HIGH LEVEL APPLICATIONS
OPEN XAL ACCELERATOR HIERARCHY

Day 1

Accelerator View for the Physicist/Developer

Outline

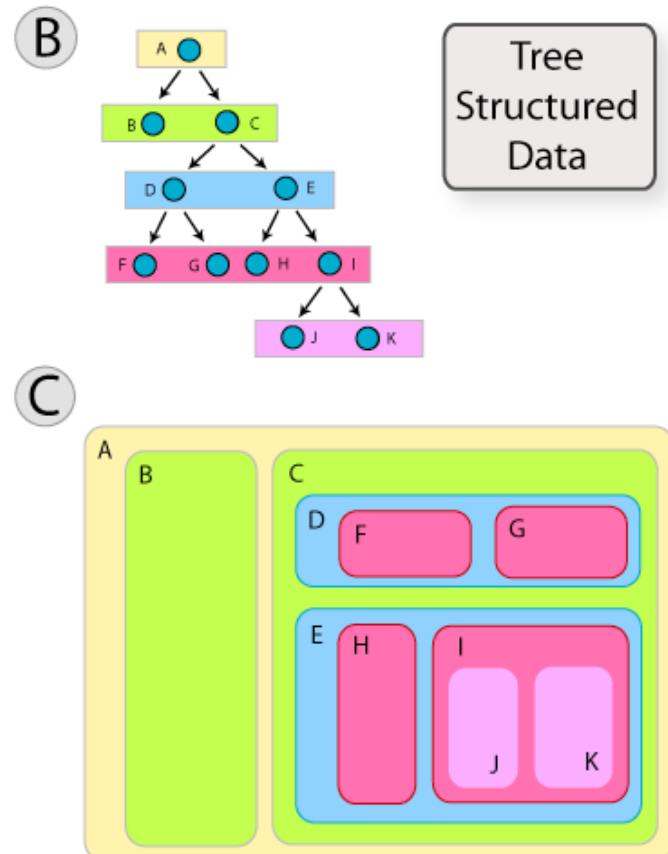
- Tree Data Structure – Representing the Accelerator
- Accelerator Components
  - Accelerator
  - Sequences
  - Nodes
- Specific Nodes
  - Power Supplies
  - Magnets
  - BPMs

