10].

Editing and issuing a BML order or report is simple to achieve using the GUI. It requires 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 provides for serialization for BML documents, enabling manual edit of XML, including validate, save, and push capabilities.

2. GMU Open Source BML Systems

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. The software and extensive documents are available at http://c4i.gmu.edu/BML.

2.1 BML

Battle Management Language (BML) and its various proposed extensions 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. This is accomplished by providing a repository service that the participating systems can use to post and retrieve messages expressed in BML, which allows a configuration with multiple C2 and simulation systems to operate without all systems online and functioning simultaneously, resulting in a more robust overall system.

The BML service is implemented as middleware and can be either centralized or distributed. Recent implementations have employed standard 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 [5]. As an academic research center, our mission is to support the extension of knowledge, both through research and by supporting development of early versions of a new capability. One way we do that is to implement open source software systems, designed for use in a prototyping or experimental environment, that are sufficiently robust to support community developmental efforts broadly. The Scripted BML server and BMLC2GUI were developed as part of such an effort.

2.2 Scripted BML

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 combined with a Mapping File that accepts BML *push* and *pull* transactions and processes them according to a script, also written in XML, but with an alternate representation in a more concise Condensed Scripting Language (CSL). 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 underlying the 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, described in [9], that implements a BML WS by converting BML data into a database representation and also by retrieving information 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 [7], using the Java Message Service (JMS) as implemented by JBoss in open source [12]. Version 2 also implements the XML Path Language (XPATH) [13,14], wherever a relative path in the input is required. JBoss 4.2.3 JBoss Messaging, an implementation Java JMS 1.1 [15] which 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. There is an XPath statement associated with each Topic, serving as a filter to determine whether each received message should be written to that Topic. The publish/subscribe architecture is depicted in Figure 3.

2.3 BMLC2GUI

The GUI complements SBML in our open source suite of tools. It is beneficial because C2 and simulation client systems do not necessarily display the XML documents they produce and consume. However, it sometimes is beneficial to modify those XML documents in system prototyping and software debugging and to display geospatial data from the XML documents graphically. The GUI performs these functions. It is available as open source and so can be modified to meet related needs in other BML projects. Thus it can expedite prototyping and component development for future BML systems.

The GUI 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 Subscriber in the GUI accepts new BML Reports of any type and displays them immediately on the screen in the editing/viewing panel in addition to extracting the geospatial information and displaying it in the map panel. This extends the power of publish/subscribe operation to the GUI.

3. BMLC2GUI Requirements

This section describes the basic functional requirements of the GUI, in terms of the supporting technologies it encapsulates.

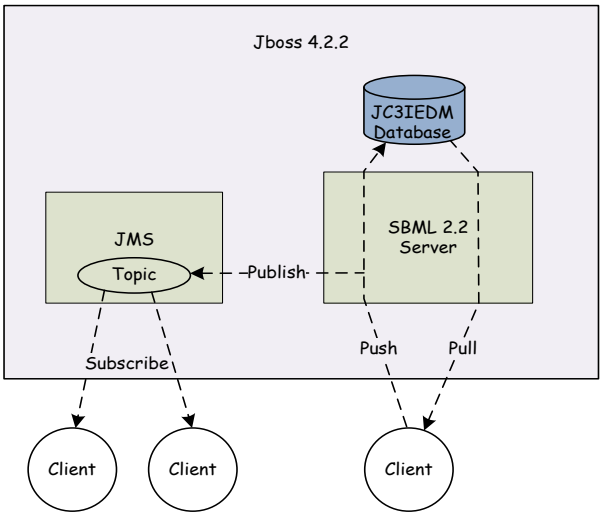

Figure 3. Publish/Subscribe Architecture for SBML

3.1 Generating BML Forms Automatically

The GUI uses the Free Community version (Open-Source) of JaxFront as its major component for editing BML documents. JaxFront is a technology developed by Switzerland's Xcentric Technology & Consulting GmbH, which can "generate graphical user interfaces on multiple channels (Java Swing, HTML, PDF) on the basis of an XML schema" [16]. It can be used to dynamically generate GUIs that allow the user to edit XML data without understanding the underlying XML technology. JAXFront is both an open-source and a commercial product. It has some restrictions if it is to be used as open-source in any project. The most significant of these is that the project has also to be open-source and publicly available. Figure 4 shows the architecture of JAXFront.

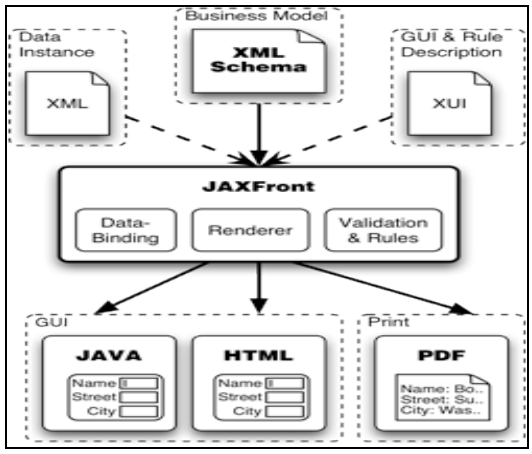

Figure 4. JAXFront's Architecture.

