Supporting NATO C2-Simulation Experimentation with Scripted Web Services

Dr. Mark Pullen and Lisa Nicklas
George Mason University C4I Center
{mpullen,lnicklas}@c4i.gmu.edu

Presentation Overview

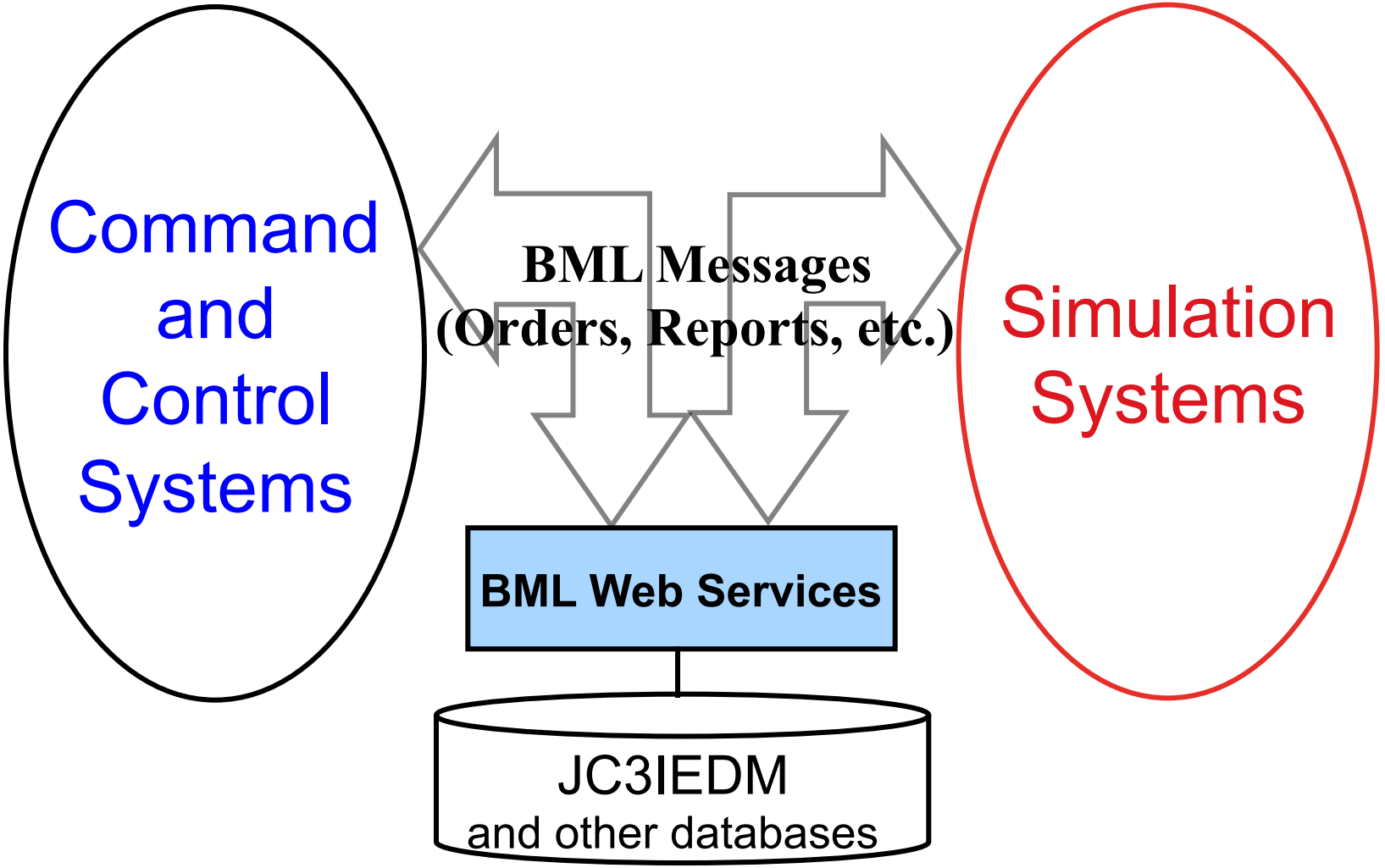
- Scripted BML background
- Scripted BML in NATO MSG-048
- Architecture of SBMLServer
- Publish/Subscribe for BML
- Recent improvements to SBML
- Conclusions

