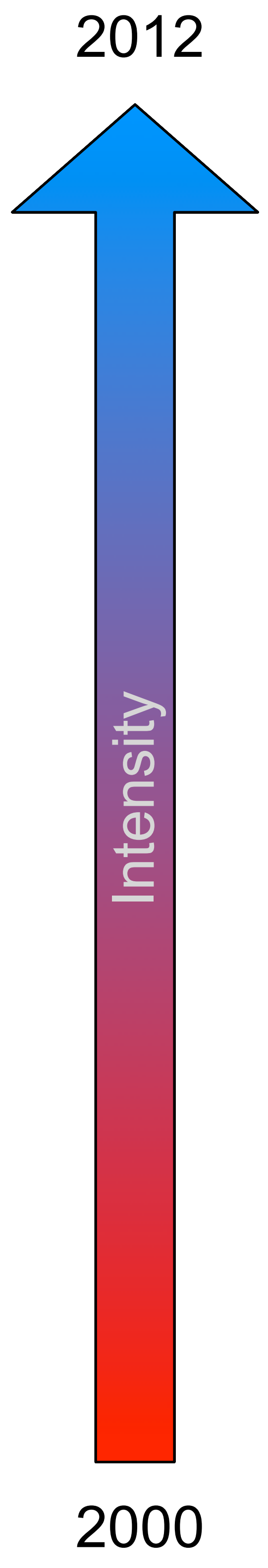
Open XAL Status Report - 2012

Thomas Pelaia II, Ph.D.

XAL Workshop 2012

December 13, 2012

XAL Loose Timeline at SNS



Software Maintenance

Neutron Production

Operations Software Development

Commissioning

Machine Study Software Development

Core Development

Construction

XAL Architecture Design

Motivation

- Collaborator interest
- Opportunity to cleanup code

Benefits

- ✓Disciplined
- ✓Collaboration
- ✓Remove obsolete and unused code
- ✓Support for latest third party libraries
- ✓Addressed thousands of compile warnings
- ✓Simpler, more powerful build system
- ✓Easier maintenance
- ✓Performance Improvements (Live model sync)
- ✓Bug fixes

Open XAL Build Architecture

- Hierarchical Ant build system
  - Configuration trickles down from top
  - Batch build targets point down
- Clean Source Separation
- Zero Configuration with options
- Maintenance Free build targets
- Integrated unit testing
- Support for standalone applications
- Option for deployment installation

Why Ant

- Command line tool
- Popular Java counterpart to Make
- Power and Simplicity
- Integration with many IDEs

IDE Independent
Listed Alphabetically

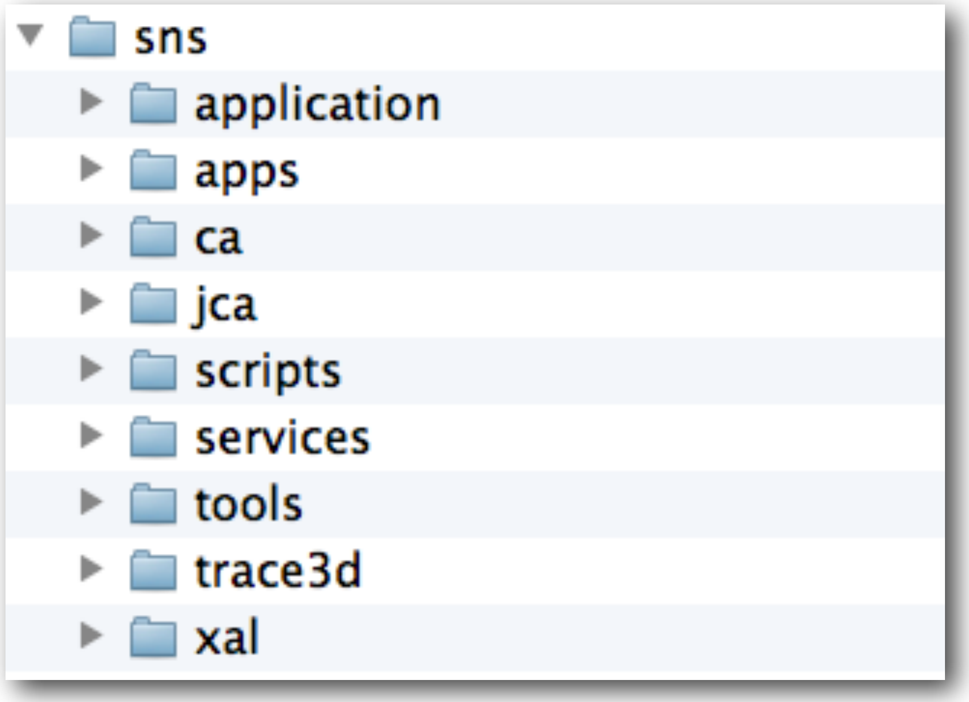
- Current XAL IDEs at SNS
  - Eclipse
  - JEdit
  - Netbeans
  - Xcode

