Beamline Control Module

Daron Chabot
Canadian Light Source

EPICS Collaboration Meeting 2008
Padova, Italy

Outline

- Beamline Control Module (BCM) overview
- Motivating factors
- BCM design and implementation
    - Device model and data acquisition (DAQ) engine
- Future plans

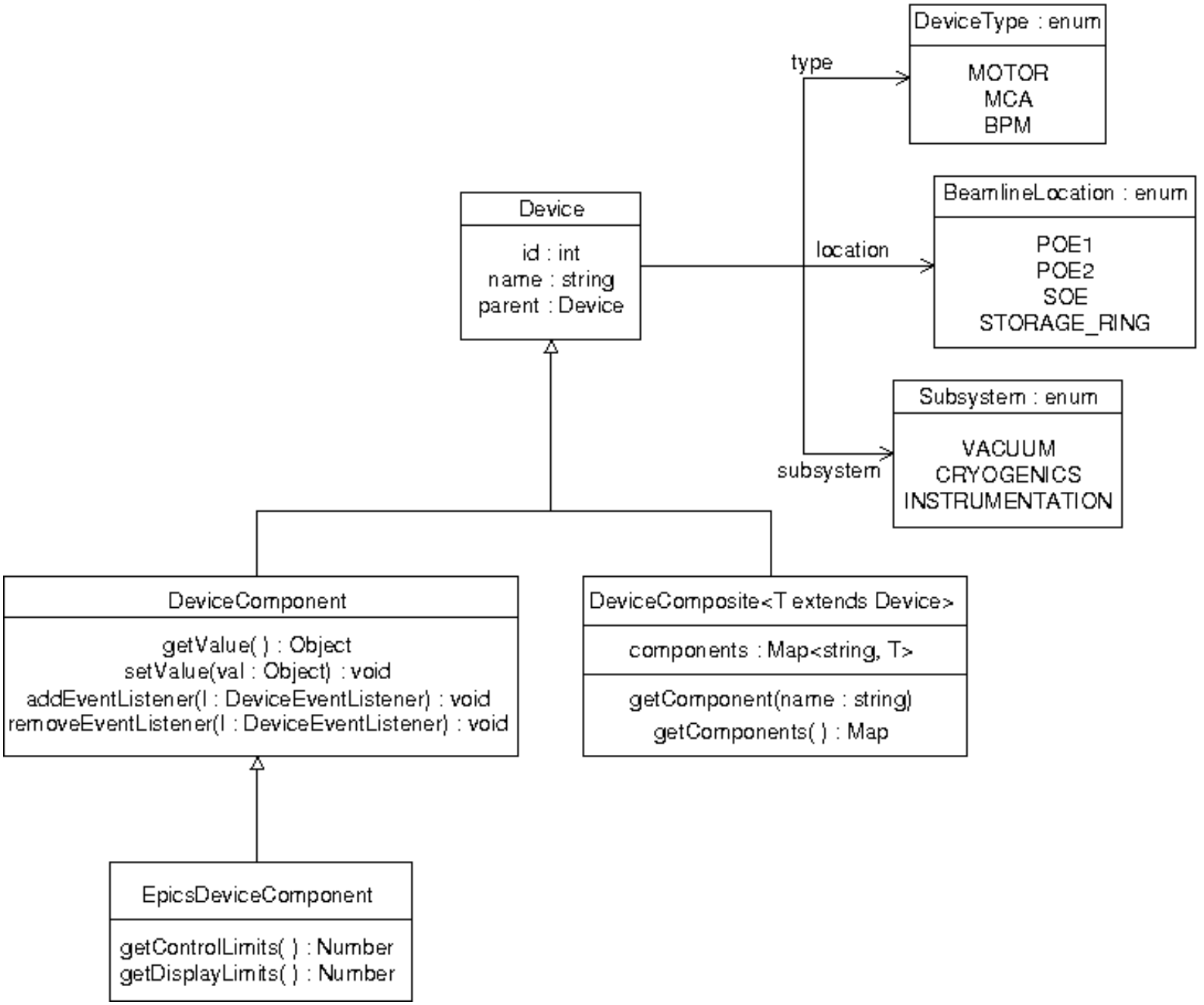
BCM Overview

- Java framework featuring:
  - Configuration-based approach
  - Device abstraction layer (metadata) and libraries
  - Data acquisition (DAQ) engine
- Prototype implementation dependencies:
  - Channel Access for Java (CAJ)
  - Log4j
  - Spring framework
  - JMS (ActiveMQ)

Motivation

- Web-access project (RBA/Science Studio) would benefit from hardware abstraction layer
    - Needed the “M” in the MVC design pattern
- Implies that solution needs to “play well” with J2EE-type applications
- Lack of existing basis to start from