Scripted BML Background

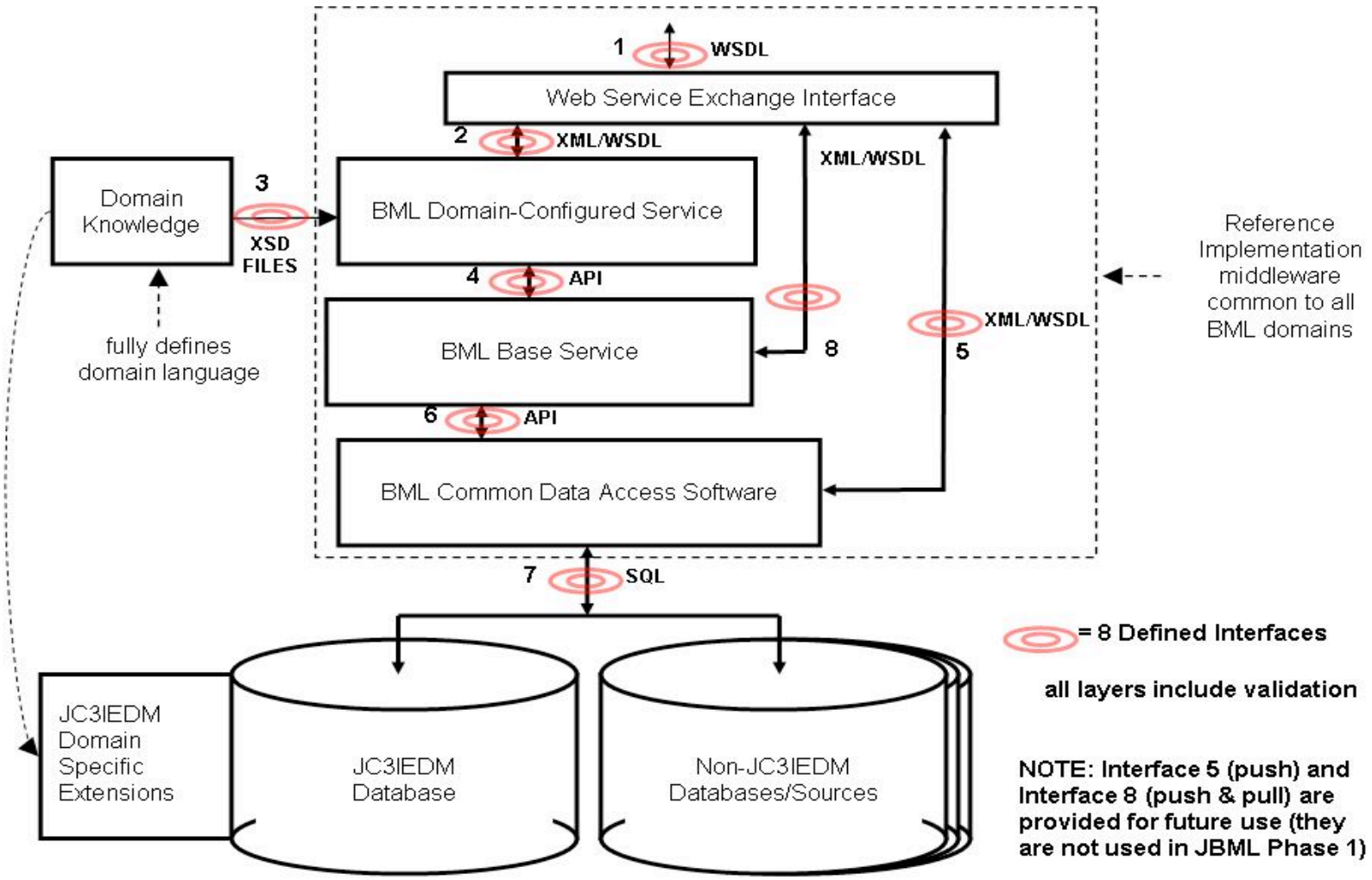
BML Purpose and Operation

- 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



Original BML WS Architecture



