Maturing Supporting Software for C2-Simulation Interoperation

J. Mark Pullen and Lisa Nicklas
George Mason University C4I Center
Presentation Overview

• BML Architecture
• Scripted BML Background
• SBML Enhancements
  – Programmability
  – Publish/subscribe of dynamic topics
  – Performance issues
  – Example of application
• Client support: BMLC2GUI
• Conclusions
BML Purpose and Operation

• Facilitates C2-Simulation interoperation
  – Exchange of Orders and reports in standard format

• Current architecture uses a repository service to hold state submitted by client C2 and Simulation systems
  – Web service with XML input – Network Centric
  – Data stored in JC3IEDM and can be replicated
BML Architecture

Command and Control Systems

BML Messages (Orders, Reports, etc.)

Simulation Systems

BML Web Services

JC3IEDM and other databases
A Note on Open Source

- Our laboratory has open source policy
- Community shares supporting software
  - Often, server/middleware
- Supports re-use at system and component levels
- Avoids vendor lock-in and dead-end
- Open release avoids some restrictions (ITAR)
  - But can’t be used if software contains any sensitive information
BML Community is Well-Suited to Open Source Server/Middleware

- International group with shared need for software to support interoperation
- Initial capability comes from a university
  - Sharing is natural in academic world
- Other forms of sharing require complex agreements
- Works best when participants provide specific feedback on problems
  - MSG-048 sometimes identified location of bug and fix
  - Next step: contribute improvements
Scripted BML Server Background

• Middleware functions don’t change
  – Mapping BML to JC3IEDM and push/pull to database
  – Program these once and get them right

• Interpreted WS offers flexibility
  – Rapid implementation of new BML constructs
  – Easy to modify underlying data model
    • MIP standard also continues to change
  – Reduces time and cost for prototyping
  – Scripting language provides a concise definition of BML-to-data model mappings
  – Although bugs still happen, the number of possible mistakes is far smaller

• Scripted operation may, however, be slower
  – Multithreading helps this
  – But a hard-coded implementation is likely to perform better
The Old Way: IDEF1x Mapping Definition

JBML mapping to JC3IEDM

Schema field <What>

Schema Reference:
<xsd:complexType name="CommandType">
  <xsd:sequence>
    <xsd:element name="What" type="GroundBMLWhatType"/>
    ...
  </xsd:sequence>
</xsd:complexType>

Where content-category-code is set to ‘ORDER’ (‘ORDER’)

reference
reference-id
content-category-code

action-reference-association
  action-id (FK)
  reference-id (FK)
  action-reference-association-index
  category-code

action
  action-id
  category-code

action-task
  action-task-id (FK)
  category-code
  activity-code

The category-code value will be set to ‘Is directed by’ (‘ISDSCR’)

The category-code value will be set to ‘ACTION-TASK’ (‘ACTTA’)

1

Used as the <OrderId>

2

category-code is set to ‘ORDER’ (‘ORD’)

Not machine readable though highly structured
Script is a concise XML coding of this
Scripted BML WS Design

- Basic operations: *push* and *pull*
  - Currently, servers for SQL and RI databases
  - Scripts implement BML Orders and Reports

- Script defines implementation of Business Objects (constituents of the higher-level BML grammar) over the JC3I EDM data model
  - BO is an XML subtree rooted at a defined node in the XML file – can invoke other BO

- Interpreter uses two files plus WS input
  - Mapping file contains script
  - BML schema file provides necessary context
  - XML namespace capable
Scripted BML WS Configuration

Two implementations: MySQL and SIMCI RI
SBML Enhancements

• Condensed Scripting Language (CSL)
• Pushing a Complete Thought in JC3IEDM
• BML Namespaces
• Multithreaded Operation
• Logging/Replay
• SISO C-BML Implementation
• OPORD and NATO OPORD
• Publish/Subscribe Dynamic Topics
• RESTful service and multi-language interface
Condensed Scripting Language

- Coding the script in XML makes parsing simple
- But XML is verbose and thus hard to read
- We’ve defined a condensed format which is isomorphic to the XML script and thus simple to translate
  - Intended to invoke business objects that produce a “complete thought” in JC3IEDM
  - We expect the result will be more modular as well as much more readable
  - So we needed a mechanism to make this work
Where CSL Is Used

