**Web Service Middleware for BML**

Dr. Mark Pullen  
mpullen@gmu.edu

---

**Background**

- **GMU-ACS-VMASC team developed DMSO XBML**
  - Starting point Army BML prototype (CAPES-OTB)
  - Added Web Services and C2IEDM to Ground system
  - Worked with Gestalt to demonstrate Ground-Air operation, adding TBMCS and JSAF
  - Successfully linked with French APLET system using CAPES-JSAF over WS/C2IEDM

- **GMU-ACS-VMASC-JATTL(Gestalt)-NPS funded for JBML phase 1 under Army/TEC (Sep 06 – Jun 07)**
  - Support SISO C-BML standard effort with inputs & open source code
  - Ground-Air-Maritime BML proof of principle
  - Support NATO MSG-048 process
Slide 3

**JBML Vision**

**C2 Domain Language(s)**
- Ground BML
- Maritime BML
- Air BML
- geoBML
- Logistics BML
- Peacekeeping BML
- Crisis Management BML
- ...  

**Modeling and Simulation Systems**

**JC3IEDM representation**

---

Slide 4

**JBML Definition**

- BML - an unambiguous language to:
  - Command and control live and simulated forces conducting military operations, and
  - Provide for situational awareness and a shared, common operational picture.

*Shared Semantics between C2 and M&S via a Common Tasking Description*
C2IEDM / JC3IEDM* and BML

- BML represents C2 orders and status in a way that is:
  - Unambiguous, so it can be processed by computers
  - Human understandable, so it can be read and expressed by users
- JC3IEDM represents C2 data in a way that is concise and detailed
  - Can be extended to represent any needed C2 data
  - Can express the content of any BML message
  - But is virtually impossible for humans to read and comprehend quickly
  - Also, experience shows different groups find different, incompatible ways to express a given set of data in JC3IEDM

* We will treat these names as interchangeable in this presentation

---

JBML and Grammar

- BML projects to date (even XBML) have focused on point-to-point interfaces
  - Later ones have been implemented in XML/SOAP
  - But there has been no overall design to create a general, flexible BML that can grow to encompass the whole projected scope of BML
- We need a rational and extensible approach to expressing BML information
  - C-BML Study Group recommended that the future standard be based on a lexical grammar
  - We intend to get a start on this approach
Hieb and Schade BML Grammar

- As a first step toward rationalizing BML, Ulrich Schade and Mike Hieb analyzed the existing ground BML.
- They concluded that any statement in ground BML can be expressed using the construct:

\[ B \rightarrow \text{Verb Tasker Taskee (Affected | Action)} \]

Where Start-When (End-When) Why Label (Mod)*

- “Where” is a “location phrase”;
- the “When”’s are “time phrases”;
- “Why” gives the purpose of the action;
- “Label” is a label given to the task in order allow it to be referred in other basic expressions.

- This grammar is easy to represent in XML
  - For example the Tasker is tagged <TaskerWho>

JBML Web Service Infrastructure

- Currently Web Services are the communication means of choice for distributed system development.
- BML can be passed between C2 and simulation Web services in two ways:
  - Synchronous connection: direct real-time path between C2 and Simulation, pairwise
  - Asynchronous connection: BML messages are buffered in one or more intermediate systems that store system state

- Synchronous is more responsive but asynchronous is more flexible
  - One repository serves multiple C2 and Simulation systems
  - Not necessary for all systems to be continuously available
  - Particularly good for development, experimentation, and distributed operation where it is hard to keep all systems responding constantly
  - Not necessary for all systems to be continuously available
  - Particularly good for development, experimentation, and distributed operation where it is hard to keep all systems responding constantly
Slide 9

**BML Demo Environment Architecture**

Slide 10

**JBML Web Service Infrastructure**

- Using a Web service repository will simplify JBML and C-BML development
  - Enables distributed development over Internet
  - Each participant can test interactions asynchronously
  - The same BML interfaces can be used synchronously later
- To support a standards effort, the Web service software needs to be fully exposed
  - Serves as Reference Implementation, available as open source for all to understand
  - Ideally, configurable by standards developers without creation of new software
  - This was our target in creating the JBML Service Architecture
- This does not imply that future systems that comply with the standard must use a repository
  - They could use synchronous interfaces
**Domain-Configured Service**