Why Scripted WS

- 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

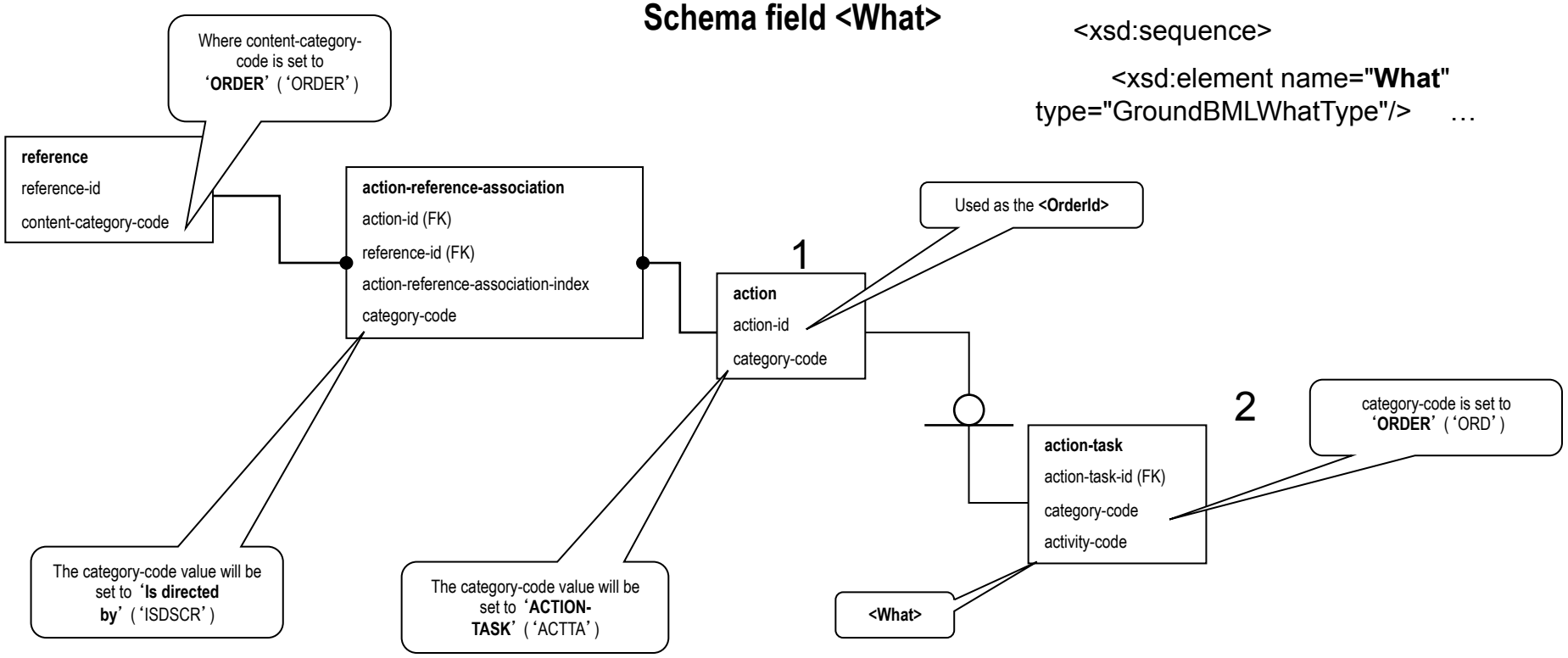
Schema field <What>

Schema Reference:

