Work Beyond XAL at FRIB

Paul Chu / Vasu Vuppala
Outline

- Libraries built on top of XAL
- XAL in the big (HLA) picture
- Service-Oriented Architecture (SOA)
  - Data persistence
- XAL utilities and tools
- Prepare for future
Many modules can be reused if packaged properly
- Should not be final GUI applications but callable APIs
- Modules can be called in services

SLAC example

1. All under one CVS module
2. Use ant to manage build
3. "Top level tag for official release" i.e. slaclib.jar
4. Individual module can be built for specific version (e.g. Message-RO-5-3.jar) and use classpath to override slaclib.jar
High-Level Application Architecture

- Hardware layer
- Control system interface
- Persistence layer
- API layer – XAL, solver…
- Application layer
- Other modules, e.g. multi-particle tracking code
- Components reusability
  - If properly architected
  - Can last much longer
- Integrated systems but can be deployed separately
Service-Oriented Architecture

- Arrange applications’ functions as services
- Thin client suitable for mobile devices
- Data persistence, e.g. RDB
- Computing power
- Service management
- Reusability
- Easy deployment
- Beyond SOA
  - Services can live in Cloud

Facility for Rare Isotope Beams
U.S. Department of Energy Office of Science
Michigan State University

P. Chu / V. Vuppala, Dec 14, 2012, Slide 5
• A standard data persistence mechanism should be provided, or every institute has to come up with own database for XAL data

• Standard data persistence can be accessed via Data API

• (RESTful) Data Service can build on top of the Data API

• Data Service can be a Cloud Computing Service

• In relational database
  • Lattice schema
  • Model schema
  • Save/restore schema
  • Physics data schema
    » Scan application data
    » Experiment data
- Initial beam conditions as Model data
- All element related attributes are in property/value pairs so they are general enough for all accelerators
- XAL specific tables
  - RF Cavity table (containing RF accelerating gaps)
  - Transit-Time Factor (TTF) table for RF cavities
FRIB Data Upload

- A standard Excel file with multiple tabs
- All DB access via API (based on JPA)
- Device Settings or optimized lattice with settings w/ physics names
- Name mapping between the above 2
- Consolidate to 1 master file (Super Spreadsheet)
Prototype RDB access with Java Persistence API (JPA)
• New ORM Included in JDK 7

RDB connection info set in persistence.xml

Entity classes generated by NetBeans 7 IDE

Scripting in JYTHON for RDB access is easy

NetBeans has visual editor for Java GUI with JPA

```python
import sys
from javax.annotation import Resource
from javax.persistence import EntityManager
from javax.persistence import EntityManagerFactory
from javax.persistence import PersistenceUnit
from javax.persistence import Persistence
from mydbtest import *

emf = Persistence.createEntityManagerFactory("fdbdevPU")
em = emf.createEntityManager()
userTransaction = em.getTransaction()

date = Date(112, 3, 3, 15, 10, 0)
newModel = Models()
# set comments
newModel.setComments("Test persistence")
# set date

try:
    userTransaction.begin()
    # enter data to DB
    em.persist(newModel)
    # commit change
    userTransaction.commit()
except Exception, e:
    print "Got an error" + str(e)
```
Database API and XAL API should be similar, if not identical
- Security package
- GUI widgets – use JavaFX?
- Data plotting package
XAL setup should be much easier
- A standard spreadsheet template should be sufficient
- Database generated
- Spreadsheet data uploaded to DB
- XAL files generated from DB

Third party solver/optimizer as service

Third party modeling tools, e.g. FEL model, multi-particle tracking

Everything should be Web compatible
Cloud Computing

- Advantages
  - No capital IT cost – pay per use
  - Failure as a feature – recover seamlessly
  - Expandability

- Services
  - Storage-as-a-service
  - Database-as-a-service
  - Information-as-a-service
  - Process-as-a-service
  - Application-as-a-service
  - Platform-as-a-service
  - Integration-as-a-service
  - Security-as-a-service
  - Management/governance-as-a-service
  - Testing-as-a-service
  - Infrastructure-as-a-service

“Cloud Computing and SOA Convergence in Your Enterprise”
D. S. Linthicum
Controls Database Collaboration

- Requirements
- Proteus: Configuration
  - Graphical User Interface
  - Application Programming Interface
    » Matlab, Python, Java
- Proteus: Naming System
- Current Status
- Integration
- In Development
## CDB Requirements 1

<table>
<thead>
<tr>
<th>#</th>
<th>Database</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Component</td>
<td>Physical and logical information about the machine and its component systems</td>
</tr>
<tr>
<td>2</td>
<td>Infrastructure</td>
<td>Cables, IOC, Racks, Rooms etc</td>
</tr>
<tr>
<td>3</td>
<td>Lattice</td>
<td>Location, length, setting and/or signal records</td>
</tr>
<tr>
<td>4</td>
<td>Model</td>
<td>Physics model</td>
</tr>
<tr>
<td>5</td>
<td>Logbook</td>
<td>Logbook entries</td>
</tr>
<tr>
<td>6</td>
<td>State</td>
<td>Save/restore state of FRIB segments</td>
</tr>
<tr>
<td>7</td>
<td>Alarm</td>
<td>Set changes, set/read mismatches</td>
</tr>
<tr>
<td>8</td>
<td>Inventory</td>
<td>Spare parts, stock items</td>
</tr>
</tbody>
</table>
## CDB Requirements 2