Why not Maven?

- Any thoughts?
- Any Experts?
- Added complexity
- Motivation
- Return on Investment

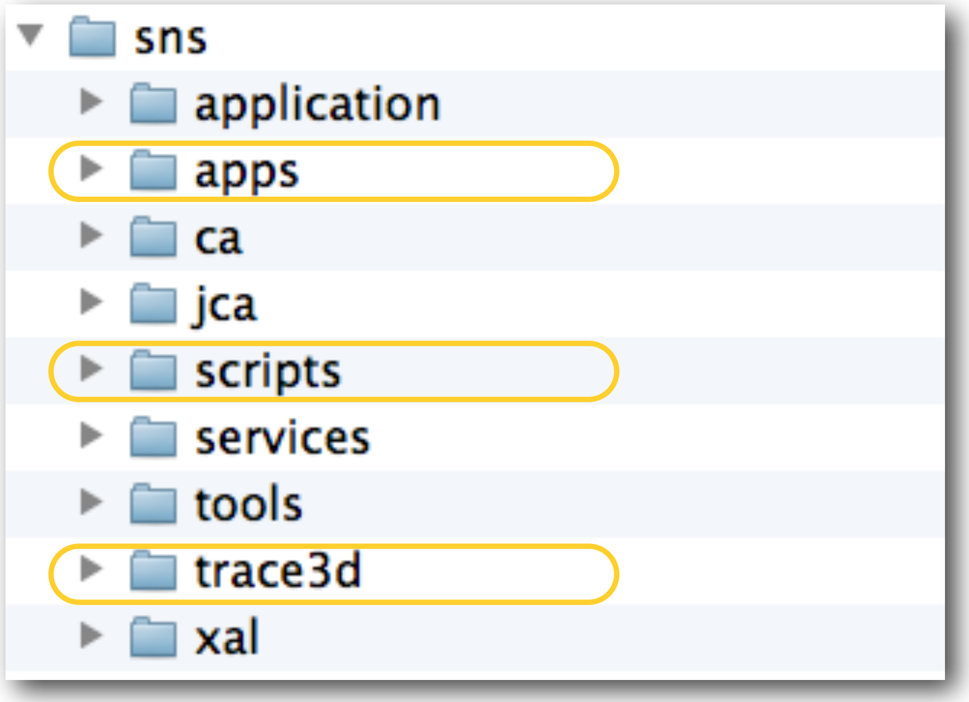
XAL Mixed Source

- **Build files, resources, scripts and unit test code mixed with main source code**
  - **Complicated build rules**
  - **Unit Tests ship with deployed executables**
- **Applications, Services and Core embedded within the same package tree**