3.2 JAXFront's XUI Editor

In addition to the open-source distribution libraries, JAXFront provides an "XUI Editor". The XML User Interface (XUI) editor is used to produce XUI definitions (.xui files). The XUI file defines and describes features of the user interface in XML, providing control and customization capability for the user interface.

The GUI is able to generate the user interface without an XUI definition causing a default view, by assigning a null value to the xuiUrl configuration variable. However, we use the XUI editor to create a "Tab" view that generally is more suitable for BML documents.

3.3 Embedded Geographic Information System Interface

OpenMap is an open-source JavaBeans-based geospatial development tool. OpenMap can be used to quickly 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. Currently, the GUI uses OpenMap 4.6.5, released March 5, 2009 [17].

OpenMap supports various map data files; as of writing the GUI 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. The GUI uses the OpenMap implementation of the MilStd2525b unit symbols as illustrated in figure 5. It also can show associated geographical information layers if available in the map file.

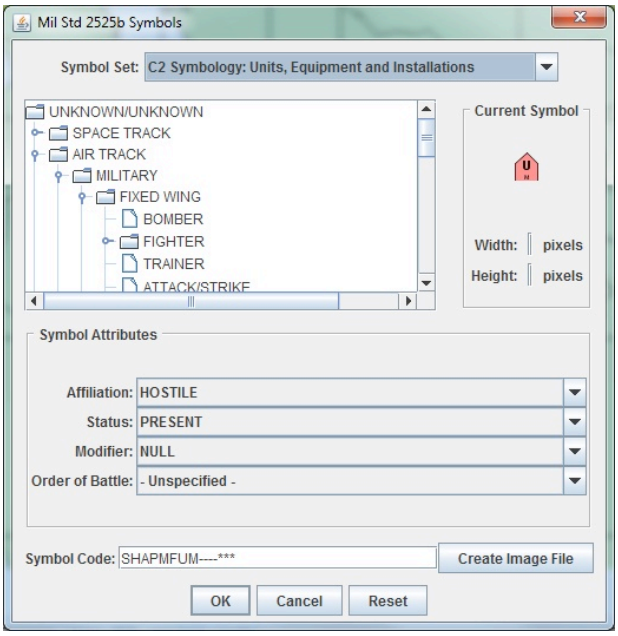

Figure 5. OpenMap MilStd2525b implementation

4. Development of the BMLC2GUI

The development of the GUI required two significantly different efforts. The first of these was the development of the tool itself, including implementation of the two major open-source tools described above, without which the GUI could not exist. The second effort was integration of other BML technologies developed by the larger BML team. These included implementation of the SBML client, subscription to the service, and interaction with the various query services that respond to the GUI requests. The following subsections provide more details about the development. Additional information is available in [18]; open-source software and documentation is available at http://c4i.gmu.edu/BML.

4.1 BMLC2GUI development goals inspired by C2LG-GUI

Command and Control Lexical Grammar (C2LG) is a formal grammar supporting and enforcing BML [19,20]. 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 [8].

The BMLC2GUI's goals differ from those of the C2LG GUI:

- Open source: The BMLC2GUI employs two well-developed open source tools; JaxFront and OpenMap, and is itself open source software. This makes it available for the BML community to adapt and customize without being limited and constrained by the licensing conditions of other software. C2LG was developed under support of the German Ministry of Defense and is not available for general use.

- Quick response to changes: in a prototyping or experimental environment, where the XML schema may change frequently, the ability for the tools to adapt automatically without reprogramming is important.

- Ease of use: The BMLClient depicted in Figure 3 employs XML tools including an XML Editor but without requiring the user to know technical details of operating system interaction that is required to employ the SBML package. The GUI hides these details from the user. Further, because of its open source nature, the GUI makes available a working code example of SBML client implementation for developers of other clients.

- Low development cost: making use of open-source tools greatly accelerated the development and reduced the cost of the GUI.

4.2 User interface development

The core of the GUI employs 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 requires the following parameters to generate a JAXFront document:

- The XML document: in W3C-standard XML format [21]
- The schema: in W3C-standard XSD format [22]
- The optional JaxFront XML User Interface (XUI) file
- Identification of the document's root node

BML orders and reports both are treated as XML documents. The only difference between the two lies in the parameters used to build the document, particularly the XUI file, which controls the view of the document in addition to the schema. JAXFront's DOMBuilder uses Xerces DOM [23] to build the renderable XML document.

The other important JAXFront 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 GUI uses a Java swing JSplitPane that displays the JAXFront Editor Panel in its left component. A complementary tool is displayed in the right component; in the present implementation, this is the OpenMap MapPanel.