**Domain Layer**

- The Domain-Configured Service (DCS) implements BML in a domain context.
  - In the case of an OPORD, the transaction at this layer would specify all information about a given task (i.e., who, what, when, where, and why).
  - For a Position Report*, the transaction at this layer would include all information about the updated location (e.g. who, where, when-valid, precision, and etc)
- The DCS is designed to serve as infrastructure for a growing BML standards effort.
- It is built to be configurable by an XML Schema Definition (XSD) and based on the layering concepts VMASC has put forward in various papers
- Also supports Poll command to retrieve an order

*JBML Phase 2
Foundation of BML-DCS

- Implemented in the Document-Literal XML mode
- Generic Web service that driven by an XML schema called Domain Knowledge Schema (DKS)
  - Basis for standard to define all possible C-BML orders
- Based on the concept of Schade and Hieb:
  - the entire existing BML grammar can be described in terms of a few primitives
    - `<command>`  `<tasker-who>`  `<taskee-who>`
    - `<affected-who>`  `<what>` (action)  `<where>`
    - `<start-when>`  `<end-when>`  `<why>`
    - `<label>`  `<modifier>`
- JBML is defining an XML format that can be used by every BML order based on these tags plus a small number of extensions
- The current version of the schema implements the tags above

BML Base Service
Composite Layer

- The BML Base Service (BBS) provides primitive BML elements such as Who, What, When, Where, and Why.
  - Other elements may be introduced for new and existing BML domains as required.
  - The BBS accesses all of the database tables affected by the primitive, through the Common Data Access software.
  - We believe the standard at this layer should identify the database tables to be updated for each BML primitive and the validation conditions to be applied.
  - This avoids the confusion inherent in multiple implementers taking different interpretations of the C2IEDM/JC3IEDM
- BBS Interfaces:
  - Upwards: XML-based API interface (4) to the DCS.
  - Downwards: uses the BML Common Data Access API.
  - In order to expose the BBS externally, access will be provided through the DCS using a “pass through” tag.
BML Common Data Access Service

Atomic Layer

- The CDAS provides mechanisms to:
  - inspect database tables externally
  - both read and update the database tables directly
- The CDAS will support inspection of every database table used in any domain of BML, in order to support understanding of system behavior during development.
  - Open-source GUI available on the JBML website
  - Changes to the database do not overwrite the previous values but instead mark them invalid and provide new valid values.
- There are two higher level interfaces to the CDAS:
  - Internal interface (6), defined as an XML-based API
    - active in both directions (write and read)
  - External interface (5) defined using a WSDL (XML/SOAP based)
    - One way interface intended to be used for inspecting (reading) tables
    - Lower level interface (7) is SQL based.

CDAS Implementation

- GMU has implemented the CDAS in open source Java code, based on original VMASC development
  - Available on JBML website
- Amounts to a wrapper that translates get and put of tables to SQL
  - A single module supports any number of tables
  - Provides error control
  - Filters to limit access to those tables intended for BML use
  - Automatically moves updated table rows to a backup database, forming history of data state
- Also implemented a simple Java-based GUI that allows inspection of the C2IEDM / JC3IEDM tables
JBML Web Service Infrastructure

- Currently Web Services are the communication means of choice for distributed system development
- BML can be passed between C2 and simulation Web services in two ways:
  - Synchronous connection: direct real-time path between C2 and Simulation, pairwise
  - Asynchronous connection: BML messages are buffered in one or more intermediate system that store(s) system state
- Synchronous is more responsive but asynchronous is more flexible
  - One repository serves multiple C2 and Simulation systems
  - Not necessary for all systems to be continuously responsive
  - Particularly good for development, experimentation, and distributed operation where it is hard to keep all systems responding constantly

---

JBML Web Service Infrastructure

- Using a Web service repository simplifies JBML and C-BML development
  - Enables distributed development over Internet
  - Each participant can test interactions asynchronously
  - The same BML interfaces can be used synchronously later
- To support a standards effort, the Web service software needs to be fully exposed
  - Serves as Reference Implementation, available as open source for all to understand
  - Ideally, configurable by standards developers without creation of new software
- DCSOrderPush, DCSOrderPull
  - Top-level definition of Joint order C2 and Simulation services

- OrderTypes
  - Defines Joint operations order consisting of multiple Air, Ground, and Maritime commands

- AirTypes, GroundTypes, MaritimeTypes
  - Define domain-specific information

- FiveWTypes
  - Defines common Who / What / When / Where / Why etc.

