Overall Design of SSS Software

Review of SSS Readiness for EVLA Shared Risk Observing, June 5, 2009

David M. Harland

SSS Group Lead

Introduction

- SSS Applications
- Philosophies
  - Design
  - Code Borrowing
  - Process

Applications

The following applications conform to the design presented in the following slides:

- Observation Preparation Tool (OPT)
- Source Catalog Tool (SCT)
- Resource Catalog Tool (RCT)
- Dynamic Scheduler (next version)
- EVLA Parameters Database Tool (parminator)

Special Mention:

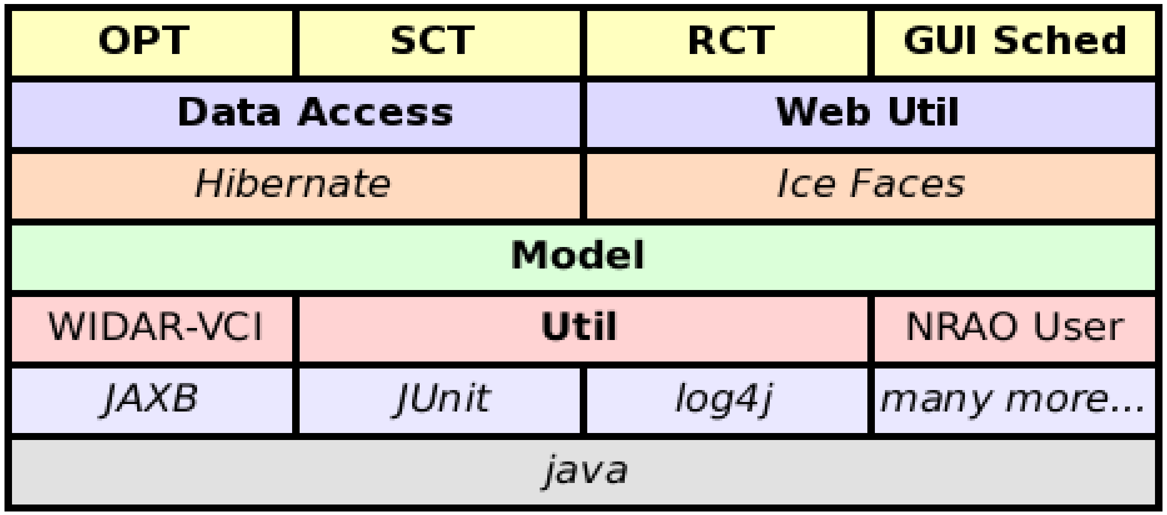
- Proposal Submission Tool (PST)
- Archive Access Tool (AAT)

Design Philosophy

- Object Oriented
- Model-View-Controller
- Large Scale Layering
  - Utility
  - Model
  - Data Storage & Access
  - View Components
  - Applications
- Layers within Layers
  - Submodels, packaging, rational divisions of responsibilities

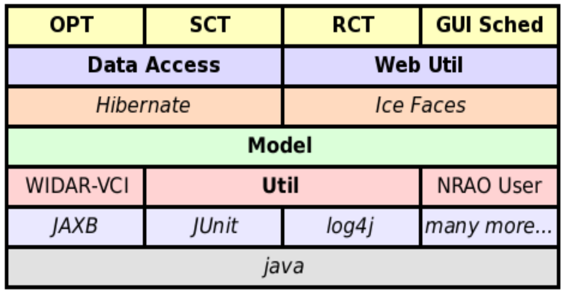
Layering

- Foundation is java.
- The lower the level, the more general and independent the code.
- Lower layers are not allowed to be aware of higher layers.
- Layers at same level are permitted to know about each other, but only when absolutely necessary.



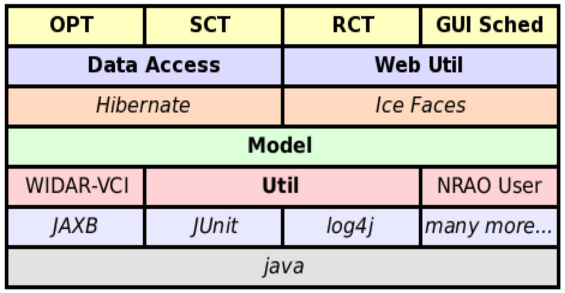
Utility Layer

- Sits directly on top of java and third-party software.
- Extremely generic; does not depend on SSS object models.
- Compiled into its own library (jar file); could be used by any NRAO java project.
  - VLBA is using.
- Examples:
  - General astronomy (ephemeris tables, Doppler tracking, coordinate systems)
  - General electronics (local oscillators, filters, mixers)
  - Sorting aides
  - Validation framework



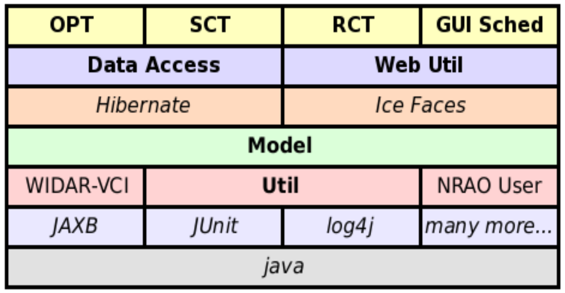
Model Layer

- Aware of utility layer, but not vice versa.
- Major Models:
  - Project Model
  - Source Model
  - Hardware Model
- Defines structure of data *and* the operations that may be performed on that data.
  - These are *object models*, not just data models.
- Defines the “business” logic.
- Fully expressible as XML.