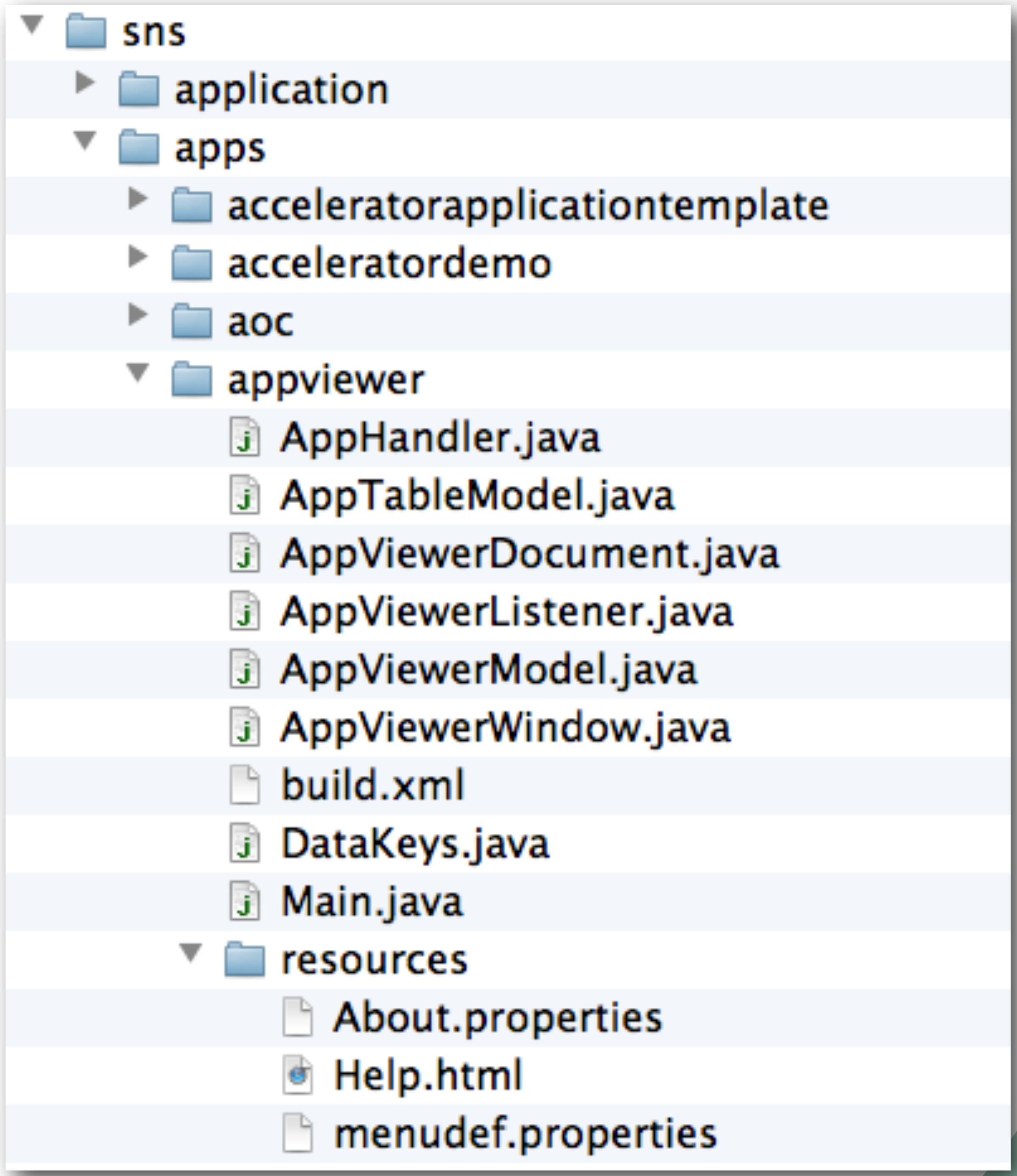
XAL Mixed Source

- **Build files, resources, scripts and unit test code mixed with main source code**
  - **Complicated build rules**
  - **Unit Tests ship with deployed executables**
- **Applications, Services and Core embedded within the same package tree**