All throughout code excerpts are given to demonstrate applications

Tree Structure Data Representation

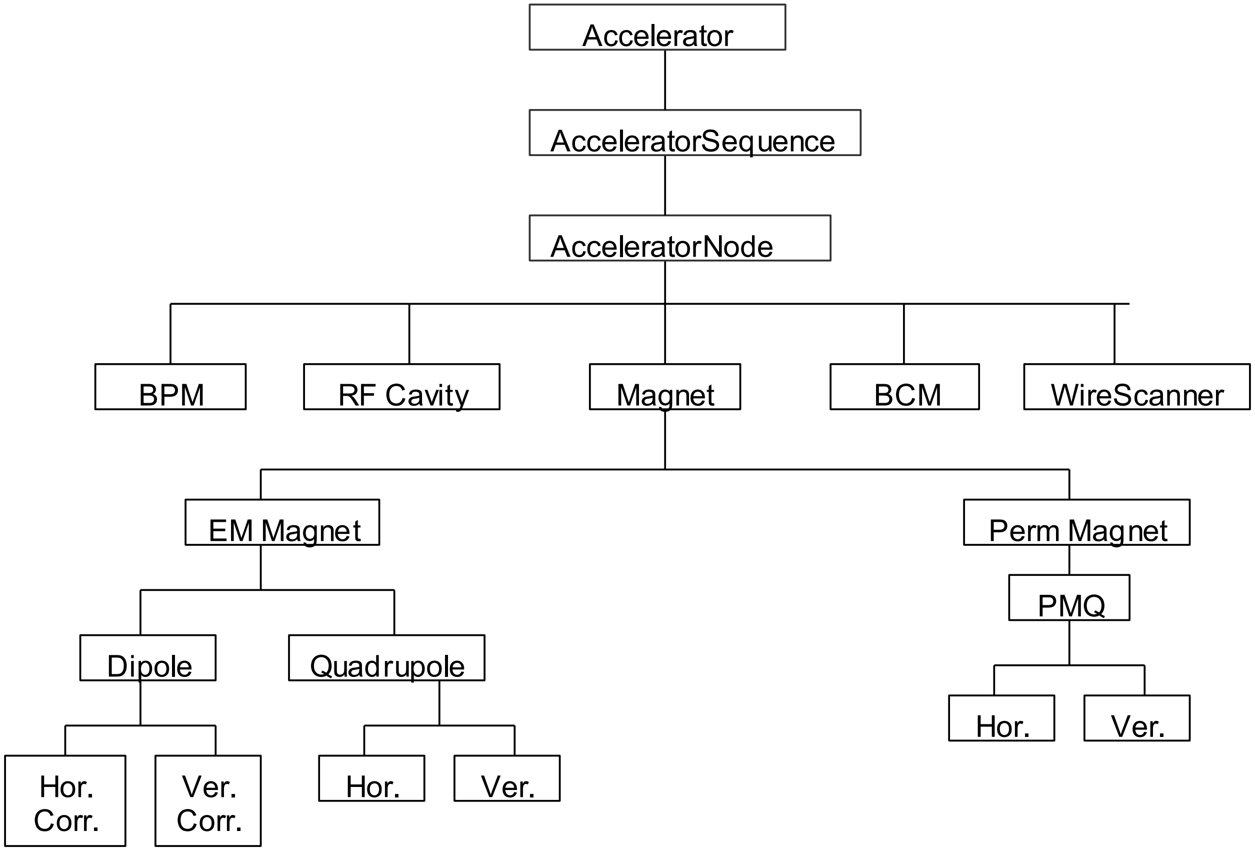
A

id	parent_id
A	NULL
B	A
C	A
D	C
E	C
F	D
G	D
H	E
I	E
J	I
K	I



Commonly used structure in computer science

Accelerator Hierarchy



- Accelerator hierarchy from the accelerator physicist point-of-view

Accelerator Hierarchies
Advantages

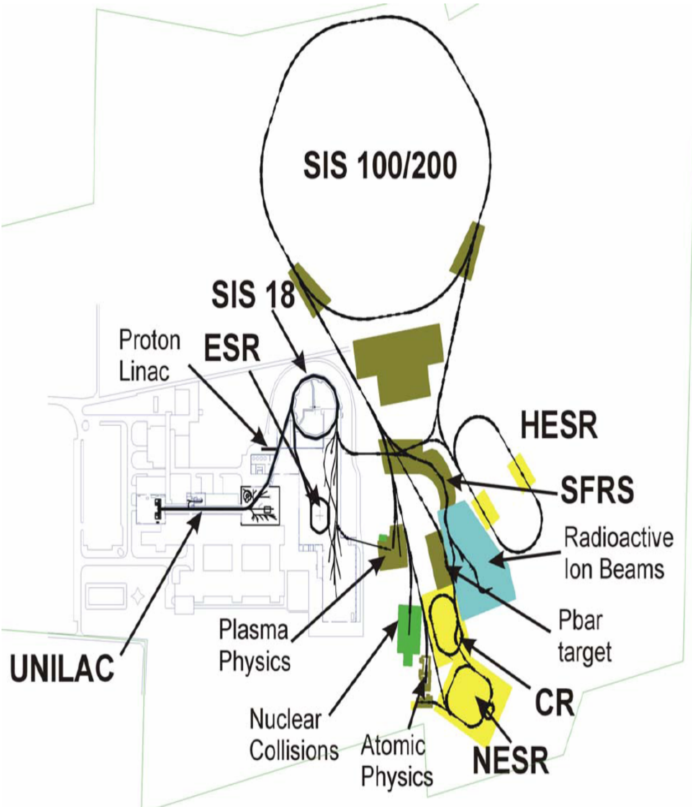
- Straightforward to create flexible, robust applications that can applied to multiple parts of the accelerator
  - Valuable for commissioning and accelerator modification in that applications naturally respond to changes in the accelerator
- Allows writing applications by function/task
  - As opposed to creating many specific applications, each pertinent to only one special part of the accelerator
  - Operation is on parts of a tree, without knowing *a priori* the structure of the tree

Accelerator Hierarchies

- The concept of using an Accelerator hierarchy is not unique:
- UAL (Java - http://www.ual.bnl.gov)
  - Online and offline accelerator modeling
  - Used at BNL
- CDeV (http://www.jlab.org/cdev/)
  - Developed at JLab
  - Interface to control system
- MAD 8 (http://hansg.home.cern.ch/hansg/mad/mad8/mad8.html)
  - MAD with C++ Class structure
  - Used for optics modeling
- LEGO
  - SLAC (beamline modeling, C++)
- Many others

We will use XAL as the vehicle for exploring accelerator class structures – see JavaDoc

Accelerator Tree



- The accelerator can be a big mess – a collection of many different linear accelerators, transport lines, rings, etc.