BCM Design: Device Model



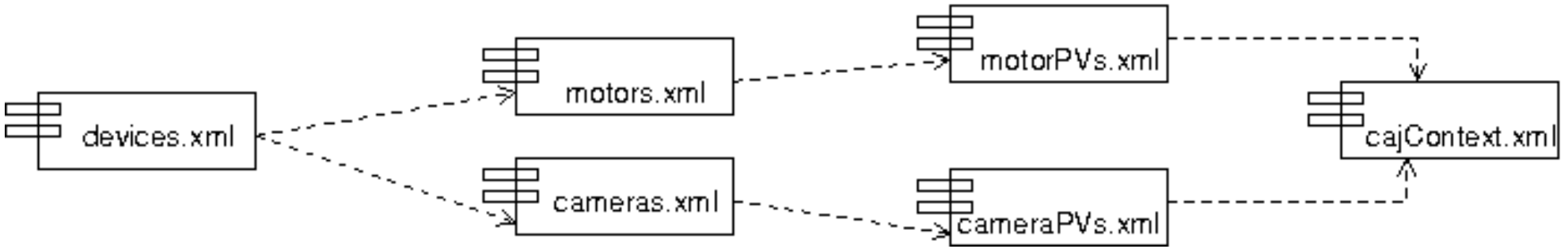
BCM Design: Device Lib

- Collection of ~12 (so far!) device interfaces
- Also includes implementations based on the EpicsDeviceComponent class
  - e.g. IMotor, IBpm, IMultiChannelAnalyzer, ICamera, etc.

```
public interface IMotor extends ICalibration {

    public double getPosition();
    public void setPosition(double aValue);
    public double getSpeed();
    public void setSpeed(double aValue);
    public double getAcceleration();
    public void setAcceleration(double aValue);
    public void moveAbsolute(double aValue);
    public void moveRelative(double aValue);
    public void stop();
    public String getState();
    public boolean isMoving();
}
```

BCM: Configuration

- Configuration is a hierarchy of XML files
- Spring instantiates Devices from XML descriptions of objects through a DeviceFactory (interface)
- So-called Inversion of Control (the other IOC)
    - Also known as Dependency Injection