XAL Mixed Source

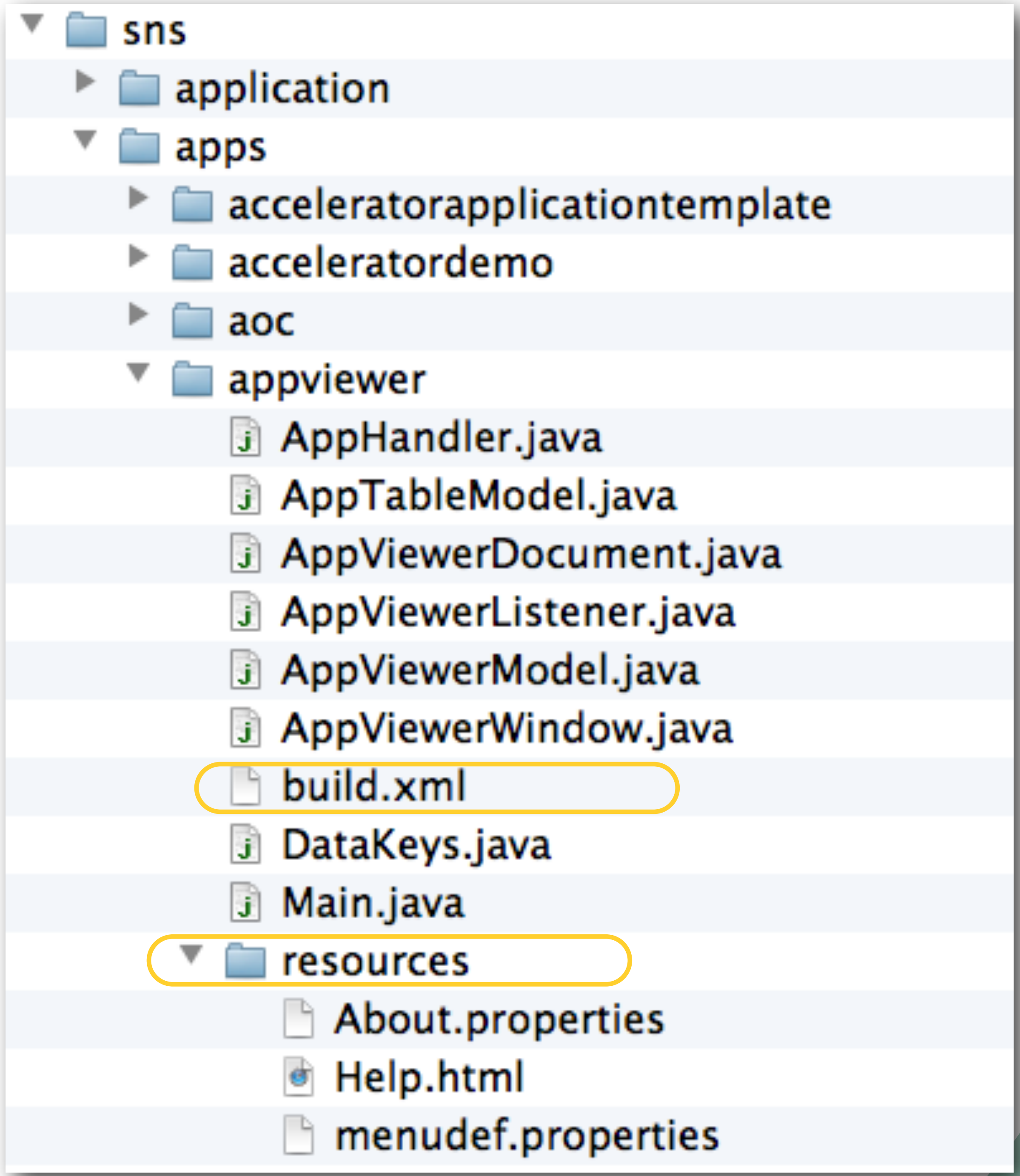
- Build files, resources, scripts and unit test code mixed with main source code
  - Complicated build rules
  - Unit Tests ship with deployed executables
- Applications, Services and Core embedded within the same package tree



```
▼ sns
  ▶ application
  ▼ apps
    ▶ acceleratorapplicationtemplate
    ▶ acceleratordemo
    ▶ aoc
    ▼ appviewer
        AppHandler.java
        AppTableModel.java
        AppViewerDocument.java
        AppViewerListener.java
        AppViewerModel.java
        AppViewerWindow.java
        build.xml
        DataKeys.java
        Main.java
      ▼ resources
          About.properties
          Help.html
          menudef.properties
```

XAL Mixed Source

- Build files, resources, scripts and unit test code mixed with main source code
  - Complicated build rules
  - Unit Tests ship with deployed executables
- Applications, Services and Core embedded within the same package tree