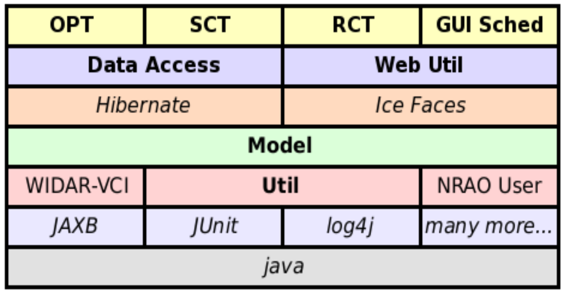
Data Access Layer

- Knows where and how to store and fetch model.
- Aware of utility & model layers, but not vice versa.
- Hides details of data access from applications.
- Allows future decision to change from, eg, RDBMS to OODBMS with no impact on utility and model layers, and minimal impact on application layer.



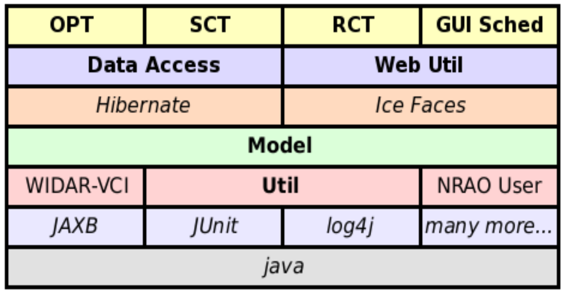
Web Interface Utility Layer

- A collection of components used by applications for gathering information from users.
- Helps keep consistent look and feel across applications.
- Reduces amount of coding needed per application.



Application Layer

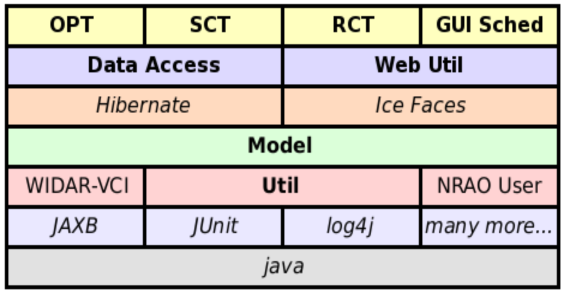
- The top layer; has access to all underneath it.
  - Acts as controller in model-view-controller (MVC) paradigm.
  - Uses javascript, html, java server faces to create view.
  - Relies on model and utility layers to implement business logic.
  - Tells data access layer when to fetch and save.



Effectiveness of Layering

The following table tallies the number of java classes (types of objects) in each of the layers. The table shows that the number of classes supporting the applications far outweighs the number of classes in any one application. If we accept such class counts as an indicator of labor saved when writing and maintaining an application, then we may conclude that such layering is an effective design philosophy.

Layer	*Number of Classes*
Utility	164
Model	283
Data Access	65
Web Util	100
RCT	43
SCT	40
OPT	34



Use of Non-SSS Code (1 of 5)

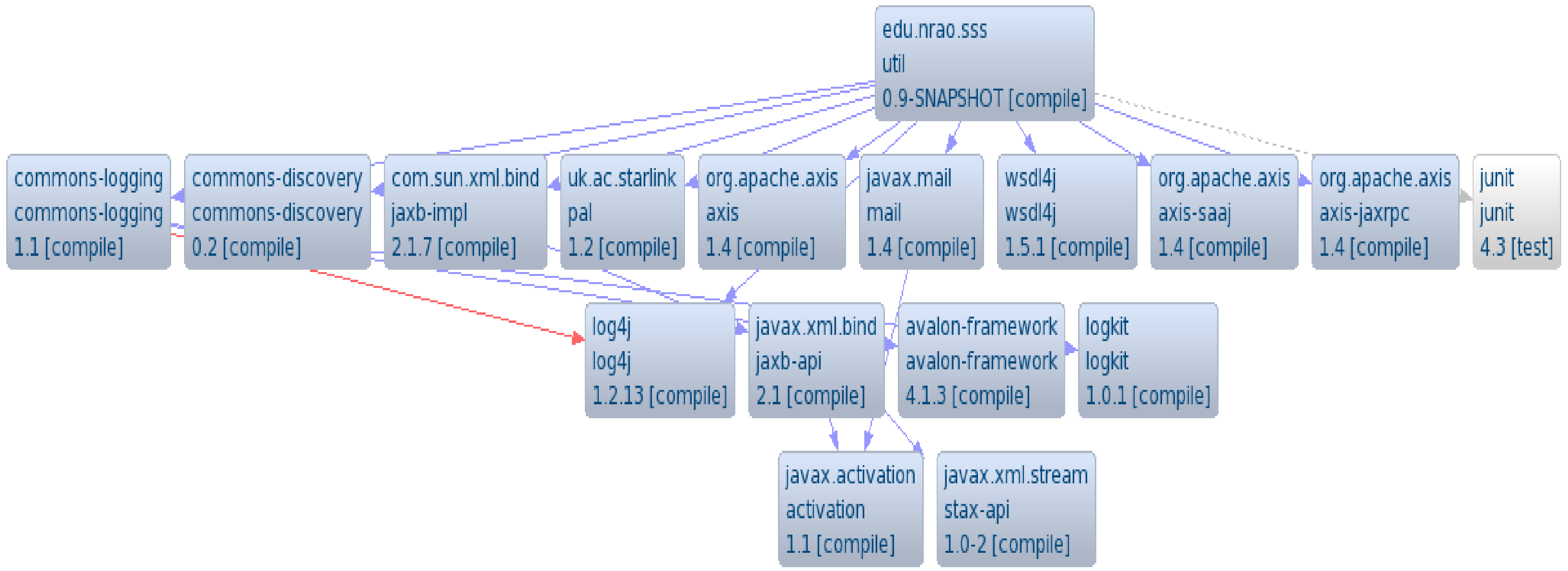
- NRAO / DRAO Libraries
  - *nrao-user*. Java classes generated from XML schema files. Used to create user objects from XML received from query service.
  - *nrao-proposal* [coming soon]. Java classes generated from XML schema files. Used to create proposal objects from XML received from query service.
  - *vci*. Virtual correlator interface for WIDAR correlator. Java classes generated from XML schema files. Used to create VCI objects that will are converted to XML messages for the correlator.
- Other Astronomy Libraries
  - *uk.ac.starlink.pal*. Partial java port of SLALIB. Used for Doppler tracking and some coordinate system conversions.