After successful rendering of the XML document, the GUI extracts geospatial information and displays it on the map. Latitude and longitude (LAT-LON) coordinates represent positions of objects. This is accomplished by using the JAXFront document "getRootType" method to parse the document into a string and then populate an array with the document's elements and values.

The MapHandler class in OpenMap is responsible for managing the layers of geospatial data that need to be displayed on the MapPanel. The GUI implements a RouteLayer class for drawing BMLs geospatial data. The RouteLayer constructor accepts an array representing the LAT-LON coordinates of control measure objects and the locations and positions of other objects. Based on the type of the object and count of LAT-LON pairs, the RouteLayer draws the corresponding shape, which can be one of the following basic shapes:

- Point: if there is only one LAT-LON coordinate pair
- 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 (actually, this is a sequence of connected line segments).

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 to draw the correct symbol at the exact position. If the unit information is unavailable, then a default symbol will be drawn.

The GUI can be configured to subscribe to the SBML server's publish/subscribe service. If this is done, the GUI will receive Java Messaging Service (JMS) Topic messages from the server. These are BML/XML documents: Orders and Reports. The GUI 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 arrive consequently, and they are all related to a single BML information element such as unit position, the GUI can be considered to be in a role serving as a command and control (C2) system for situational awareness. Figure 6 shows a unit moving along a track, which is actually an accumulated view of multiple position reports.

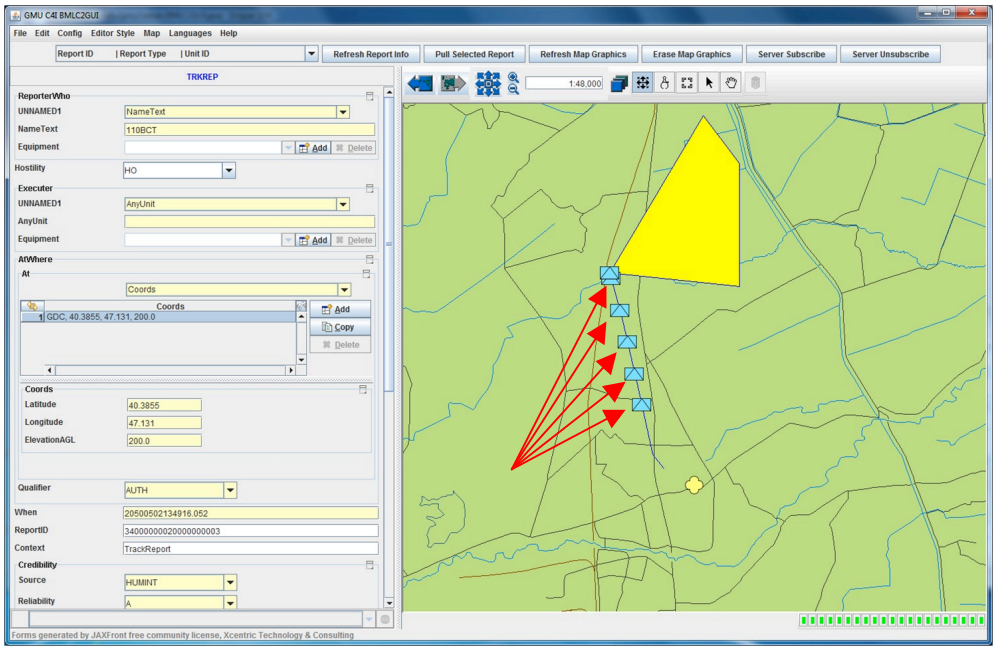

Figure 6. Unit Movement Track

4.3 SBML service interface development

The BMLC2GUI provides a graphical interface to BML Web services that are intended to be accessed by exchange of XML messages. These services are provided by the SBML server and accessed through an SBML client application imbedded in the GUI:

- *CallListWho:* This SBML command is used by the GUI to retrieve all the necessary information about a unit given its UnitID in order to compose the MILSTD2525b key (14 character string). OpenMap's implementation of MILSTD2525b then takes this string and generates the unit's symbol at the reported position.

```
<callListWho>
 <UnitID>1-A-2-BN-5-CAV</UnitID>
</callListWho>
```

- *GetLatestReportIDs:* This SBML command is used by the GUI to build a list of reports information: Report ID, Type and Object/Unit ID. This command is used when the user wishes to refresh the list of Latest reports.