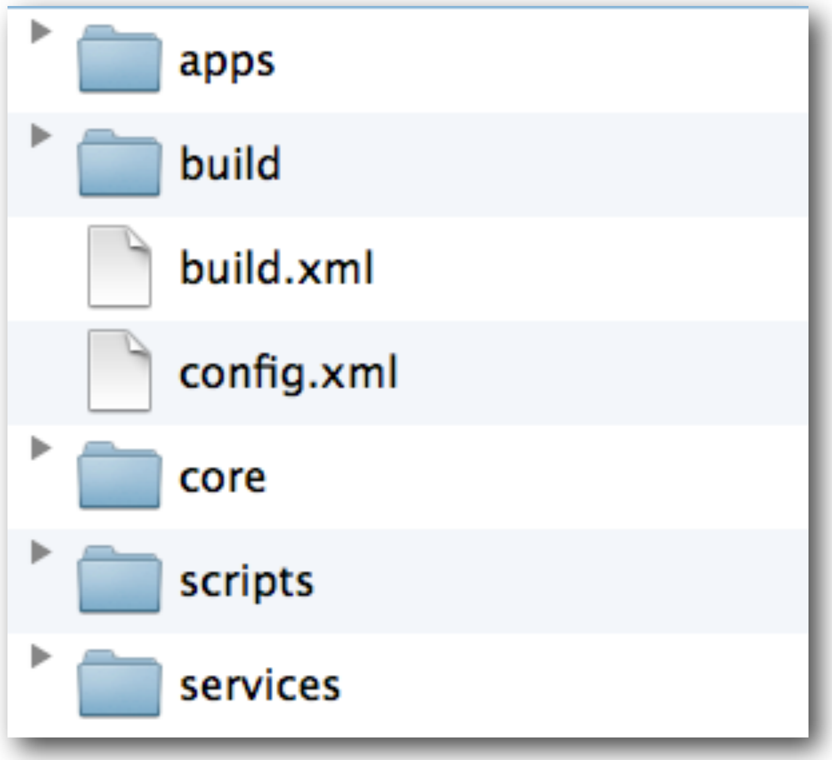


```
▼ sns
  ▶ application
  ▼ apps
    ▶ acceleratorapplicationtemplate
    ▶ acceleratordemo
    ▶ aoc
    ▼ appviewer
        AppHandler.java
        AppTableModel.java
        AppViewerDocument.java
        AppViewerListener.java
        AppViewerModel.java
        AppViewerWindow.java
        build.xml
        DataKeys.java
        Main.java
      ▼ resources
          About.properties
          Help.html
          menudef.properties
```

Open XAL Clean Source Separation
Role Based in Directory Structure

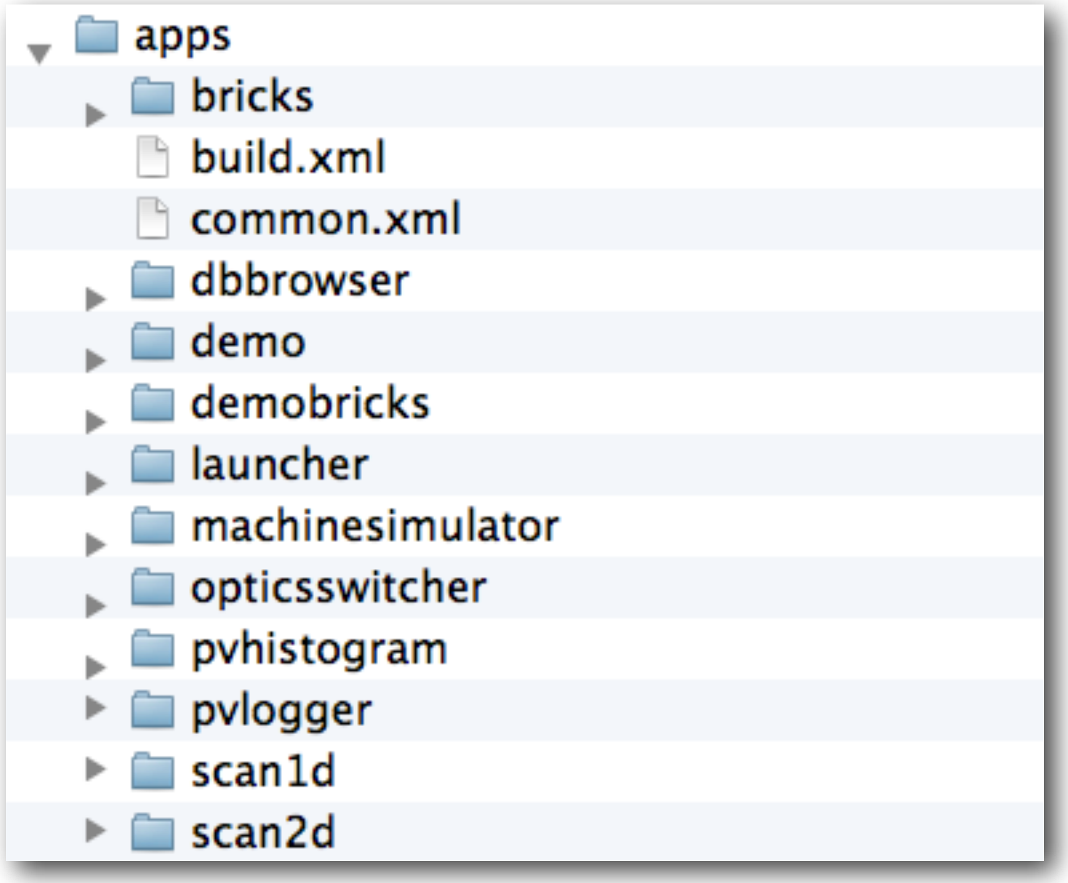
- Applications, Services and Core
  - Easier navigation
- Unit testing and Production sources
  - Separate unit test and deployable executables
- Build files, Libraries, Resources and Source Tree
  - Simpler, maintenance free build rules