- msdlTypes
  - Describes reusable MSDL schema from their webpage
  - This approach implements SISO guidance to make MSDL and C-BML interoperable

---

**XSD OrderType**

```xml
<xsd:complexType name="OrderType">
  <xsd:sequence>
    <xsd:element name="CommandersIntent" type="FreeTextType" minOccurs="0"/>
    <xsd:element name="Command" type="CommandType" maxOccurs="unbounded"/>
    <xsd:element name="OrderIssuedWhen" type="WhenType"/>
    <xsd:element name="OrderID" type="OrderIDType"/>
    <xsd:element name="TaskerWho" type="WhoType"/>
    <xsd:element name="TaskOrganization" type="msdl:TaskOrgType" minOccurs="0"/>
    <xsd:element name="ControlMeasures" type="MultipleControlMeasuresType" minOccurs="0"/>
    <xsd:element name="Reference" type="ReferenceType" minOccurs="0"/>
    <xsd:element name="Declassification" type="DeclassificationType" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```
XSD Joint CommandType

```xml
<xsd:complexType name="CommandType">
    <xsd:choice>
        <xsd:element name="GroundCommand" type="GroundCommandType" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="AirCommand" type="AirCommandType" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="MaritimeCommand" type="MaritimeCommandType" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:choice>
</xsd:complexType>
```

XSD GroundCommandType

```xml
<xsd:complexType name="GroundCommandType">
    <xsd:sequence>
        <xsd:element name="TaskeeWho" type="WhoType"/>
        <xsd:element name="What" type="GroundBMLWhatType"/>
        <xsd:element name="Where" type="WhereType"/>
        <xsd:element name="StartWhen" type="WhenType"/>
        <xsd:element name="EndWhen" type="WhenType" minOccurs="0" maxOccurs="0"/>
        <xsd:element name="AffectedWho" type="WhoType" minOccurs="0" maxOccurs="0"/>
        <xsd:element name="Why" type="GroundWhyType" minOccurs="0" maxOccurs="0"/>
        <xsd:element name="Label" type="LabelType"/>
    </xsd:sequence>
</xsd:complexType>
```
• A major JBML deliverable is a set of candidate drafts for C-BML Phase I standard
• C-BML has committed to use the Joint Command, Control and Consultation Information Exchange Data Model (JC3IEDM) to represent C-BML data
• JBML uses a middle level (composite layer) service to access the tables of JC3IEDM database as logical transactions
  – For example, Who/What/When/Where/Why
  – One of these may require multiple table accesses to push or pull the attributes of associated objects
• In the following slides we introduce our preferred format to document this critical aspect of our deliverable

Our diagrams are represented in IDEF1x:
• Entity is a box
  – For database, contains relational table
  – Top line is its name
  – Key attributes, foreign keys (FK) are contained in the box
• Line ending in solid circle shows a child entity
• Line with open circle and crossed by another line shows an incomplete set of subcategories of the entity
• We have added balloons and step numbers to clarify
**Slide 25**

**JBML mapping to JC3IEDM**

**Schema field <OrderId>*

**Schema Reference:**

```xml
<xsd:complexType name="OrderType">
    <xsd:sequence>
        ...
        <xsd:element name="OrderId" type="OrderIdType"></xsd:element>
        ...
    </xsd:sequence>
</xsd:complexType>
```

**Slide 26**

**JBML mapping to JC3IEDM**

**Schema field <TaskerWho>*

**Schema Reference:**

```xml
<xsd:complexType name="OrderType">
    <xsd:sequence>
        ...
        <xsd:element name="TaskerWho" type="WhoType"></xsd:element>
        ...
    </xsd:sequence>
</xsd:complexType>
```
Conclusions

- Web services are proving to be a powerful tool for distributed system development
  - Net-centric and GIG compatible
  - Facilitates rapid development of complex systems
- The JBML web services enabled such a development for Phase I
  - Ground, Air and Maritime
  - Coherent principles and architecture planned to support future development
  - Open source infrastructure for SISO and NMSG
- Outlook: continued expansion of BML
  - With Web services as a cornerstone technology

Slide 27
Center of Excellence in C4I – George Mason University