motors.xml

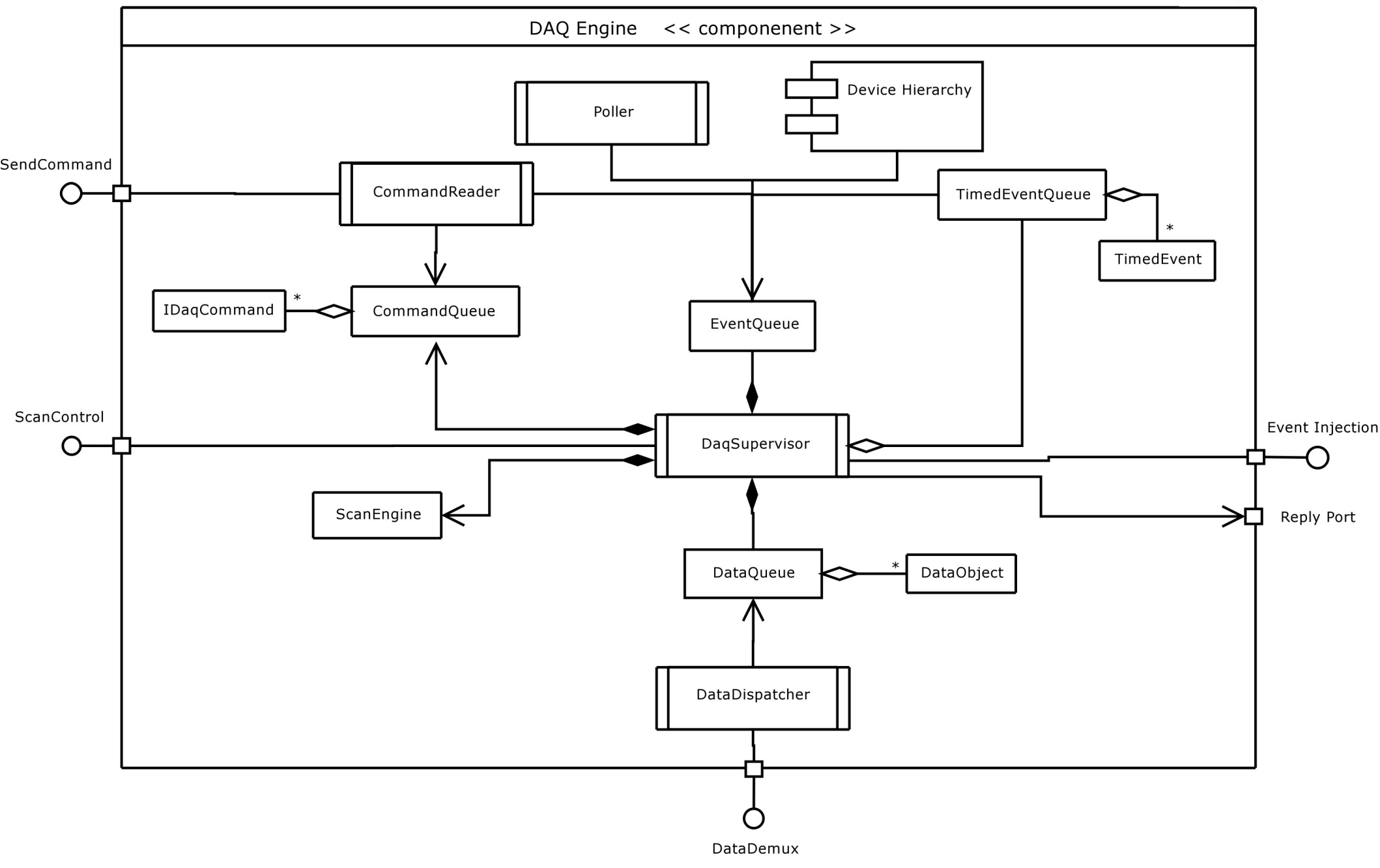
```
<import resource="pvsMotor.xml"/>

<bean id="testMotor" class="ca.lightsource.bcm.idevice.impl.Motor">
    <constructor-arg type="java.lang.String" value="SMTR16071B1001"/>
    <constructor-arg>
        <map>
            <entry key="description" value-ref="SMTR16071B1001_DESC"/>
            <entry key="positionFbk" value-ref="SMTR16071B1001_STEP_SP"/>
            <entry key="moveToSetpt" value-ref="SMTR16071B1001_STEP"/>
            <entry key="positionSetpt" value-ref="SMTR16071B1001_ST_SETPOSN"/>
            <entry key="moveRelSetpt" value-ref="SMTR16071B1001_STEP_REL"/>
            <entry key="velocityFbk" value-ref="SMTR16071B1001_VELO_FBK"/>
            <entry key="velocitySetpt" value-ref="SMTR16071B1001_VELO"/>
            <entry key="accelerationFbk" value-ref="SMTR16071B1001_ACCEL"/>
            <entry key="accelerationSetpt" value-ref="SMTR16071B1001_ACCEL_SP"/>
            <entry key="stop" value-ref="SMTR16071B1001_STOP"/>
            <entry key="status" value-ref="SMTR16071B1001_STATUS"/>
            <entry key="calibDone" value-ref="SMTR16071B1001_CALIBDONE"/>
            <entry key="calibRun" value-ref="SMTR16071B1001_CALIBRUN"/>
            <entry key="calibLog" value-ref="SMTR16071B1001_CALIBLOG"/>
        </map>
    </constructor-arg>
    <property name="location" value="POE"/>
</bean>
```

motorPVs.xml

```
<import resource="cajContext.xml"/>

<!-- pvs for the motor, SMTR16071B1001 -->
<bean id="SMTR16071B1001_DESC" class="ca.lightsource.bcm.device.impl.epics.EpicsDevice">
    <constructor-arg value="SMTR16071B1001:desc"/>
    <constructor-arg ref="cajContext"/>
</bean>

<bean id="SMTR16071B1001_STEP_SP" class="ca.lightsource.bcm.device.impl.epics.EpicsDevice">
    <constructor-arg value="SMTR16071B1001:step:sp"/>
    <constructor-arg ref="cajContext"/>
</bean>

<bean id="SMTR16071B1001_STEP" class="ca.lightsource.bcm.device.impl.epics.EpicsDevice">
    <constructor-arg value="SMTR16071B1001:step"/>
    <constructor-arg ref="cajContext"/>
</bean>
```

DAQ Engine

- Scan control interface
  - Start, stop, pause, resume data acquisition
- Configurably responds to:
  - Periodic events (“do X every Y seconds”)
  - Polling events (“do X as often as possible”)
  - Device events (i.e. EPICS monitors, alarms, disconnects)
  - Command events (scan control, custom commands,etc)
- Event responses (callbacks) are added via XML specification (similar to Device config files)


DAQ Engine << componenent >>
Poller
Device Hierarchy
SendCommand
CommandReader
TimedEventQueue
*
TimedEvent
IDaqCommand
*
CommandQueue
EventQueue
ScanControl
DaqSupervisor
Event Injection
ScanEngine
Reply Port
DataQueue
*
DataObject
DataDispatcher
DataDemux

BCM: Future Plans

- Source config info from RDB (replace Spring)
- Create beamline-specific scan-types (templates): PX screening,MAD/SAD scans,etc
- Add to device library (interface and impl)
- Rich-client GUI to complement web UI
    - A Beamline “Explorer” with DAQ config and control