Project Root



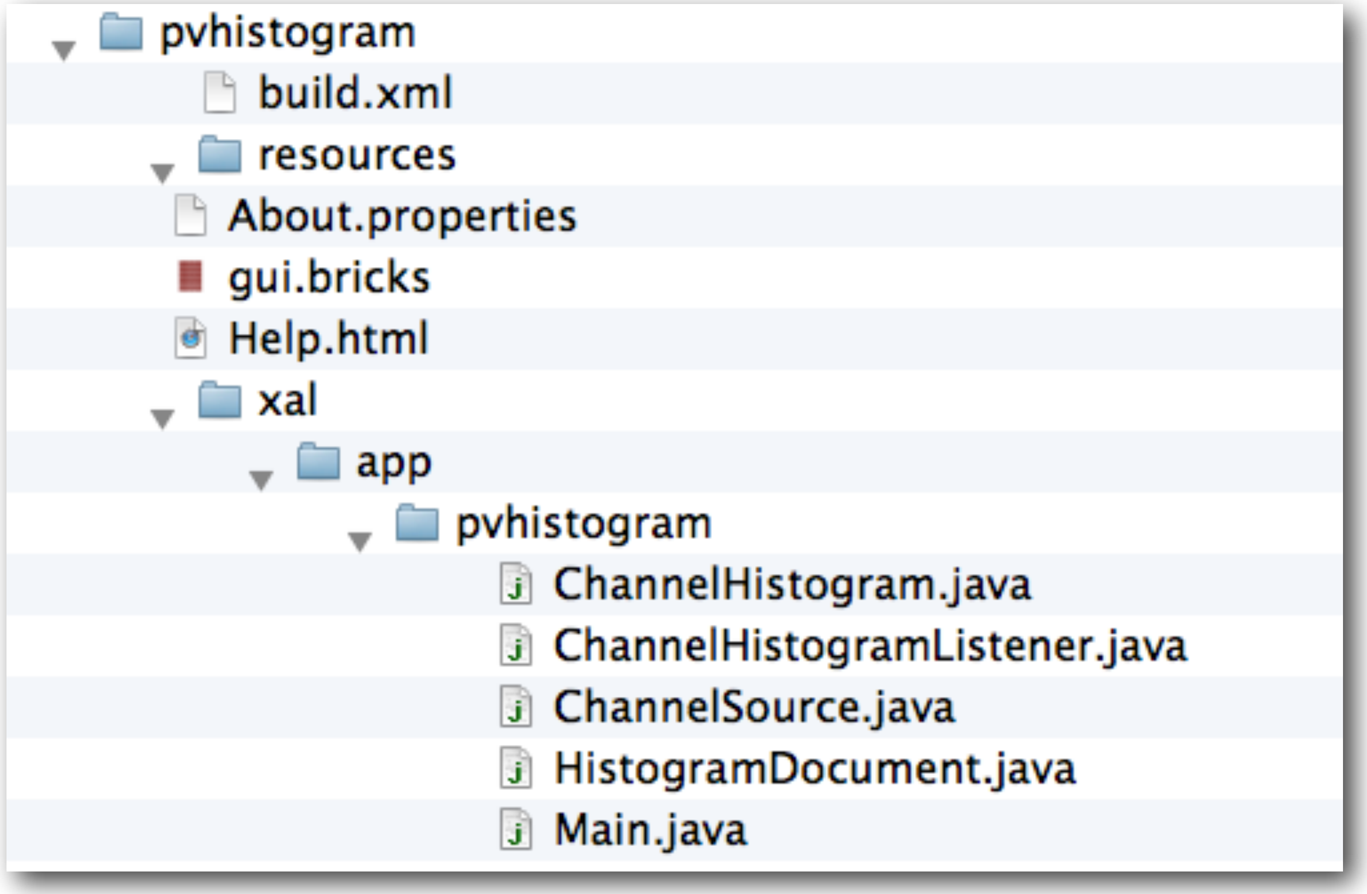
apps
build
build.xml
config.xml
core
scripts
services