Use of Non-SSS Code (2 of 5)

- Open Source / Third-Party Libraries
  - *Hibernate*. An object-relational mapping tool. Helps with storing model in RDBMS.
  - *IceFaces*. Helps with graphical widgets in browser.
  - *JUnit*. Unit testing framework.
  - *log4j*. Message logging framework.
  - Many more. Most of the 3rd party libraries upon which we depend in turn depend on other libraries.

Use of Non-SSS Code (3 of 5)

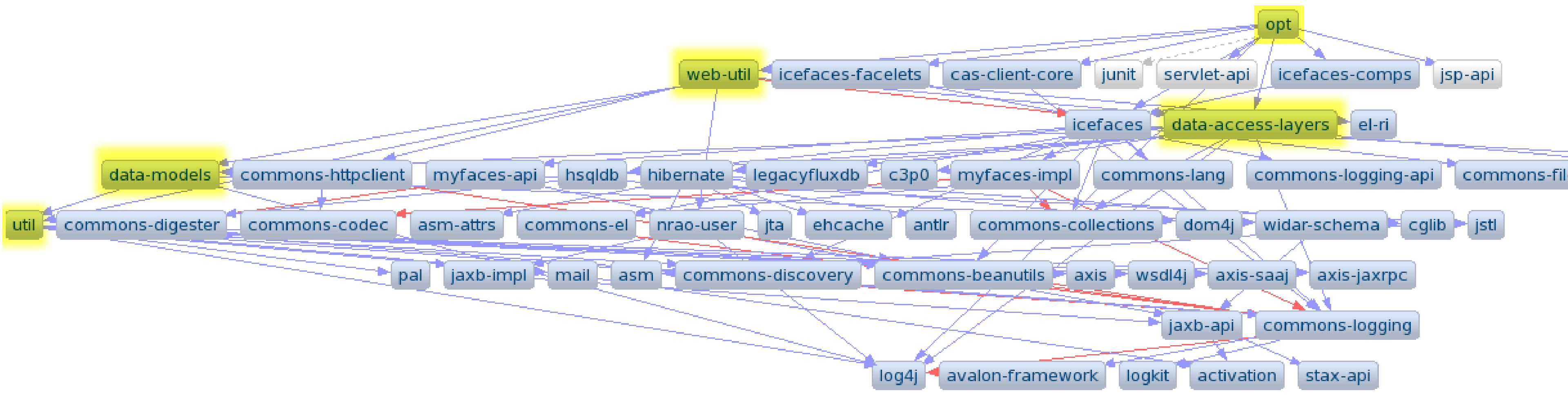
Pictures of Complexity - Dependencies for Utility Layer:


edu.nrao.sss util 0.9-SNAPSHOT [compile]
commons-logging commons-logging 1.1 [compile]
commons-discovery commons-discovery 0.2 [compile]
com.sun.xml.bind jaxb-impl 2.1.7 [compile]
uk.ac.starlink pal 1.2 [compile]
org.apache.axis axis 1.4 [compile]
javax.mail mail 1.4 [compile]
wsdl4j wsdl4j 1.5.1 [compile]
org.apache.axis axis-saaj 1.4 [compile]
org.apache.axis axis-jaxrpc 1.4 [compile]
junit junit 4.3 [test]
log4j log4j 1.2.13 [compile]
javax.xml.bind jaxb-api 2.1 [compile]
avalon-framework avalon-framework 4.1.3 [compile]
logkit logkit 1.0.1 [compile]
javax.activation activation 1.1 [compile]
javax.xml.stream stax-api 1.0-2 [compile]

EVLA

Use of Non-SSS Code (4 of 5)

Pictures of Complexity - Dependencies for OPT:



NRAO

Use of Non-SSS Code (5 of 5)

Depending on the Code of Others

Pros:

1. Lots of logic we did not have to write.
2. Planet full of people using and testing.
3. Written by topical experts.
4. So far all our borrowed code is “free”.

Cons:

1. Get what you pay for? Support is hit or miss.
2. SSS not in control of timing of bug fixes.
3. Black boxes. Often when problem occurs SSS cannot be sure borrowed software is at fault, or if instead SSS is using it in improper manner. Lots of time can be lost in debugging effort.
4. Can become defunct. E.g., uk.ac.starlink.pal is no longer supported.

Development Process (1 of 2)

- Issue Tracking
  - JIRA
  - For bugs, improvements, new features
- Source Code Control
  - Subversion
  - Branching, tagging, trunk
    - Documented procedures for “when to” and “how to”.
- Build
  - Maven 2
    - Helps us manage dependencies.
    - Makes it easy for new developers to build systems from scratch.
  - Automated procedure for nightly builds from trunk.
  - All projects are required to have maven 2 build configuration.