The Accelerator Object

Tree Root Note (gov.sns.xal.smf.Accelerator)

- The accelerator is the highest level interface to information you need
  - This object contains everything you want to know, but were afraid to ask, about the accelerator
- Typically this information is stored in a permanent, semi-static data source (e.g. a database).
  - At SNS the database schema does not directly reflect the XAL class structure – it must meet the needs of many other groups
- In XAL, we use an XML representation of the accelerator as the immediate source
  - The XML can be automatically generated from a database – recommended
  - This file reflects the class structure used in XAL
  - Details on the XML file specifics will be covered later

XML Accelerator Representation (To be covered in more detail later)

```
<sequence id="MEBT" len="3.633">
  <attributes>
    <sequence predecessors="RFQ"/>
  </attributes>
  <node type="marker" id="Begin_Of_MEBT" pos="0" len="0"/>
  <node type="QH" id="MEBT_Mag:QH01" pos=".128" len=".061" status="true">
    <attributes>
      <magnet len=".061" polarity="-1" dfltMagFld="-34.636"/>
      <align x="0.0" y="0.0" z="0.0" pitch="0" yaw="0" roll="0"/>
      <aperture shape="0" x=".016"/>
    </attributes>
    <ps main="MEBT_Mag:PS_QH01"/>
    <channelsuite name="magnetsuite">
      <channel handle="fieldRB" signal="MEBT_Mag:QH01:B" settable="false"/>
    </channelsuite>
  </node>
```

Accelerator Object
Retrieval in Open XAL

- From an XML file, use methods in the class

```
xal.smf.data.XMLDataManager
```

- If you have set a default accelerator using the optics switcher application:

```
Accelerator accel = XMLDataManager.loadDefaultAccelerator()
```

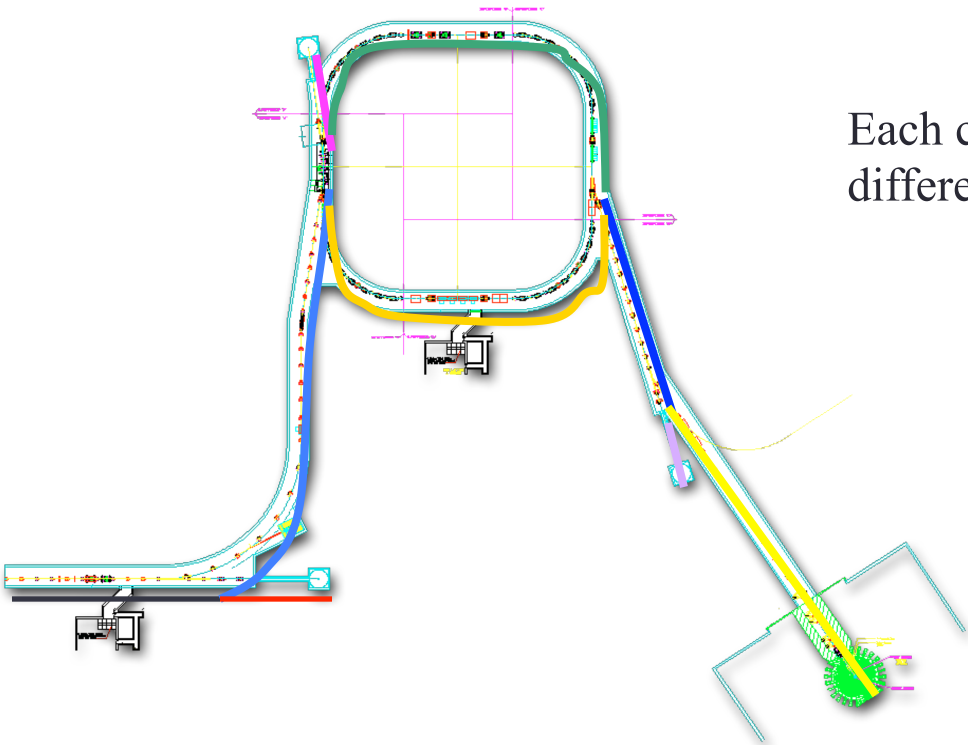
- Or manually read an accelerator from a file:

```
Accelerator accel = XMLDataManager.acceleratorWithPath(String strPath)
```