<xsd:complexType name="CommandType">

<xsd:sequence>

<xsd:element name="**What**" type="GroundBMLWhatType"/> …



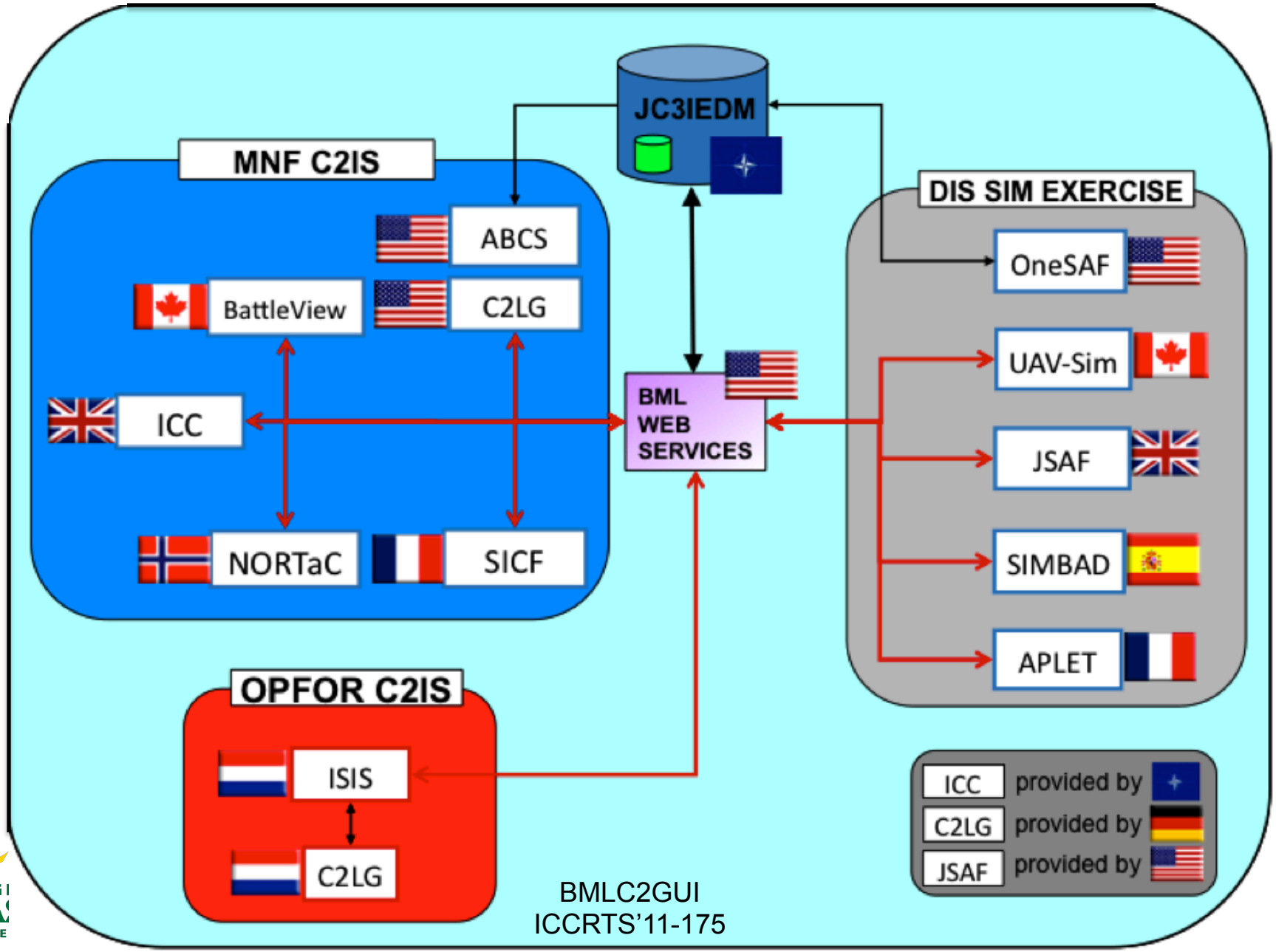
Not machine readable though highly structured