Development Process (2 of 2)

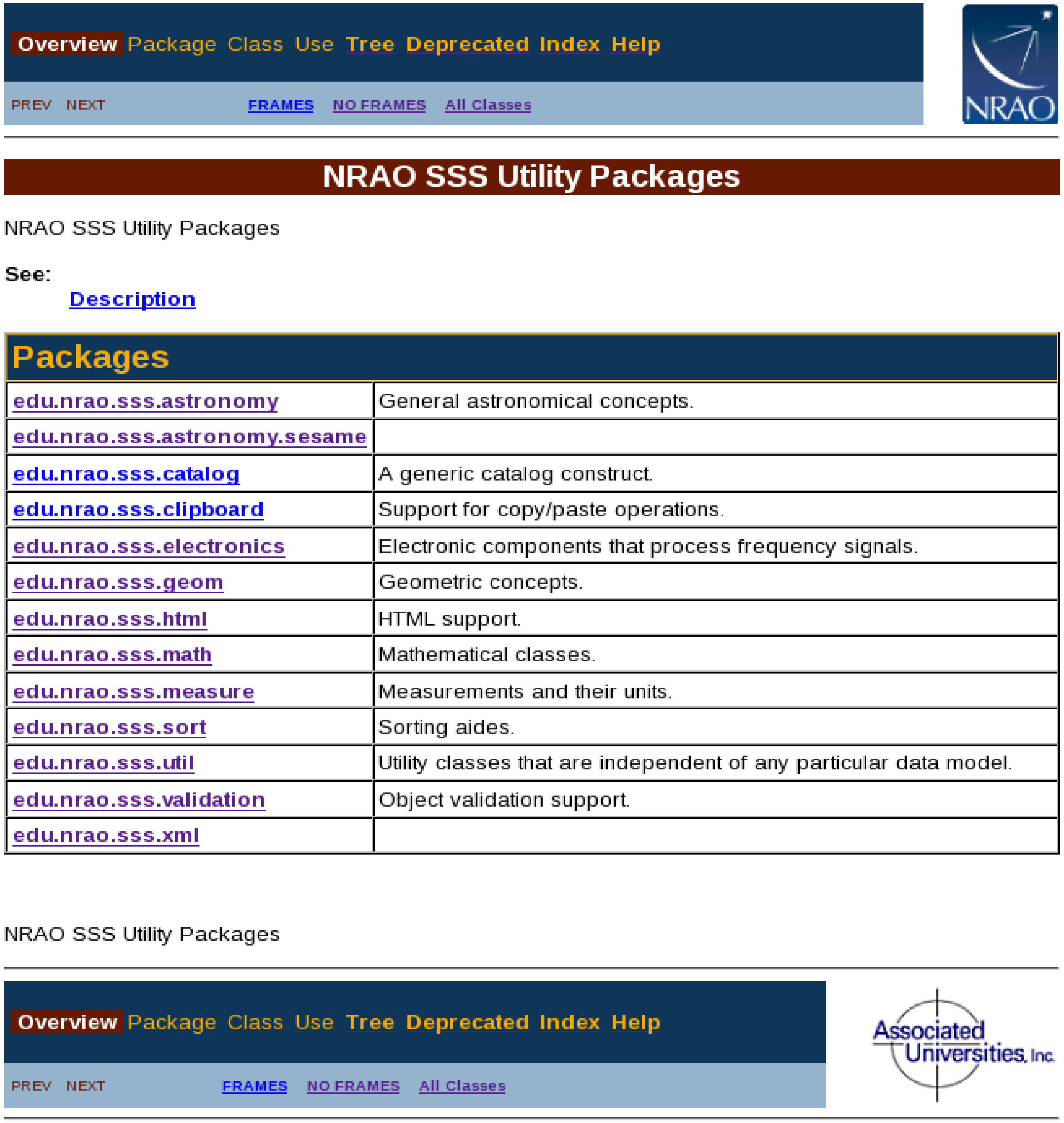
- System Documentation
  - Java makes it very easy to keep package, class, and method documentation up to date.
    - The utility, model, data access, and web utility layers are all expected to have good documentation at this level.
  - Higher level documentation
    - Some UML diagrams.
- Testing
  - JUnit framework used.
    - Nightly build runs suite of unit tests.
    - It is mainly the utility and model layers that use JUnit.
  - Applications deployed to test server.
    - Users expected to bless test code prior to it going into production.

References

- SSS Software Documentation (https://staff.nrao.edu/evla/sss/index.shtml)
  - Contains mid-level documentation, UML diagrams, API documentation, links to applications, meeting minutes, links to test programs.
- SSS Internal Wiki Pages (https://staff.nrao.edu/wiki/bin/view/EVLA/SSS)
  - Contains policies and procedures for building applications, technical references, knowledge base.
- SSS XML Schema Files (ver 0.8.0: https://e2e.nrao.edu/schemas/sss/0.8.0/)

End of Presentation

Supplementary Material



Overview Package Class Use Tree Deprecated Index Help
PREV NEXT FRAMES NO FRAMES All Classes