Applications File Structure



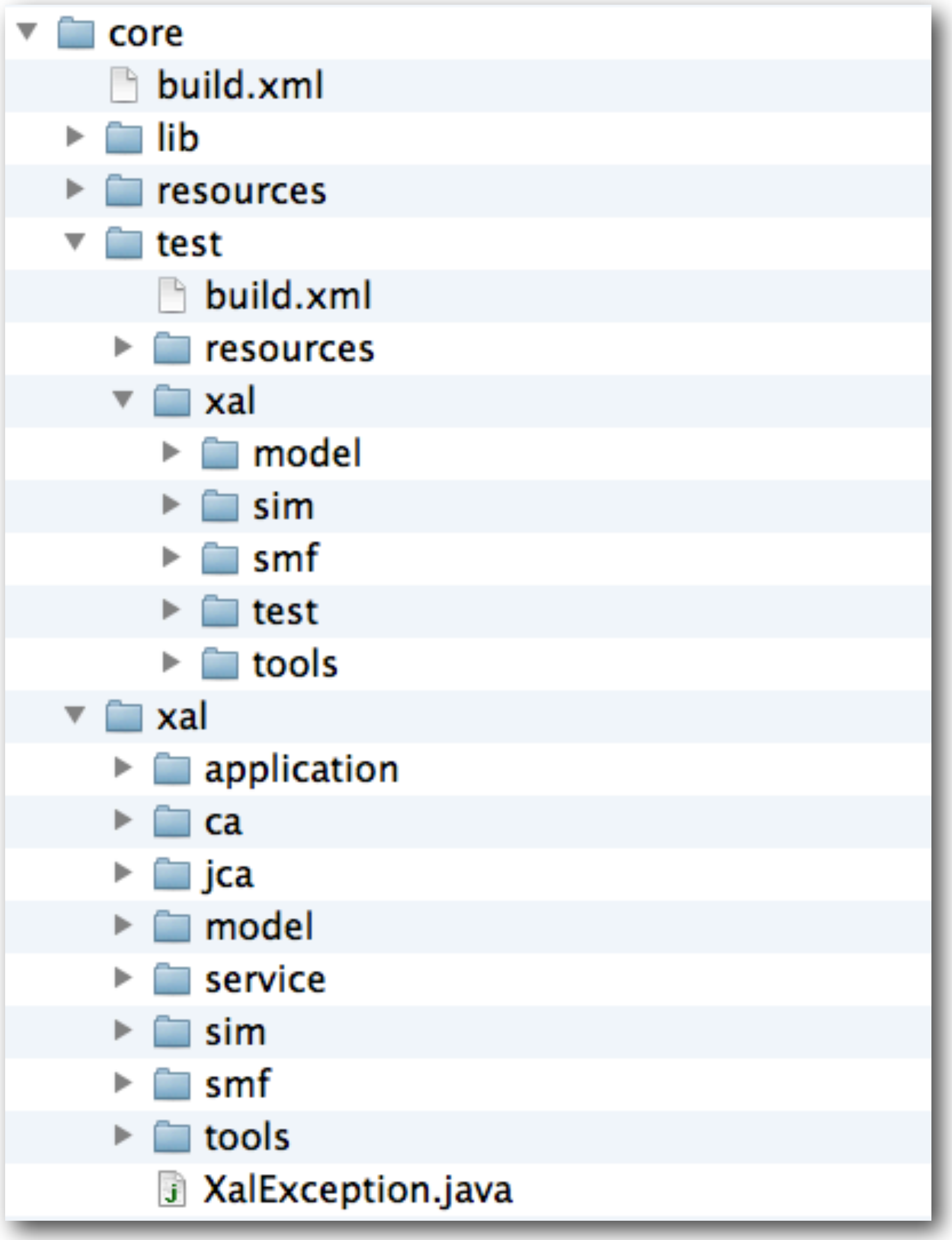
```
▼ apps
   ▶ bricks
     build.xml
     common.xml
   ▶ dbbrowser
   ▶ demo
   ▶ demobricks
   ▶ launcher
   ▶ machinesimulator
   ▶ opticsswitcher
   ▶ pvhistogram
   ▶ pvlogger
   ▶ scan1d
   ▶ scan2d
```