Script is a concise XML coding of this

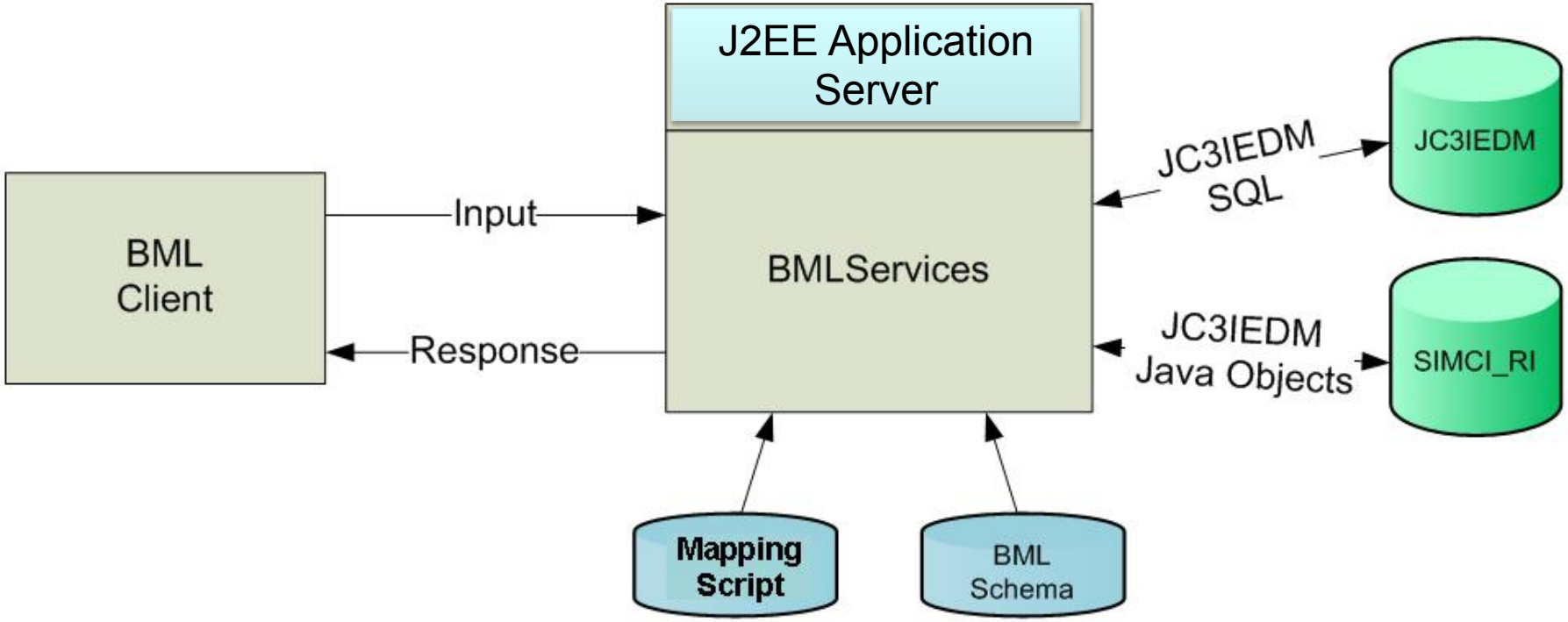
Scripted BML WS Design

- 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 JC3IEDM 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