NRAO SSS Utility Packages

NRAO SSS Utility Packages

See:
Description

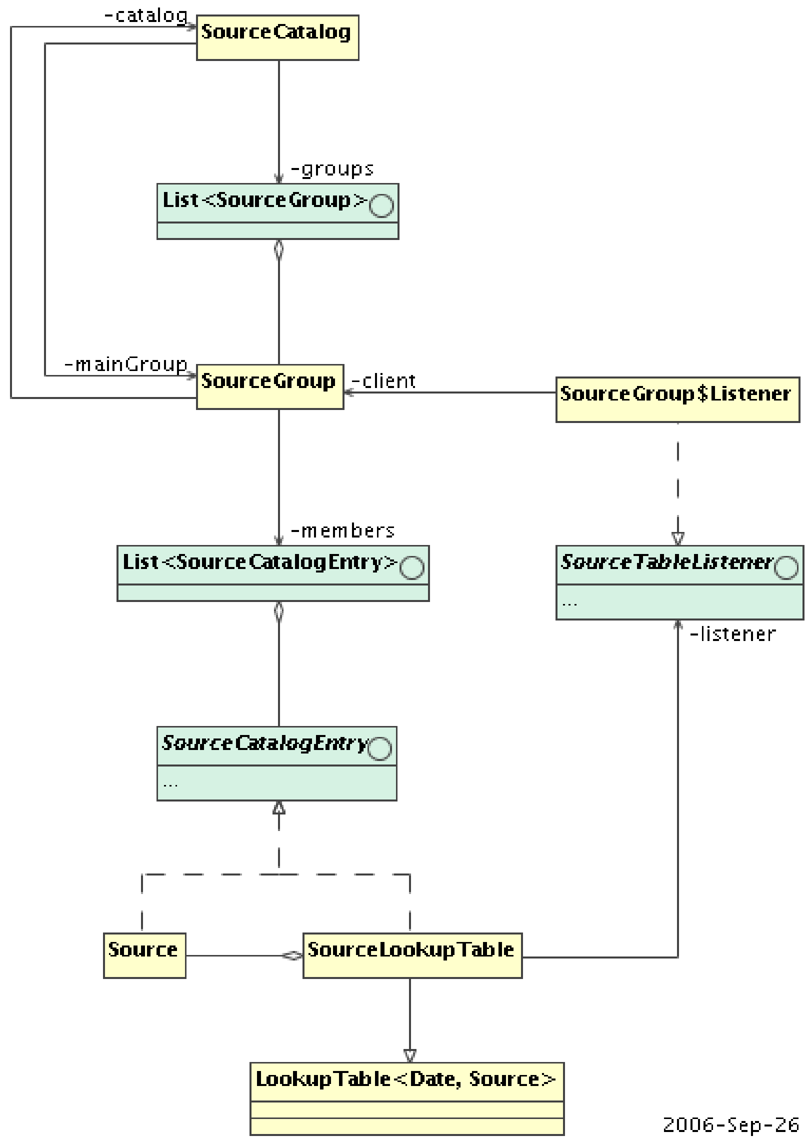
Packages	
edu.nrao.sss.astronomy	General astronomical concepts.
edu.nrao.sss.astronomy.sesame	
edu.nrao.sss.catalog	A generic catalog construct.
edu.nrao.sss.clipboard	Support for copy/paste operations.
edu.nrao.sss.electronics	Electronic components that process frequency signals.
edu.nrao.sss.geom	Geometric concepts.
edu.nrao.sss.html	HTML support.
edu.nrao.sss.math	Mathematical classes.
edu.nrao.sss.measure	Measurements and their units.
edu.nrao.sss.sort	Sorting aides.
edu.nrao.sss.util	Utility classes that are independent of any particular data model.
edu.nrao.sss.validation	Object validation support.
edu.nrao.sss.xml	

NRAO SSS Utility Packages

Overview Package Class Use Tree Deprecated Index Help
PREV NEXT FRAMES NO FRAMES All Classes