- And many other options

Accelerator Sequences

(gov.sns.xal.smf.AcceleratorSeq)



Each colored section can be a different sequence

- Sequences are contiguous sections of beamline that are logically related
- They are defined so that they may conveniently pasted together, e.g. whenever there is a “fork in the road”, create a new sequence
- Longer sequences can be composed as collections of connected sequences
- Sequences have names, lengths and allowed predecessors

Accelerator Sequences
Continued

- Sequences are the component that many applications work with
  - Display a beam trajectory through a part of the machine
  - Set the RF phase and amplitude for a Drift Tube Linac Tank
  - Make a bump in the beam trajectory in a section
- General purpose application will first select a sequence and then perform its function on this piece of accelerator
- See xal.samples.xalSeqs.py for example usage

Accelerator Sequences
Retrieval from Open XAL

- Get a single specified sequence you are interested in:

```
AcceleratorSeq seq = accel.getSequence("seqName")
```

- Or you can paste together a collection of sequences

```
# Make a list containing sequences we'd like to  paste
together

lst = ArrayList();
lst.add(lebt)
lst.add(rfq);
lst.add(mebt);
lst.add(dtl1);
#lst.add(dtl2);

# make the new "combo" sequence containing the stuff we
want:
newSeq = AcceleratorSeqCombo("testSeq", lst);
```

Accelerator Nodes
(xal.smf.AcceleratorNode)

- A node is an abstract representation of a beamline hardware element
  - Magnet, RF cavity, diagnostic device, …
- Accelerator Sequences are build up from Nodes
- Nodes are real pieces of equipment, typically in or near a beamline
  - No drift spaces are included – these are calculated later – stay tuned
- Drift spaces between pieces of equipment are NOT included
  - They are modeling elements
- Nodes have properties including name, distance from start of the sequence, status, etc.
- Location refers to the longitudinal center of the device
- Nodes can share at the same location
  - E.g. Quadrupole magnet with dipole corrector windings
- The same device can be in two sequences
- Nodes have methods that can be used to get/put information directly to/from the machine

Accelerator Nodes

Node Types (see base class xal.smf.AcceleratorSeq)

- Magnets – to affect the transverse dynamics of the beam
- RF cavities – to affect the longitudinal dynamics of the beam
- Beam Position Monitor (BPM) – to measure the transverse (and sometimes longitudinal) position of the beam
- Beam Current Monitor (BCM) – to measure the beam intensity
- See xal.samples.xalNodes.py for example usage

Accelerator Nodes

Open XAL Node Types used at SNS

- Bend
- BLM
- BPM
- CCL
- CurrentMonitor
- CvgGauge
- Dipole
- DTLTank
- Electromagnet
- ExtractionKicker
- GenericNode
- HDipoleCorr
- IonGauge
- Magnet
- MagnetMainSupply
- MagnetPowerSupply
- MagnetTrimSupply
- Marker
- NeutronDetector
- PermanentMagnet
- PermQuadrupole
- ProfileFit
- ProfileMonitor
- Quadrupole
- ReBuncher
- RfCavity
- RfGap
- RingBPM
- SCLCavity
- Sextupole
- Solenoid
- TrimmedQuadrupole
- Vacuum
- VDipoleCorr

Accelerator Nodes

Node Retrieval in Open XAL

- Node types have identification strings (e.g. "Q" for quad)
- Nodes can be selected from a sequence by type
  - quads = sequence.getNodesOfType("Q")
  - Magnets = sequence.getNodesOfType("magnet")
- You can use and create filters (and / or etc.)
  - Package xal.smf.impl.qualify
  - See xal.samples.Qualifier.py for usage examples
- Many nodes have convenience methods to directly perform operations
  - BPM – get beam position
  - Magnet – get magnetic field
  - Typically blocking actions

Accelerator Nodes

Node Connection to Signals

A Hardware Node has a "ChannelSuite"

- Keyed collection of control system channels associated with this node
- Channels facilitate network connections to real hardware objects

- Each channel in the suite is internally reference by an XAL *handle*,
  - The handle-channel binding is created in the XML file

- Thus if a control system channel changes, the XAL software does not

- Example:

```
<node type="DCH" id="DTL_Mag:DCH513" pos="3.345482" len=".0225" status="true">
    …
   <ps main="DTL_Mag:PS_DCH513"/>
  <channelsuite name="magnetsuite">
    <channel handle="fieldRB" signal="DTL_Mag:DCH513:B" settable="false"/>
  </channelsuite>
</node>
```