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

- *ReportPull:* This SBML command 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>00000000000000000000901</ReportID>
</ReportPull>
```

5 BMLC2GUI Functions

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

5.1 Editing a BML document

The GUI can edit any type of BML/XML document. The user has the capability in the GUI to create a new order or a report or to edit an existing one. This includes changing, deleting, validating, and serializing the document; also the ability to push the document to a Web service. This capability was the core of our project: providing an easy-to-use, user-friendly interface to BML/XML documents that otherwise would have to be edited manually through a text editor or with a more generic XML editor.

5.2 Serialization of a BML document

The GUI provides a capability to access the XML source of any document being edited. The user can save this serialized source to a standard XML file that can be used with any application supporting the XML standard. This feature is very useful to users may want to view the XML-coded format and also in setting up experiments related to newly developed BML capabilities. We anticipate that it also will be useful with integration efforts when BML is used between systems or organizations, including civil/military interoperation.

5.3 Validating a BML document

The GUI provides a capability to validate a BML document against the XML schema pertaining to that document. This allows validation before sending the document to the Web service in order to confirm conformance with the schema, without which the BML document will not be usable by the Web service. When the validation option is selected, validation problems can be displayed in the GUI's status bar and also a red box can be produced around the XML text area exhibiting the problem.

5.4 Pulling a BML document

The GUI provides a capability to pull BML documents from the SBML server in three possible ways:

1. From the file system, by entering the OrderID or ReportID. If the Order or Report exists, it will be displayed in the editor otherwise an error message will be generated.
2. By activating the "SBMLSubscriber" listener client application, which subscribes to a Topic, listens to published BML documents (most often, Reports), automatically detects the type of BML document, and displays it in the editor area. It will also extract any geospatial data from the report and display it on the map.
3. 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 "Pull Selected Report" will open it.

5.5 Pushing a BML document

The GUI provides a capability to create, edit, validate (optionally) and push any BML/XML document. This is used most often for Orders. In the community of BML users, it is highly desirable to have a tool that provides a clear, simple user interface for creating and editing BML documents. The GUI provides the capability for validation also as described in section 5.3; however, in some situations (for example, when it is known the producing software is providing valid output documents), validation might be skipped.

When the user is ready to push the order, the GUI provides a simple user interface to do so with a click of a button. This activates the SBMLClient application with parameters representing the target of this push operation (server designation plus the document to be pushed). Based on these parameters, the server processes the Order or Report, publishes it, and saves it to the intended servers according to its configuration.