MSG-048 2009 Architecture



Scripted BML WS Configuration

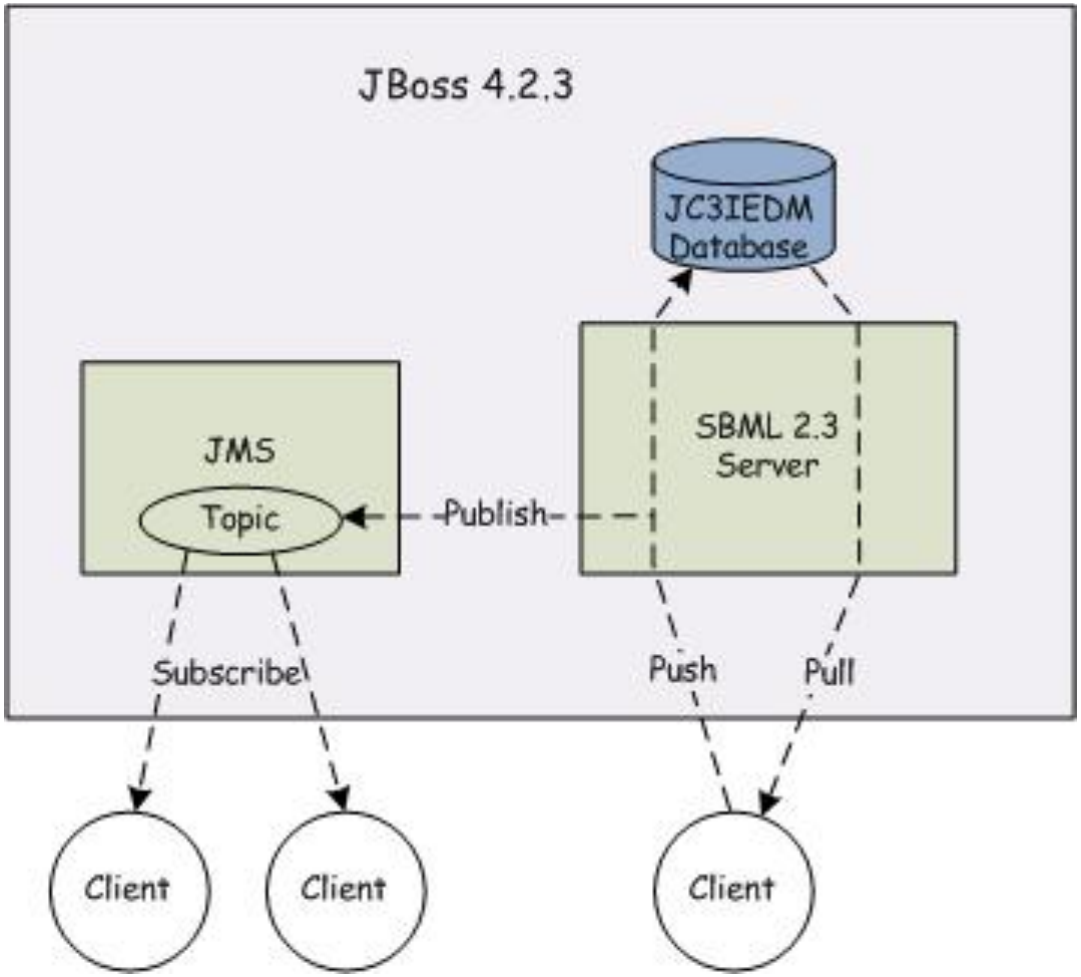


Two implementations: MySQL and SIMCI RI

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



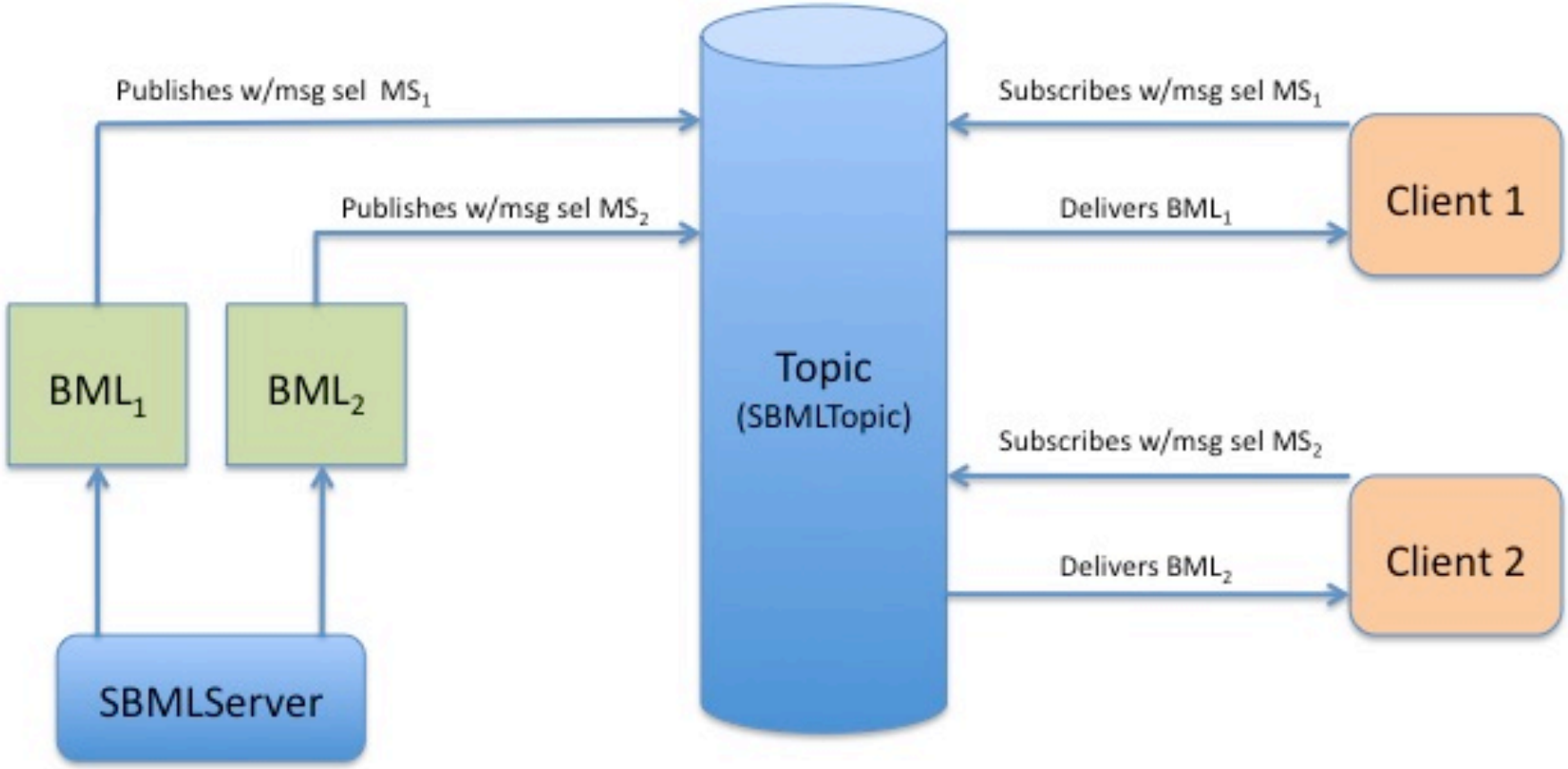
Advantages of Publish/Subscribe

- Avoids inefficiencies:
  - Server must re-read information written to database
  - Redundant polling
  - Separate server cycle needed for each client
- Implements effective distribution
  - Create a Topic for each interest category
  - Clients subscribe by Topic
  - Server automatically forwards transactions matching the Topics
- However, our implementation of publish/ subscribe used by MSG-048 has static topics