BML Client → Input → Tomcat / Axis

Response → BML Services

Input: BML Client → CSL Script → CSL Translator

Output: BML Services → JC3IEDM SQL, JC3IEDM Java Objects

Mapping Script → BML Schema
Pushing a Complete Thought in JC3IEDM

• MIP provides recommended usage for JC3IEDM
  – “A database update or query must constitute a complete logical military thought.”
  • JC3IEDM 3.0.2 Annex O. 5/14/2009

• We understand this to mean all the data about a business object (composite) should be completed at the same time
  – Don’t push incomplete data to database

• We’ve added terms in scripting language to define a complete thought and an SBML mechanism to consolidate the push
  – ri_start and ri_end
Condensed Scripting Language Example

```plaintext
BOTransaction TaskeeWhoPush (task_act_id) ()
{
    ri_start Unit unit_id;
    GET unit unit_id (formal_abbrd_name_txt = UnitID);
    PUT act_res (
        act_id = task_act_id,
        act_res_index = act_res_index, cat_code = "RI",
        authorising_org_id = unit_id);
    PUT act_res_item (
        act_id = task_act_id,
        act_res_index = act_res_index,
        obj_item_id = unit_id);
    ri_end;
    BOReturn
    {
        BOReturnElement
        {
        Tag Result "OK";
        }
    }
}
Polling vs Publish/Subscribe

• “Pure” Web Service is always accessed by *push* or *pull* transaction from client
  – No provision for server to initiate action
• For clients to stay up to date they must pull latest status from server at rate determined by their need for up-to-date information (called *polling*)
  – Result: MSG-048 server in 2008 spent most of its time responding to status pulls
• Publish/subscribe gets around this by letting clients identify the categories of information they need – they subscribe to *Topics*
  – Server sends them a copy of every update associated with each subscribe Topic
  – More timely updates and a dramatic reduction in overhead
Publish/Subscribe Architecture
Publish/Subscribe Dynamic Topics

- Topic assignment
  - MSG-048 experimentation was the largest BML coalition to date, so we kept it simple
  - Topics were chosen in advance and coded in server
- Dynamic topics
  - More powerful approach allows Topics to be assigned at runtime
  - Implemented using JMS Message Selectors working through a single static Topic
  - Client defines interest using XPATH
Message Selectors for Dynamic Topics
Sample msgSelectors XML

<?xml version="1.0" encoding="UTF-8"?>
<Message>
    <Selector>
        <name>allGSR</name>
        <search>//TypeOfReport[. = 'GeneralStatusReport']
    </search>
</Selector>
    <Selector>
        <name>allOrder</name>
        <search>//OrderPush</search>
    </Selector>
    <Selector>
        <name>allSIMCI</name>
        <search>/*[contains(name(),'REP')]*/</search>
    </Selector>
</Message>
Java Code to add msgSelector

```java
import edu.gmu.c4i.sbmlclientlib.SBMLClient;

// create a client to the SBMLserver webservice
SBMLClient sbmlClient = new SBMLClient(host);