5.6 Retrieving latest reports

The GUI provides a capability to obtain a drop-down 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 view that report on the GUI, both document view and geospatial information representation.

5.7 Configuration and customization at runtime

The BMLC2GUI makes use of the same JAXFront editing capabilities to configure its environment variables that would otherwise need to be hard-coded (for example, the SBML server name or network address, the domain name, and the location of the folder containing most of the tool components). An XML file describing this data can be edited at runtime to redefine the GUI configuration. After editing and saving these changes, the configuration is activated immediately. Appendix 2 shows a sample configuration XML file and its schema. Figure 7 shows the GUI Configuration screen.

Some of the most important configuration items in the GUI are:

- SBML Server: designates server from which orders are pulled or subscribed
- Report BML type
- Order Server: designates server to which orders are pushed.
- Domain Name: used to indicate how SBML will handle the document
- WhereXPath of the geospatial information in the BML document:
  used to display the information on the map
- Initial map LAT-LON for startup
- Map scale for startup
- BMLC2GUI software path in the host computer

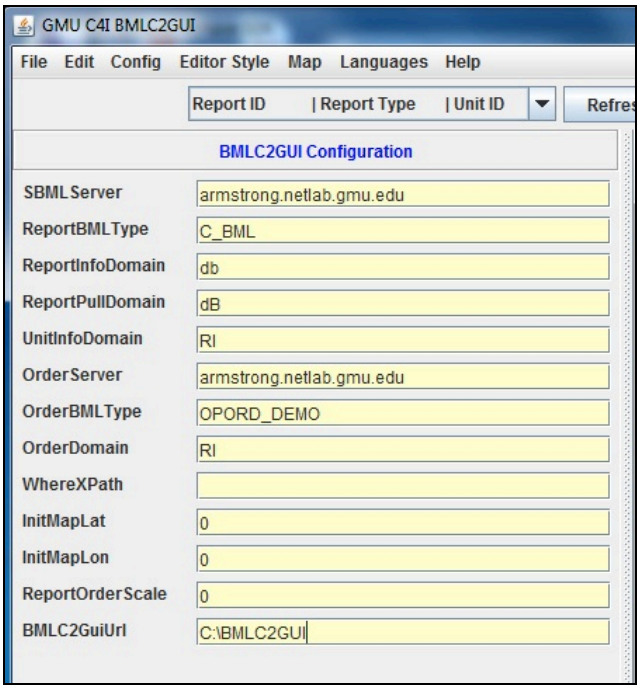

Figure 7. BMLC2GUI Configuration

5.8 Map configuration

The BMLC2GUI geospatial interface is modular and could be used with various mapping or image display components. As of this writing, the GUI uses OpenMap version 4.6.5 in this role and is able to take advantage of all OpenMap features. OpenMap supports a variety of data formats; this release of the GUI uses ESRI "shape" format (extension .shp). The user can add or remove any of the map data files by editing the "BMLC2GUI.properties" OpenMap configuration file. This avoids coding the information in the GUI code and gives the user the flexibility to specify graphic layers that can be enabled to support the C2 view.

6 Using the GUI as Surrogate C2 system

In this section we provide two examples that demonstrate the use of the BMLC2GUI tool as an editor with geospatial capabilities and a surrogate command and control tool.

6.1 *Example 1: IBML Order and Reports: OneSAF and IBML/SBML*

This example shows the use of the GUI in the Integration of OneSAF and Integrated BML (IBML) [8] technologies. The role that the GUI played in the reported integration experiment spanned all phases. While this experiment provided evidence of the compatibility with other technologies, it also showed the accuracy of the tool in representing available geospatial information. Figure 8 shows the principal functions employed.

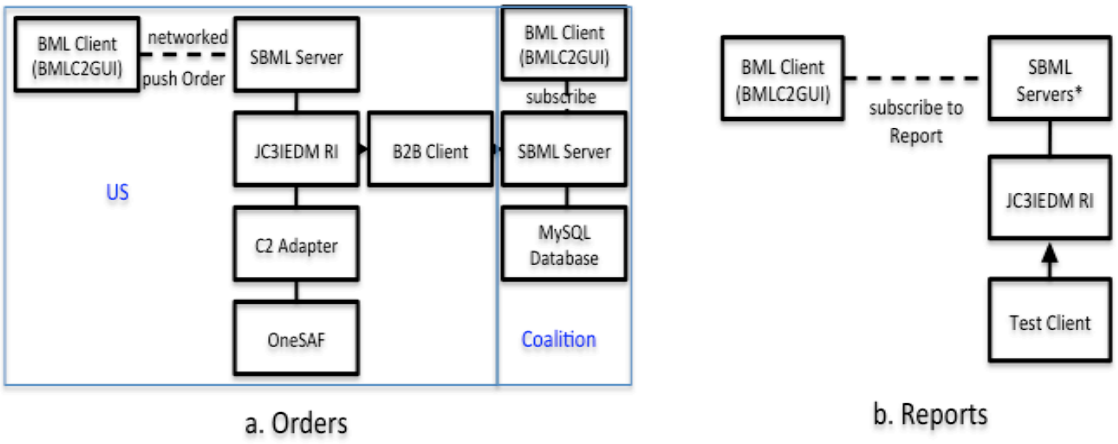

Figure 8. Using the BMLC2GUI in Integration Process

The steps of the experiment and the role of the BMLC2GUI are as follows:

1. Editing the IBML order: The GUI was used to open an existing IBML order and edit it to reflect the intended contents. Figure 9 shows the GUI's editor with the IBML order.

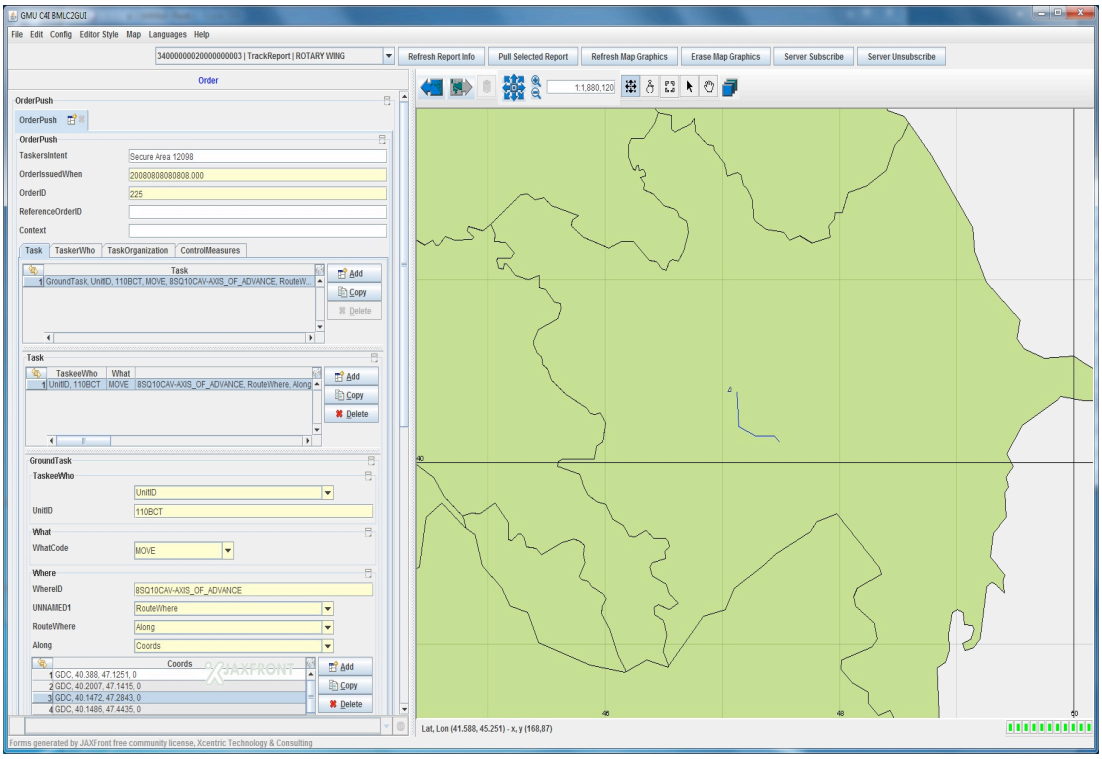

Figure 9. IBML Order in text panel of GUI

2. Validating the IBML Order: The GUI operator can validate the IBML order by selecting a menu option. The validation results will be displayed and the operator can behave accordingly.

3. Pushing the Order: The operator selects to push the order by selecting the menu option and gets feedback whether the order was pushed successfully.