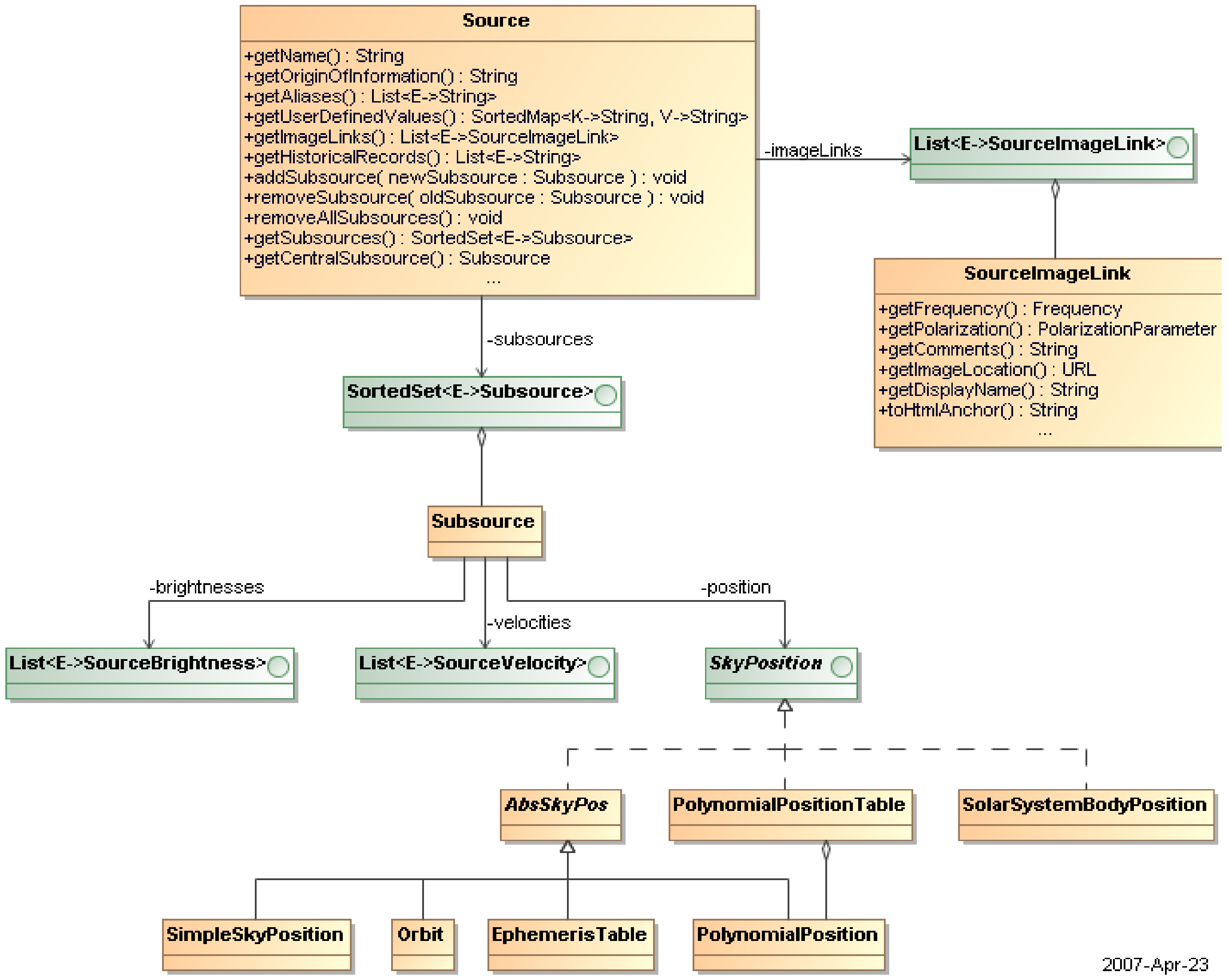
Backup Slide: Sample API Doc

SourceCatalog: Overview



Backup Slide: Sample UML Class Diagram

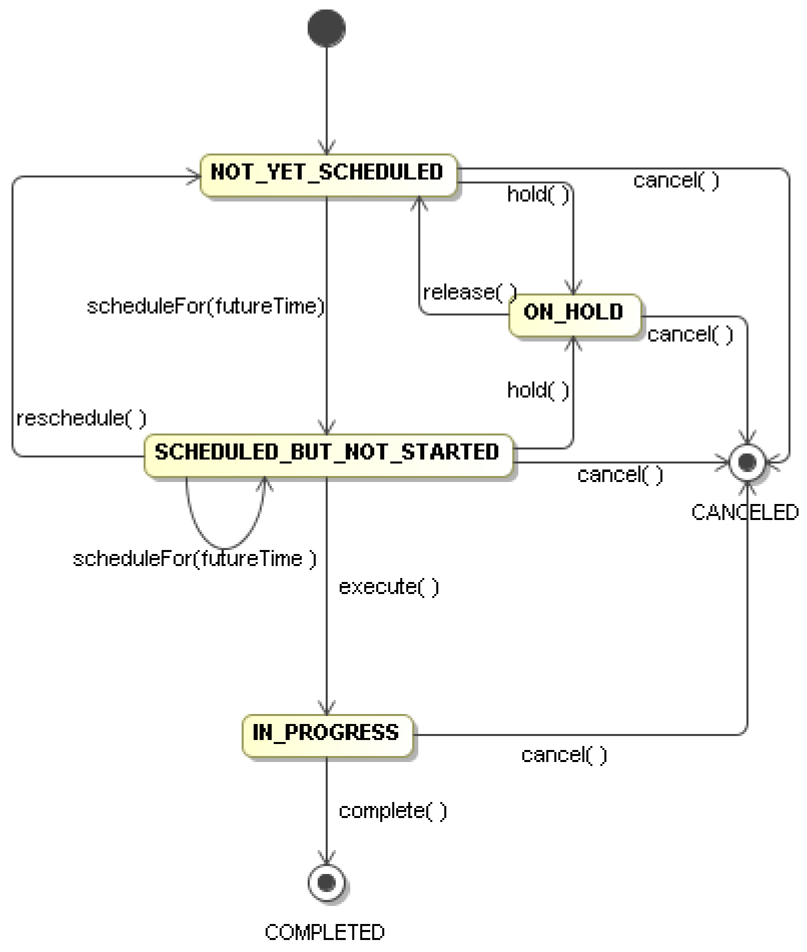
Source Model: Major Objects


Source
+getName() : String
+getOriginOfInformation() : String
+getAliases() : List<E->String>
+getUserDefinedValues() : SortedMap<K->String, V->String>
+getImageLinks() : List<E->SourceImageLink>
+getHistoricalRecords() : List<E->String>
+addSubsource(newSubsource : Subsource) : void
+removeSubsource(oldSubsource : Subsource) : void
+removeAllSubsources() : void
+getSubsources() : SortedSet<E->Subsource>
+getCentralSubsource() : Subsource
...
-imageLinks
List<E->SourceImageLink>
SourceImageLink
+getFrequency() : Frequency
+getPolarization() : PolarizationParameter
+getComments() : String
+getImageLocation() : URL
+getDisplayName() : String
+toHtmlAnchor() : String
...
-subsources
SortedSet<E->Subsource>
Subsource
-brightnesses
-velocities
-position
List<E->SourceBrightness>
List<E->SourceVelocity>
SkyPosition
AbsSkyPos
PolynomialPositionTable
SolarSystemBodyPosition
SimpleSkyPosition
Orbit
EphemerisTable
PolynomialPosition

2007-Apr-23

Backup Slide: Sample UML Class Diagram

States & Transitions of the EventStatus of a ScheduleEntry



This diagram shows the different *EventStatus* states that a *ScheduleEntry* object may have. It also shows all valid transitions from one state to another, along with the names of the events that cause those tranisitions.