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>
```

Recent Improvements to SBMLServer

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

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

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:
  - About 10 Reports/second with 8 processors on lab server
  - Would expect this to scale to at least 16 processors

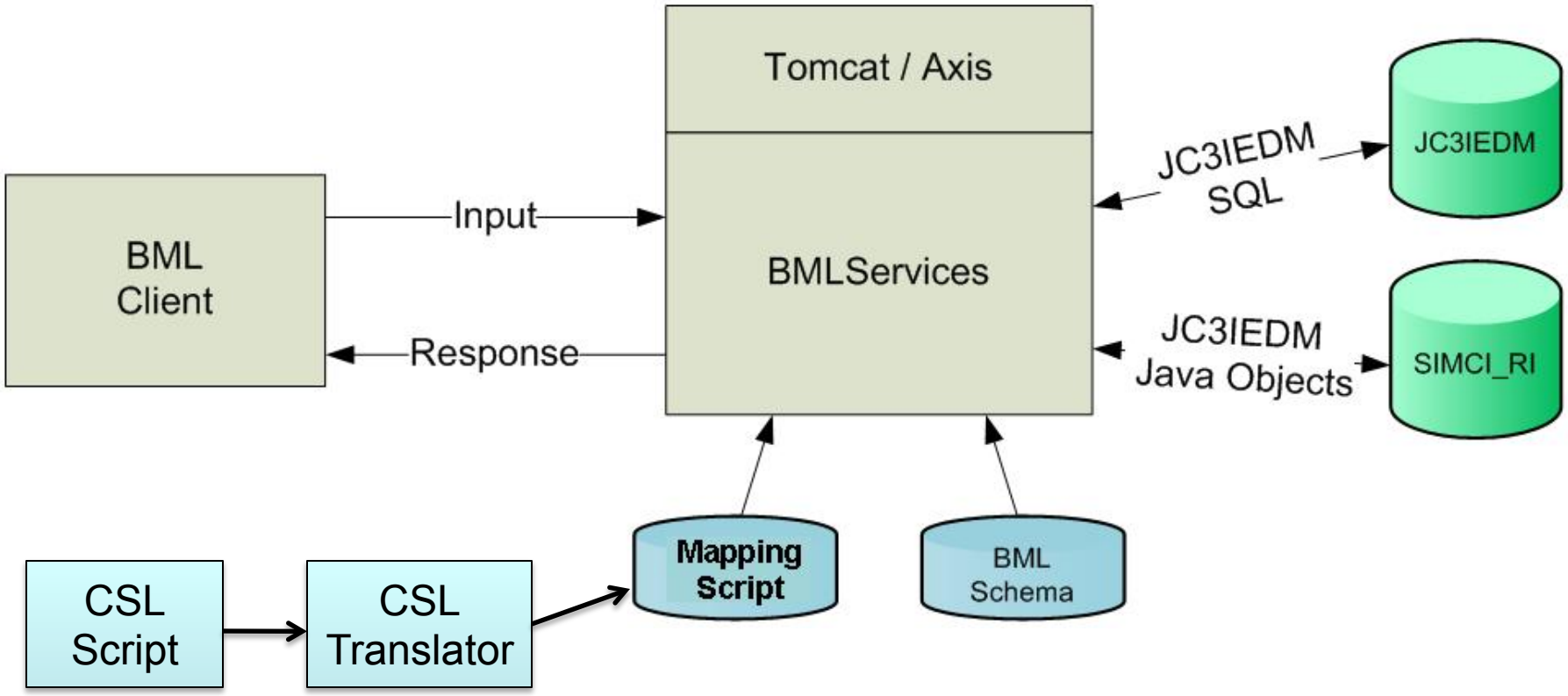
Logging/Replay in SBML

- 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

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
  - The result is more modular as well as much more readable
  - So we needed a mechanism to make this work

Where CSL Fits in SBML



Condensed Scripting Language Example
part one

```
 BOInput
{
  BOTransaction WhatWhenPush(...
  {
    //fragment from WhatWhenPush
    Call TaskeeWhoPush TaskeeWho (task_act_id) () ;
  ...
  }
```

Condensed Scripting Language Example

part two

```
BOTransaction TaskeeWhoPush (task_act_id) ()
{
   GET unit unit_id (formal_abbrd_name_txt EQ UnitID);
   PUT act_res (
act_id EQ task_act_id,
act_res_index EQI act_res_index, cat_code EQ "RI",
authorising_org_id EQ unit_id) ;
   PUT act_res_item (
act_id EQ task_act_id,
act_res_index EQ act_res_index,
      obj_item_id EQ unit_id ) ;
   BOReturn
   {
      BOReturnElement
      {
          Tag Result "OK";
       }
   }
}
```

OPORD Schemas for SBML

- 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
- Ongoing SIMCI converting this to NATO OPORD
  - We have provided SISO C-BML Light compliant script for NATO OPORD
  - Also SISO C-BML Full but without JC3IEDM support

RESTful Services

- Representational State Transfer (REST)
  - More efficient because it does not use SOAP
  - Our measurements indicate 15% improvement
- Client Language flexibility
  - Jboss supports both 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

Conclusions

- Scripted BML WS served well as development tool for NATO MSG-048
  - Enables developing reliable services more rapidly
  - Open source http://netlab.gmu.edu/OpenBML
  - Offered Reference Implementation for SISO C-BML
- Many improvements, inspired by NATO experimentation needs, have made SBML Server more useful and robust
- We look forward to continued improvements supporting NATO MSG-085