4. Pulling the Order: The operator pulls the Order by selecting a menu option and providing the OrderID; the GUI retrieves the order from the web services and the persisted database.

5. Subscription: The Operator subscribes to the Subscription service of the SBML server by clicking a button. The OneSAF simulation will start generating reports according to the Order. These Reports go to the SBML server and are published to subscribing clients.

6. Receiving reports: The GUI will start receiving position reports as shown in Figure 10. The unit details level depends on the unit and the Report format, and can be confirmed on the OneSAF display.

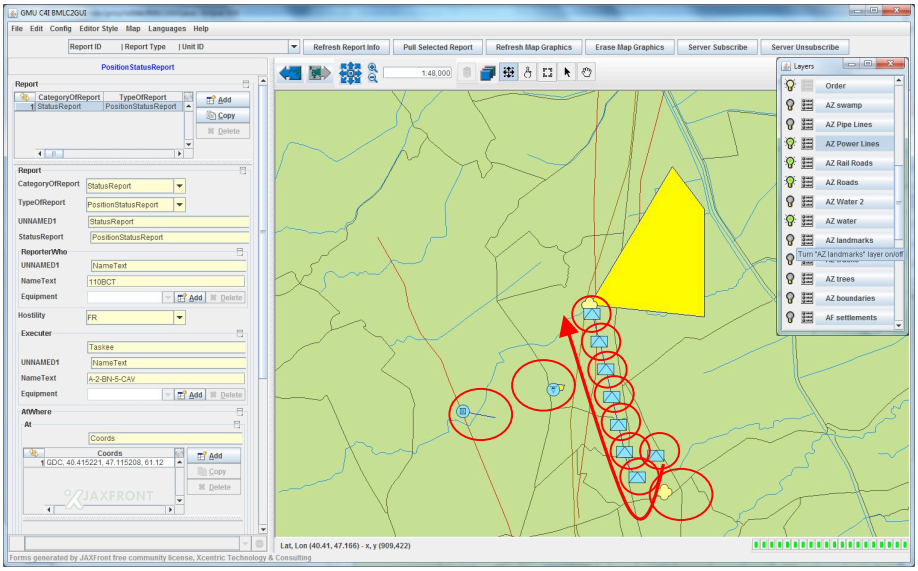

Figure 10. Experiment position, track, mine and bridge reports

6.2 Example 2: NATO OPORD using SISO C-BML draft schema

This example shows the benefit of using the BMLC2GUI with BML technologies under development. At the time of writing, a new C-BML schema is being developed by the SISO Coalition Battle Management Language Product Development Group (C-BML PDG) [24]. In situations where the schema is still under development it is beneficial to give the users and the development community an immediate feedback for every change or addition. This is facilitated by the GUI's ability to generate forms at runtime, based on the latest schema draft.

Supporting the C-BML development team at GMU, the BMLC2GUI assisted in various ways.

1. Based on the schema under development, the GUI provided a graphical interface form for OPORD editing; entry and modification giving the developers an instantaneous feedback of the quality and the correctness of each new idea implemented. This editor part was generated easily through a menu item selection and provided a tab view of the OPORD. It also provided drop down menus for the selection of already enumerated values from the schema. Figure 11 shows the OPORD in the GUI.

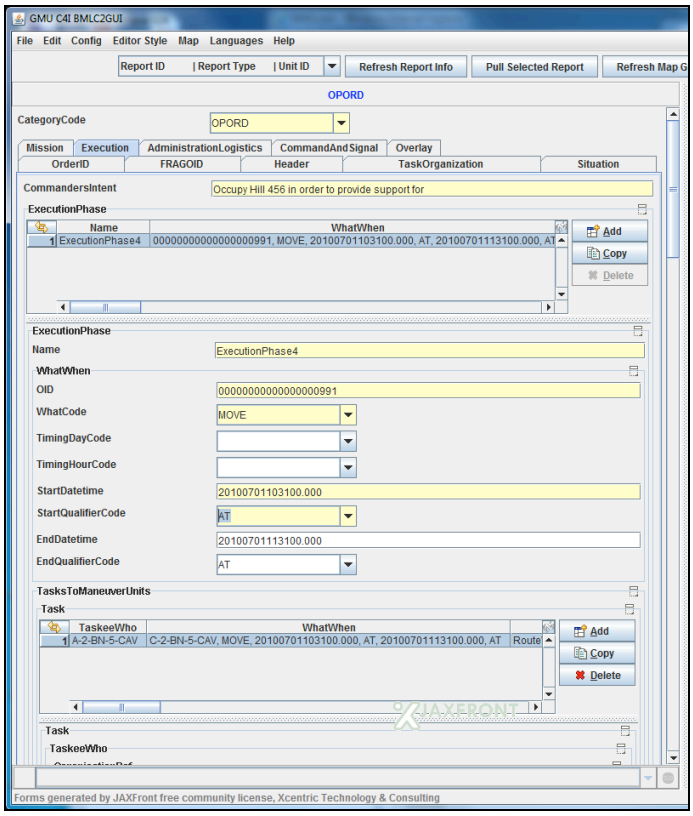

Figure 11. OPORD in the BMLC2GUI Editor based on draft schema

2. The GUI played a useful role in validating the schema and the OPORD documents because of the built-in validation capabilities provided by JaxFront. This is essential for any development effort working with XML encoding; having it provided automatically by the GUI relieved the developers from running a separate validation step.

3. The GUI was used to generate example OPORD documents containing geospatial information, based on the new schema. The GUI is able to generate realistic examples containing accurate location and unit movement information and display them on the map in addition to geographical layers available. Figure 12 shows the geographical information representation of the OPORD on the map.

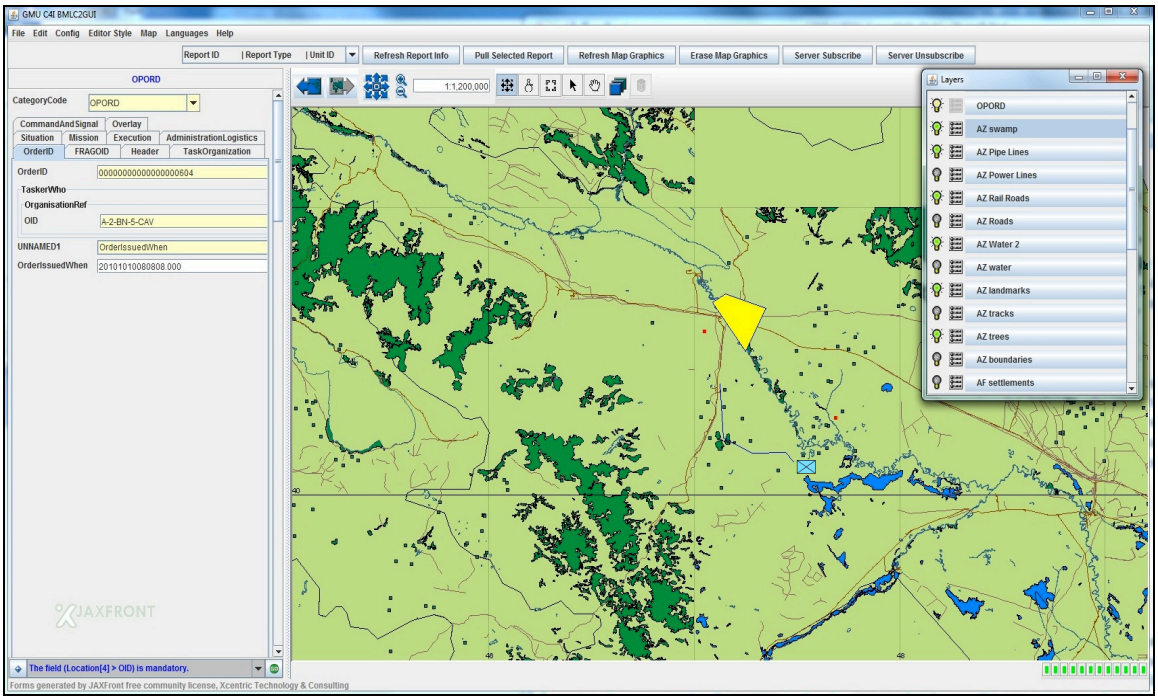

Figure 12. BMLC2GUI OPORD Mapping capabilities supported by OpenMap.

4. The BMLC2GUI was used to receive the reports generated after pushing the OPORD through the SBML web services using the publish/subscribe capabilities of the SBML Server and the client implemented in the GUI tool. Various types of reports are displayed immediately after being published, so developers may see the units or objects either positioned in their locations or moving towards their targets in a manner similar to Figure 10 above.

7 Conclusions and Future Work

The BMLC2GUI provides a powerful, easy-to-use tool for BML developers that also can be used as a surrogate C2 system, with these capabilities:

- Generates easy-to-use forms for entering, reading, and editing XML documents, based on latest schemas available at runtime.
- Relieves the user from the use of a separate XML editor and knowledge of platform specific instructions to run BML pull and push commands.