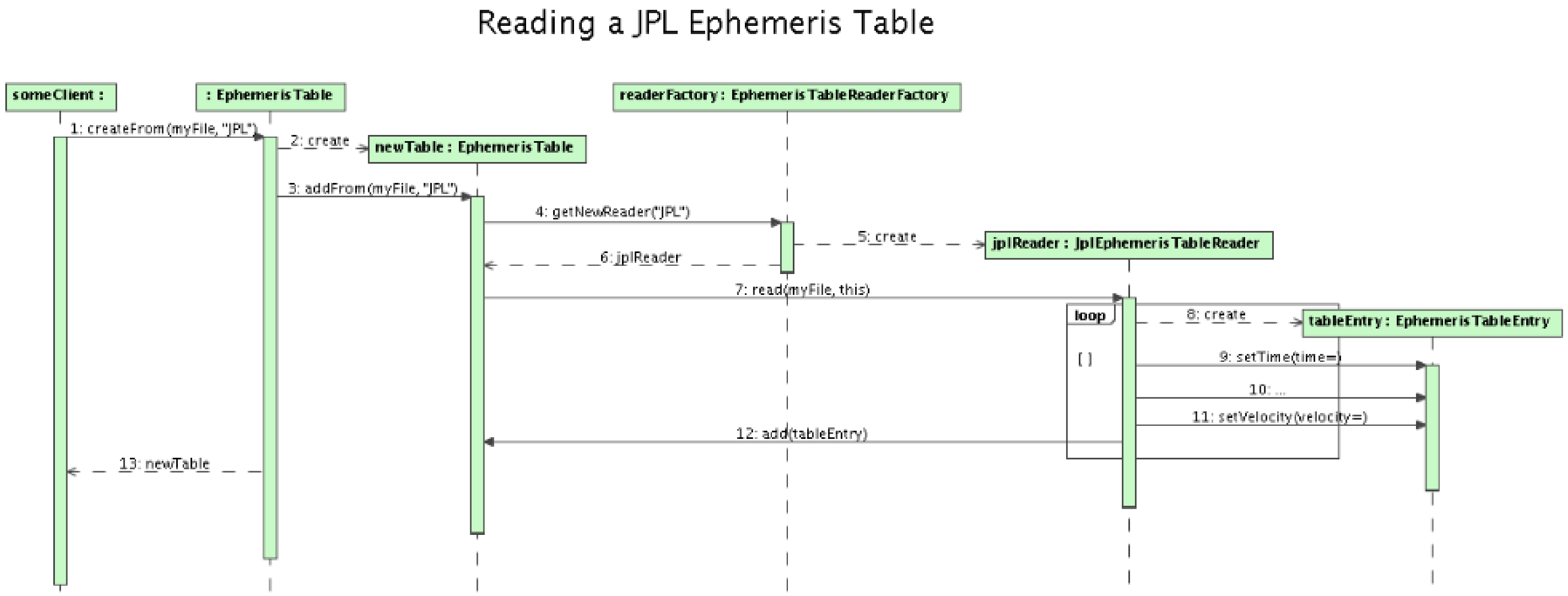
We use a *ScheduleEntry* object to represent the scheduling of a single iteration of a *SchedulingBlock* (*SB*). The scheduler is free to schedule each entry of a single *SB* at dispersed points in time.

In this model, an entry that is *IN_PROGRESS* can proceed only to a final state of either *COMPLETED* (meaning the execution was successful) or *CANCELED* (meaning the execution failed). An entry that moves from *IN_PROGRESS* to *CANCELED* is said to have been *aborted*, and a count is kept of all the entries of an *SB* that were aborted. We cannot pause, or put *ON_HOLD*, an entry that is already *IN_PROGRESS*.

2007-Aug-27

(See full page on transitions.)

Backup Slide: Sample UML State Diagram


Reading a JPL Ephemeris Table
someClient :
: EphemerisTable
readerFactory : EphemerisTableReaderFactory
newTable : EphemerisTable
jplReader : JplEphemerisTableReader
tableEntry : EphemerisTableEntry
1: createFrom(myFile, "JPL")
2: create
3: addFrom(myFile, "JPL")
4: getNewReader("JPL")
5: create
6: jplReader
7: read(myFile, this)
loop
[ ]
8: create
9: setTime(time=)
10: ...
11: setVelocity(velocity=)
12: add(tableEntry)
13: newTable

Backup Slide: Sample UML Sequence Diagram

```xml
<?xml version="1.0"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">

  <modelVersion>4.0.0</modelVersion>
  <groupId>edu.nrao.sss</groupId>
  <artifactId>data-models</artifactId>
  <packaging>jar</packaging>
  <version>0.9-SNAPSHOT</version>
  <name>Shared Data Models</name>
  <url>http://www.aoc.nrao.edu/asg-internal/maven/m2-sites/shared/data-models/</url>

  <properties>
    <widar.version>3.6-SNAPSHOT</widar.version>  <!-- Coordinate w/ dependency, below -->
  </properties>

  <dependencies>
    <dependency>
      <groupId>edu.nrao.sss</groupId>
      <artifactId>util</artifactId>
      <version>0.9-SNAPSHOT</version>
    </dependency>

    <dependency>
      <groupId>nrao</groupId>
      <artifactId>widar-schema</artifactId>
      <version>3.6-SNAPSHOT</version>  <!-- See also properties section, above -->
    </dependency>

    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.3</version>
      <scope>test</scope>
    </dependency>

    <dependency>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
      <version>1.2.13</version>
    </dependency>

  </dependencies>
```

Backup Slide: Sample Maven2 Build File (1 of 3)

```xml
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <source>1.6</source>
        <target>1.6</target>
        <compilerArgument>-Xlint:all</compilerArgument>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-javadoc-plugin</artifactId>
      <configuration>
        <aggregate>true</aggregate>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-source-plugin</artifactId>
      <executions>
        <execution>
          <id>attach-sources</id>
          <phase>verify</phase>
          <goals>
            <goal>jar</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>2.3</version>
      <configuration>
        <argLine>-Xmx512m</argLine>
      </configuration>
    </plugin>
  </plugins>

  <resources>
    <resource>
      <directory>src/main/resources</directory>
      <filtering>true</filtering>
    </resource>
  </resources>
</build>
```