<table>
<thead>
<tr>
<th>#</th>
<th>Database</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Physics</td>
<td>Results from physics experiments</td>
</tr>
<tr>
<td>10</td>
<td>Traveler</td>
<td>Production, test, design data</td>
</tr>
<tr>
<td>11</td>
<td>Maintenance</td>
<td>Preventive maintenance data, failure analysis, lifetime analysis</td>
</tr>
<tr>
<td>12</td>
<td>Operations</td>
<td>Beam statistics, run hours, beam on target, shift summary, downtime, bypass records</td>
</tr>
<tr>
<td>13</td>
<td>MPS</td>
<td>Machine state dumps</td>
</tr>
<tr>
<td>14</td>
<td>FRIB Requirements</td>
<td>Parameter list, system and component requirements</td>
</tr>
<tr>
<td>15</td>
<td>Interlocks</td>
<td>Interlock hierarchy information [optional]</td>
</tr>
</tbody>
</table>
Application Architecture

- **Application layer**
  - Operator interfaces
  - High-level applications
  - Libraries

- **Service layer**
  - Access to data
  - Programming Interface

- **Data layer**
  - Managed data
  - Instrument data
  - No direct access
CDB Schema Domains

class Domain Model

Online Model
- Code
- Element
- twiss

Controls
- PV
- PVgroup

Behavioral

Structural

Infrastructure
- Cable
- Class1
- IOC

Component
- Logical
- Physical
- Properties
- Type

Configuration
- Control Hierarchy
- Extended Lattice
- Housing Hierarchy
- Power Hierarchy

Functional

eLog
eTraveler
Inventory
Physics Apps
Save-Restore
MPS
Collaboration Goals

- Be Like EPICS for Data Services
  - Easy to Install
  - Easy to Configure
  - Extensible
  - Well-Defined Interfaces
  - Documentation
  - Training

- Domain Goals
Status - Summary

- Logbook: In production at FRIB and BNL
- eTraveler: In production at FRIB
- Machine Configuration Tool: In production at FRIB
- Save/Restore: In production at BNL
- Model: Under development at FRIB and IHEP
- Signals: Under development at FRIB and BNL
- Physics Applications: Under development at FRIB
- Authentication: Under development at FRIB
Partners

- Brookhaven National Lab, New York, USA
- Facility for Rare Isotope Beam, Michigan, USA
- Institute for High Energy Physics, Beijing, China
Demo

- **Proteus: Configuration**
  - Graphical User Interface
  - Programmatic Interface

- **Proteus: Naming System**
  - Graphical User Interface

- **URL:** [http://controls.frib.msu.edu](http://controls.frib.msu.edu)
  - Under “Applications” Menu
Integration

- **E-Traveler**
  - Devices from Configuration Database (In Production)

- **E-Log**
  - Logbook Entries for a Component (Under Development)

- **Valid Name**
  - Naming System Checks for Validity and Already Assigned Names (Under Development)

- **Online Model**
  - Machine Configuration as Input to XAL (Under Development)
Under Development

- Integration with Control System
  - Real-Time Signal Values

- More Data
  - Properties, Alignment, Measurements, CAD Drawings, Location, Cables
  - Etc.

- EPICS V4 Interface
  - pvget, eget

- Graphical Hierarchy

- Comprehensive Search

- GUI Improvements
Development Approach

- **Module**
  - Database
  - Module Applications
  - Module Services
  - Module API

- **User Applications**
Development Approach - Interfaces

- Service API
- Language Binding API
- Data API
- Module API
Suggested Development Strategy

1. Schema
2. LB API
3. Data API (optional)
4. LB API Switches to Data API
5. Service API
6. LB API Switches to Service API
EPICS V4

- Next Version of EPICS
- Expands to Non Real Time Data
- Pvget, Eget
- Directory Service
- Channel Finder
- http://epics-pvdata.sourceforge.net/
Technologies & Infrastructure

- **FRIB:**
  - MySQL
  - REST, EPICS V4
  - Java
    - JPA, EJB, JAX-RS
  - PHP, Python
  - Mercurial, Git

- **BNL:**
  - Python
    - Django
  - Java
  - Mercurial, Git

- **Infrastructure**
  - Sourceforge
  - Google+
  - Launchpad
Challenges

- Differing Deadlines, Priorities, Technologies, Processes, Terminologies
- Integration
- Making it a Product
  - Generalization
  - Packaging
  - Documentation
- Data Migration
- Performance
- User Interfaces
- Not a Typical Project
Conclusion

- Collaboration started in November 2011
- Weekly Meeting
- Quarterly Review
- Goals, High-Level Requirements, Architecture Defined
- Some applications in production, some in development, and some not initiated
- Several challenges
- Need more collaborators