- Provides capability of seeing the geospatial information in the documents being edited or viewed.
- 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.
- Has open source availability that makes it less expensive to obtain and operate such a tool and at the same time makes possible customization and enhancements by the BML community.

Our original intention for creating the BMLC2GUI was to support the development process. However, we have used it multiple times to stand in for C2 systems that either were unavailable or would have been expensive to obtain for experimental work. This gives the GUI even more potential value to the C2, simulation, and operational communities as the use of BML grows. We look forward to seeing its use and adaptation for US, coalition, and civil-military systems that interoperate C2 and simulations.

8 References

[1] 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

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

[3] Levine, S. *et al.*, "Joint Battle Management Language (JBML) Phase 1 Development and Demonstration Results," IEEE Fall Simulation Interoperability Workshop, Orlando, FL, 2007

[4] Tolk, A, M. Hieb, K. Galvin, L. Khimeche, and J. Pullen, "Developing a Coalition Battle Management Language to facilitate Interoperability between Operation CIS and Simulations in support of Training and Mission Rehearsal", 10th Command and Control Research and Technology Symposium*,* McLean, VA, 2005

[5] Pullen, J., M. Hieb and S. Levine, "Using Web Service-Based Command and Control to Support Coalition Collaboration in C2 and Simulation," 13th International Command and Control Research and Technology Symposium*,* Seattle, WA, 2008

[6] 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, 2008, Newport, RI

[7] Heffner, K. *et al.*, "NATO MSG-048 C-BML Final Report Summary," SCS/SISO Euro-Simulation Interoperability Workshop, Ottawa, Canada, 2010

[8] Pullen, J. *et al.*, "Integrating National C2 and Simulation Systems for BML Experimentation," SCS/SISO Euro Simulation Interoperability Workshop, Ottawa, Canada, 2010

[9] Pullen, J., D. Corner, L. Nicklas and S. Singapogu, "Supporting NATO C2-Simulation Experimentation with Scripted Web Services", 15th International Command and Control Research and Technology Symposium*,* Quebec, Canada, 2011

[10] US Department of Defense, "Interface Standard Common War fighting Symbology MIL-STD-2525B w/Change 1", July 2005

[11] 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

[12] http://www.jboss.org

[13] http:// www.w3.org/TR/xpath

[14] World Wide Web Consortium, "XPath Language Version 1.0", November 16, 1999

[15] Sun Microsystems, "JAVA Message Service 1.1" April 12, 2002.

[16] http://www.jaxfront.org

[17] http://openmap.bbn.com

[18] Ababneh, M. and J. Pullen, "Battle Management Language – Command and Control Graphical Use Interface," IEEE Fall Simulation Interoperability Workshop, Orlando, FL, 2010

[19] 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

[20] Schade, U., and M. Hieb. "Battle Management Language: A Grammar for Specifying Reports," IEEE Spring Simulation Interoperability Workshop, 2007, Norfolk, VA

[21] World Wide Web Consortium, "Extensible Markup Language (XML)", http://www.w3.org/standard/techs/xml

[22] World Wide Web Consortium, "XML Schema: Formal Description", http://www.w3.org/standards/xml/schema

[23] http://xerces.apache.org

[24] http://www.sisostds.org/StandardsActivities/DevelopmentGroups

Appendix 1: A Sample Mine Field Report XUI File