Backup Slide: Sample Maven2 Build File (2 of 3)

```
  <repositories>
    <repository>
      <id>nrao-internal</id>
      <name>NRAO Maven 2 Repository</name>
      <url>file:///home/asg/www-internal/maven/m2-repo/</url>
      <releases>
        <enabled>true</enabled>
      </releases>
      <snapshots>
        <enabled>true</enabled>
      </snapshots>
    </repository>
  </repositories>

  <distributionManagement>
    <repository>
      <id>nrao-internal</id>
      <name>NRAO Maven 2 Repository</name>
      <url>file:///home/asg/www-internal/maven/m2-repo/</url>
      <uniqueVersion>false</uniqueVersion>
    </repository>

    <site>
      <id>shared-websites</id>
      <url>file:///home/staff.nrao.edu/content/evla/maven/m2-sites/shared/data-models/</url>
    </site>
  </distributionManagement>

  <reporting>
    <!-- edited out for display on this slide -->
  </reporting>
</project>
```

Backup Slide: Sample Maven2 Build File (3 of 3)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  version         = "1.0"
  xmlns:xs        = "http://www.w3.org/2001/XMLSchema"
  targetNamespace = "http://www.nrao.edu/namespaces/sss"
  xmlns:sss       = "http://www.nrao.edu/namespaces/sss"
  elementFormDefault = "qualified">

  <xs:include schemaLocation="commons.xsd"/>
  <xs:include schemaLocation="measure.xsd"/>
  <xs:include schemaLocation="astronomy.xsd"/>

  <!-- ================================================== -->
  <!-- ||||||||||||||||  ELEMENTS  |||||||||||||||||||||| -->
  <!-- ================================================== -->

  <xs:element name="sourceCatalog">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name"         type="xs:string"/>
        <xs:element name="notes"        type="sss:noteList" minOccurs="0"/>
        <xs:element name="owner"        type="xs:long"/>
        <xs:group                        ref="sss:userTrackable"/>
        <xs:element name="sources"      type="sss:sourceList"/>
        <xs:element name="sourceTables" type="sss:sourceTableList"/>
        <xs:element name="sourceGroups" type="sss:sourceGroupList"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:long" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="source" type="sss:source"/>
  <xs:element name="sourceLookupTable" type="sss:sourceLookupTable"/>

  <!-- ================================================== -->
  <!-- ||||||||||||||||  COMPLEX TYPES  ||||||||||||||||| -->
  <!-- ================================================== -->

  <xs:complexType name="sourceGroup">
    <xs:sequence>
      <xs:element name="name"   type="xs:string"/>
      <xs:element name="notes"  type="sss:noteList" minOccurs="0"/>

      <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:element name="source" type="xs:IDREF"/>
      </xs:sequence>

      <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:element name="sourceTable" type="xs:IDREF"/>
      </xs:sequence>
    </xs:sequence>
    <xs:attribute name="id" type="xs:long" use="required"/>
  </xs:complexType>

          <!-- Remainder removed for display on this slide. -->
```

Backup Slide: Partial XML Schema File

```xml
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="edu.nrao.sss.model.source">

<!-- +++++ SOURCE CATALOG +++++ -->

  <class name="SourceCatalog" table="Source_Catalog">

    <id name="Id" type="long">
      <generator class="native"/>
    </id>

    <property name="Name"/>

    <property name="Owner" not-null="true"/>
    <property name="CreatedOn"/>
    <property name="CreatedBy"/>
    <property name="LastUpdatedOn"/>
    <property name="LastUpdatedBy"/>

    <list name="Notes" table="Source_Catalog_Notes">
      <key    column="SourceCatalog_ID"/>
      <index column="idx"/>
      <element column="Note" type="text" not-null="true"/>
    </list>

    <list name="groups" cascade="all-delete-orphan"
          access="field">
      <key column="Catalog_ID" not-null="true"/>
      <list-index column="idx"/>
      <one-to-many class="SourceGroup"/>
    </list>

    <list name="Entries" cascade="all-delete-orphan">
      <key column="Catalog_ID" not-null="true"/>
      <list-index column="idx"/>
      <one-to-many class="SourceCatalogEntry"/>
    </list>

  </class>

<!-- +++++ SOURCE GROUP +++++ -->

<class name="SourceGroup" table="Source_Group">

  <id name="Id" type="long">
    <generator class="native"/>
  </id>

  <property name="Name"/>

  <list name="Notes" table="Source_Group_Notes">
    <key    column="SourceGroup_ID"/>
    <index column="idx"/>
    <element column="Note" type="text" not-null="true"/>
  </list>

  <many-to-one name="catalog" class="SourceCatalog"
    column   = "Catalog_ID"
    not-null = "true"
    insert   = "false"
    update   = "false"
    access   = "field"/>

  <list name="InternalMemberList"
      table="Source_Group_Member"
      cascade="save-update">
    <key column="SourceGroup_ID"/>
    <index column="idx"/>
    <many-to-many class="SourceCatalogEntry"
                  column="Member_ID"/>
  </list>

</class>

</hibernate-mapping>
```

Backup Slide: Sample Hibernate Mapping

How Many Unit Tests?

Methods labeled with “@Test” marker:

util:	214
model:	108

Most methods have many assertions (tests) inside. Estimate the average to be 5 per test method. Gives ~1,500 individual tests. Most of these use data derived from random number generation and are inside loops of 50 to 10,000 iterations.

End of Supplementary Material