// add a new Message Selector
String s = null;
try {
    s = "//newwho:ListWho";
    String selectorName = sbmlClient.addMsgSelector(s);
    System.out.println("added msg selector "+ selectorName);
}
catch (Exception e) {
    System.out.println("Unable to add Message Selector "+ s + " " + e.getMessage());
}
```
BML Namespaces

• SBML must parse XML input
  – Both BML itself and script are XML
  – Should comply with W3C specifications

• This was hard to do, so deferred initially
  – Now we have completed it
  – Done by mapping BML to various schemas from which it is assembled
  – Allows validation of BML/XML with namespace

• Existing scripts have been modified to use namespaces correctly
Namespace Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<BusinessObjectInput …
 <!-- Define URI to prefix mapping for namespaces -->
 <Namespace>
   <uri>http://netlab.gmu.edu/IBML</uri>
   <prefix>bml</prefix>
 </Namespace>
 <Namespace>
   <uri>http://netlab.gmu.edu/JBML/BML</uri>
   <prefix>newwho</prefix>
 </Namespace>
 <Namespace>
   <uri>http://netlab.gmu.edu/JBML/MSDL</uri>
   <prefix>msdl</prefix>
 </Namespace>
 <Namespace>
   <uri>urn:int:nato:standard:mip:jc3iedm:3.1a:oo:2.0</uri>
   <prefix>jc3iedm</prefix>
 </Namespace>
</BusinessObjectInput>
```

```csharp
BOTTransaction LowerOrderPush()
{
    Assign [bml:OrderID] OrderIDwv;
    IfThen (OrderIDwv EQ "")
    {
        Abort OrderID does not exist in the input file;
    }
    GET ACT act_id (name_txt EQ [bml:OrderID]);
    Assign act_id order_act_id;
    IfThen (act_id NE "")
    {
        Abort OrderID already exists;
    }
    GET UNIT unit_id assignTo=taskerWhoUnitID
    (formal_abbrd_name_txt EQ [bml:TaskerWho/bml:UnitID]);
    IfThen (taskerWhoUnitID EQ "")
    {
        Abort Invalid or Absent TaskerWho in Order;
    }
...
Multithreaded Operation in SBML

• A known disadvantage of XML is its verbosity
• Results in a lot of network traffic and contributes to performance problems
  – Parsing and SOAP processing also take time
• When message volume is high, this can be offset somewhat by multithreading
  – SBML designed to support experiments not production
  – But even in experimental environment performance may be needed
  – So we’ve revised SBML code to work multithreaded
• Latest achievement:
  – Over 10 Reports/second with 8 processors on lab server
  – Expect this to scale to at least 16 processors
Logging/Replay in SBML

- Logging/replay is very useful in development and experimentation
  - Allows exact review and comparison of results
  - May be used to repeat input sequences for testing
- Initial SBML had only console log
  - Could capture transactions but they were difficult to extract
- We’ve added input/output logging to SBMLv2.4
  - And a replay client to regenerate traffic
- We expect this functionality will grow as users find ways it should be enhanced
RESTful Services

• Representational State Transfer (REST)
  – More efficient because it does not use SOAP
  – Our measurements indicate 15% improvement

• Client Language flexibility
  • JBoss supports SOAP and RESTful messaging
  • RESTful supports any subscriber that has access to a HTTP client library
  • This will avoid need to use Java Native Interface intermediary on C++ clients
HornetQ REST Interface

- SBML Server (SOAP)
- SBML Server (RESTful)
- JMS
- HornetQ REST Interface
- Java Subscriber (using JMS)
- Subscriber (HTTP HEAD and POSTs)
SBML Summary

• Scripted BML WS served well as development tool for MSG-048 and SIMCI
  – Enables developing reliable services more rapidly
  – Open source http://netlab.gmu.edu/OpenBML
  – Offered Reference Implementation for SISO C-BML

• Enhancements make SBML more useful
  – Complete Thought enhances modularity
  – Namespace implementation complies with W3C
  – Multithreaded operation 10X (or more) performance
  – Logging/replay improves utility
  – Publish/Subscribe now has dynamic Topics
  – Includes more efficient RESTful services that also support a range of client programming languages
SBML Application
MSG-048 2009 Architecture
Multiple SBML Servers in MSG-048

• US C2 and Simulation systems in MSG-048 came from SIMCI project which used the Java JC3IEDM RI that was not available to other MSG-048 participants
  – All other systems used the database implementation
• We built a “back-to-back client” to pass BML reports between from RI-based SBML to MSG-048 SBML
SISO C-BML Implementation

• SISO has released Phase 1 Trial Use Draft Schema and mappings
  – Standard composite “FiveWs”
  – Abstracted and Light versions

• GMU has implemented CompositesLight portion of the schema in SBML push/pull
  – With IBML Order/Report from MSG-048
  – With JC3IEDM database
  – Also full schema as push/pull of XML documents (not mapped to JC3IEDM)
OPORD Schemas for SBML

• Major application of BML is OPerations ORDer
• Recent Army CIO/G6 project supported detailed BML architecture
  – See http://c4i.gmu.edu/BML
• This included a five-paragraph OPORD based on earlier work done for AGC
  – Significantly more detailed than MSG-048 schema
  – Also posted SBML script – see webpage
• SIMCI project converted this to NATO OPORD
  – Will include SISO C-BML CompositesLight compliant script for NATO OPORD
Battle Management Language
Command and Control Graphical User Interface
(BMLC2GUI)
C2LG GUI

• Command & Control Lexical Grammar (C2LG) Graphical User Interface (GUI) was constructed by the German research center FGAN (now part of Fraunhofer FKIE)
• The C2LG GUI was created to generate “pure” BML statements that were valid grammar statements.
• In many BML activities, the C2LG GUI was used as an “integration hub” to take the input from C2 systems and construct a “valid” JBML Order that could be sent and ingested by different nation’s simulations.
• FGAN operates under rules that would not allow them to release the C2LG GUI to the whole BML community.
C2LG GUI
BML C2 GUI

- Patterned after Fraunhofer-FKIE C2LG GUI
  - Usable as editor or monitor
  - Reads/writes Orders and Reports
  - Auto-configures to any BML schema
  - View and modify a BML-XML file
  - Map/image display shows 2525B icons from XML
  - Future version will enter geolocation data in BML-XML file
  - Open source at http://c4i.gmu.edu/BML
BML C2 GUI: ORDER
BML C2 GUI : CONTROL FEATURES
JaxFront – Open Source XML Java Editing

• JAXFront architecture

BML C2 GUI: Report with 2525B Icon
BML C2 GUI : Position Status Report
SBML Services in the BMLC2GUI

• The BMLC2GUI uses the Web Services maintained by SBMLServer through the SBMLClient application.
  – CallListWho: is used by the GUI to bring up 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>

• The BMLC2GUI is open source - could be modified to use other source of Unit information
SBML Services in the BMLC2GUI

- **GetLatestReportIDs**: is used by the GUI to build a list of report information:

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

- **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.