```
<?xml version="1.0" encoding="UTF-8"?>
<?jaxfront version=2.61;time=2010-02-04
11:09:03.218;xsd=IBMLSIMCIReports.xsd;nsMapping=%KEYhttp://www.jaxfront.com/xui%VAL%KEYht
tp://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="/MINOBREP">
        <style>
            <treeEntry>
                <occurrence>
                    <visibility>
                        <never/>
                    </visibility>
                </occurrence>
            </treeEntry>
        </style>
    </component>
    <component xpath="/MINOBREP/CornerLocation/bml:ControlFeature">
        <style>
            <mode visible="false">
                <message/>
            </mode>
        </style>
    </component>
    <component xpath="/MINOBREP/Parameters/BypassGrid/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

```
<?xml version="1.0" encoding="UTF-8"?>
<?jaxfront version=2.50;time=2011-01-28
10:41:12.45;xui=BMLC2GUIConfigView.xui;xsd=BMLC2GUIConfig.xsd?>
<SBMLServer xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="BMLC2GUIConfig.xsd">
        <SBMLServer>armstrong.netlab.gmu.edu</SBMLServer>
        <ReportBMLType>C_BML</ReportBMLType>
        <ReportInfoDomain>db</ReportInfoDomain>
        <ReportPullDomain>dB</ReportPullDomain>
        <UnitInfoDomain>RI</UnitInfoDomain>
        <OrderServer>armstrong.netlab.gmu.edu</OrderServer>
        <OrderBMLType>OPORD_DEMO</OrderBMLType>
        <OrderDomain>RI</OrderDomain>
        <WhereXPath/>
        <InitMapLat>0</InitMapLat>
        <InitMapLon>0</InitMapLon>
        <ReportOrderScale>0</ReportOrderScale>
        <BMLC2GuiUrl/>
</SBMLServer>
```