Accelerator Nodes

Retrieving a Channel from an Open XAL Node

- Sometimes it is useful to work directly with channels rather than use convenience methods
- Example

```
accelerator = XMLDataManager.loadDefaultAccelerator()
sequence = accelerator.getSequence(“DTL5”)
node = sequence.getNode(“DTL_Mag:DCH513”)
channel = node.getChannel(“fieldRB”)
value = channel.getValueRecord().doubleValue()
```

Accelerator Nodes

Magnet Nodes

- Magnets are the primary means of beam manipulation in the transverse plane in accelerators

- Magnet Optics
  - Dipoles for bending,
  - Quadrupoles for focusing,
  - Sextupoles for chromaticity correction

- Power Supply
  - Common supply – power a common lattice (e.g., FODO transport)
  - Single supply – typically matching quadrupoles, injection dipoles

- Some issues concerning morphology
  - Magnets are permanent or electromagnets
  - Main magnet – main physical structure
  - Corrector magnet – typically contained with in a main magnet
  - Trim magnets – contained within magnets on bulk supplies

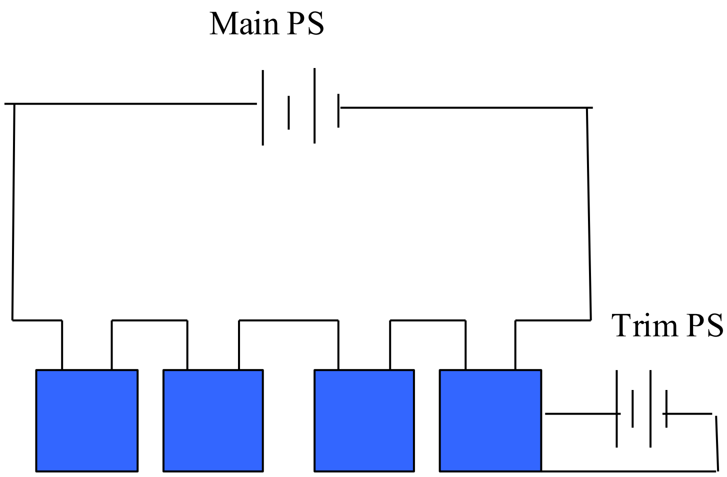
Accelerator Nodes
Magnet Nodes

- Methods for getting and setting field levels
  - Internal details of channel connection managed by Open XAL
  - getField() – returns the field
    - XAL uses MKS units , i.e. T/m^n, where n = 0 for dipole, 1 for quad, etc.
    - Invokes the $B(I)$ curves to set power supply current
  - setField(dblVal) – sets the magnet field strength to dblVal T/m^n
  - getCurrent() – gets magnet power supply current directly
  - setCurrent(dblCur) – set power supply strength directly to dblCur A

Power Supplies and Magnets

Real World Considerations

- A magnet produces a field(s) of a certain multipole(s)
- You adjust a power supply to change magnetic fields
- This may include a trim power supply
- Multiple magnets may be connected to a bulk power supply
- Power Supplies have a setpoints and readbacks
  - Setpoint is the output the user has specified
  - Readback is the actual output reading on the line
- Usually field readback is the proper value to use
- Power supplies have limits (e.g. max. current, max. field)



Accelerator Nodes
BPMs

- Beam Position Monitors (BPMs) return the beam position
  - Horizontal plane: getXAvg()
  - Vertical plane: getYAvg()
  - Longitudinal plane: getPhaseAvg()

- The Data
  - All control system connection details are hidden.
  - Open XAL uses MKS throughout!
  - Users usually prefer units of mm – many applications convert m ↔ mm

- Correlated Data
  - When collecting data from many BPMs you need to verify results are all from the same beam pulse, or for the same conditions.
  - Use low beam rep-rate
  - Open XAL provides a **correlator tool** to compare timestamps on the data

Summary: Hierarchal View of an Accelerator

- The organization follows the natural working view of the machine
- This structure facilitates the writing of introspective software
- Many applications can share the same accelerator browsing methods, look and feel.
- Is configurable from a single data-source (e.g. database)

Open XAL Package Organization

Item	***Package (Location)***
Accelerator Hierarchy	**xal.smf**
Accelerator Hardware Nodes	**xal.smf.impl**
Application Framework	**xal.extension.application**
Channel Access	**xal.ca**
General Tools	**xal.tools**
Applications	**xal.apps**
Online Model	**xal.model**
Services	**xal.service**