  ```xml
  <ReportPull>
  <ReportID>410</ReportID>
  </ReportPull>
  ```
JaxFront – Open Source XML Java Editing

• 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”.
• Web site http://www.jaxfront.org
• The BML C2 GUI uses the Free Community version (Open-Source) of JaxFront as a major component for editing BML documents.
• The user can dynamically generate GUIs that allow the user to edit XML data without being exposed to the underlying XML technology.
BMLC2GUI Forms Creation

• The core of the BMLC2GUI is the use of JAXFront’s libraries to build a new and a customized type of XML Document Object Module (DOM) that can be rendered as Java Swing objects.

• The JaxFront’s DOM Builder takes the following parameters to generate a JAXFront document:
  – The XML document.
  – The XUI (optional): XML User Interface file used to customized the view of the XML document
  – The XML root node
BMLC2GUI Geospatial Aspect

• After the successful rendering of the XML document, we start extracting the geospatial information (Latitude, Longitude coordinates representing positions or dimensions of objects) on the map.

• We parse the document and pass the elements and values to OpenMap’s MapHandler

• The MapHandler draws and controls the following types of layers:
  – Country or area of interest geospatial data layers. In our case (ESRI shape files).
  – BML objects and geospatial information: unit, minefield, bridge, spot, track.
  – MIL-STD-2525b symbols
Open Map

• Open Source JavaBeans based Geospatial development tool.
• From BBN Technologies, now part of Raytheon.  
  http://openmap.bbn.com
• BMLC2GUI is using the latest version of OpenMap 4.6.5, released March 5, 2009.
• It provides various capabilities to allow users to see and manipulate geospatial information.
• OpenMap supports various map data file formats.
Open Map

• BMLC2GUI uses ESRI (commercial) file format.
  – .shp “shape” files
• The BMLC2GUI uses OpenMap to display the different data layers on the map in addition to drawing BML objects, units and control measures at their corresponding locations.
• Users already are finding ways to enhance our interface
Open Map - MIL-STD-2525B

- The BMLC2GUI makes use of the OpenMap’s implementation of MIL-STD-2525B symbols
- The unit/object symbol is constructed from the UnitID / objectType during the geospatial information extraction.
- The corresponding unit/object symbol is drawn at the Lat-Lon coordinates
- Objects can be minefield, bridge, spot,…
BML C2 GUI Functionality Summary

- Editing a BML Document
- Serialization of a BML Document
- Validation of a BML Document
- Pulling a BML Document
- Pushing a BML Document
- Retrieving Latest Reports
- Surrogate C2 System
C2 Capability

- Primary use of BMC2GUI is to support development
- Also can be used to generate orders (but not rapidly)
- Implements the SBMLSubscriber Client application to connect to the subscription service of the SBMLServer
- This enables the GUI to listen to any published 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
  - Displayed with appropriate geospatial information
  - Can provide a simple track
- There is a lot of room for enhancement by the community
BMLC2GUI Summary

• Provides an easy-to-use, comprehensive tool for the BML end user
• Platform-independent and command-line free BML editing and viewing
• Geospatial capabilities
• Validation and Serialization
• Being open-source makes it less expensive to own and operate
  – And supports BML community enhancements
• Customization and enhancements possible by the BML community
• Open source at http://c4i.gmu.edu/BML
Conclusions

• Infrastructure software development is unglamorous but absolutely essential
• Developing a new approach such as SBML is only the start
• Research on the BML paradigm is not practical without mature, robust supporting software
• This work is enabling research at system level to understand merits of coalition BML