Application File Structure



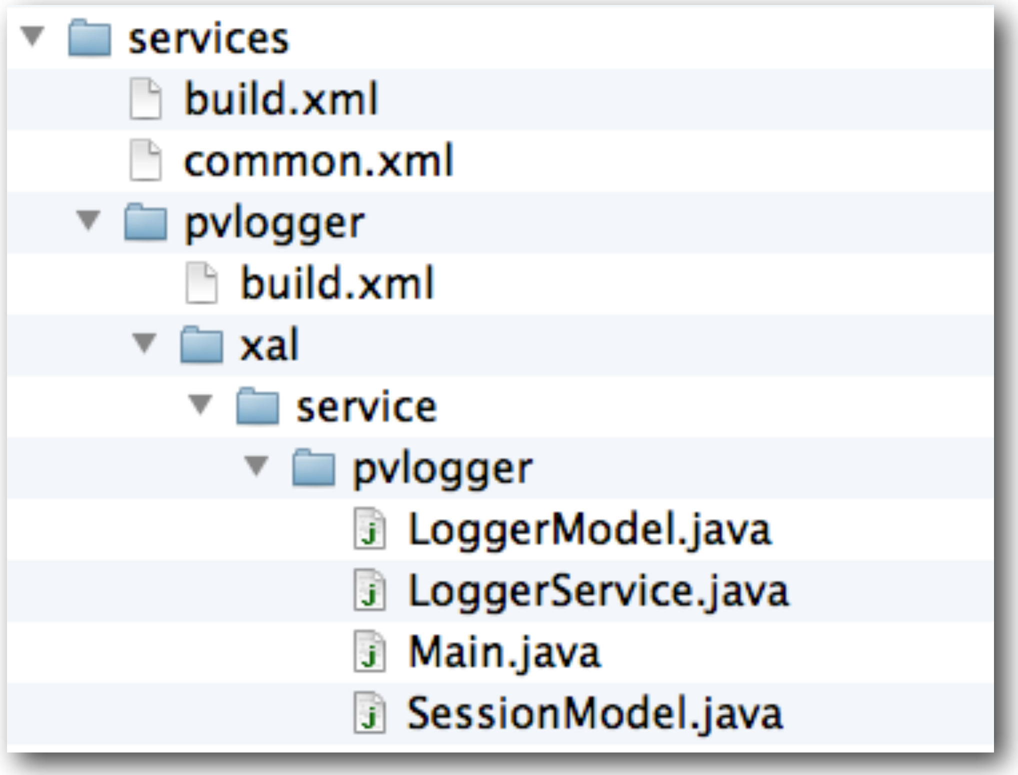
```
▼ pvhistogram
      build.xml
    ▼ resources
      About.properties
      gui.bricks
      Help.html
    ▼ xal
        ▼ app
            ▼ pvhistogram
                  ChannelHistogram.java
                  ChannelHistogramListener.java
                  ChannelSource.java
                  HistogramDocument.java
                  Main.java
```

Core File Structure



```
▼ core
    build.xml
  ▶ lib
  ▶ resources
  ▼ test
      build.xml
    ▶ resources
    ▼ xal
      ▶ model
      ▶ sim
      ▶ smf
      ▶ test
      ▶ tools
  ▼ xal
    ▶ application
    ▶ ca
    ▶ jca
    ▶ model
    ▶ service
    ▶ sim
    ▶ smf
    ▶ tools
      XalException.java
```

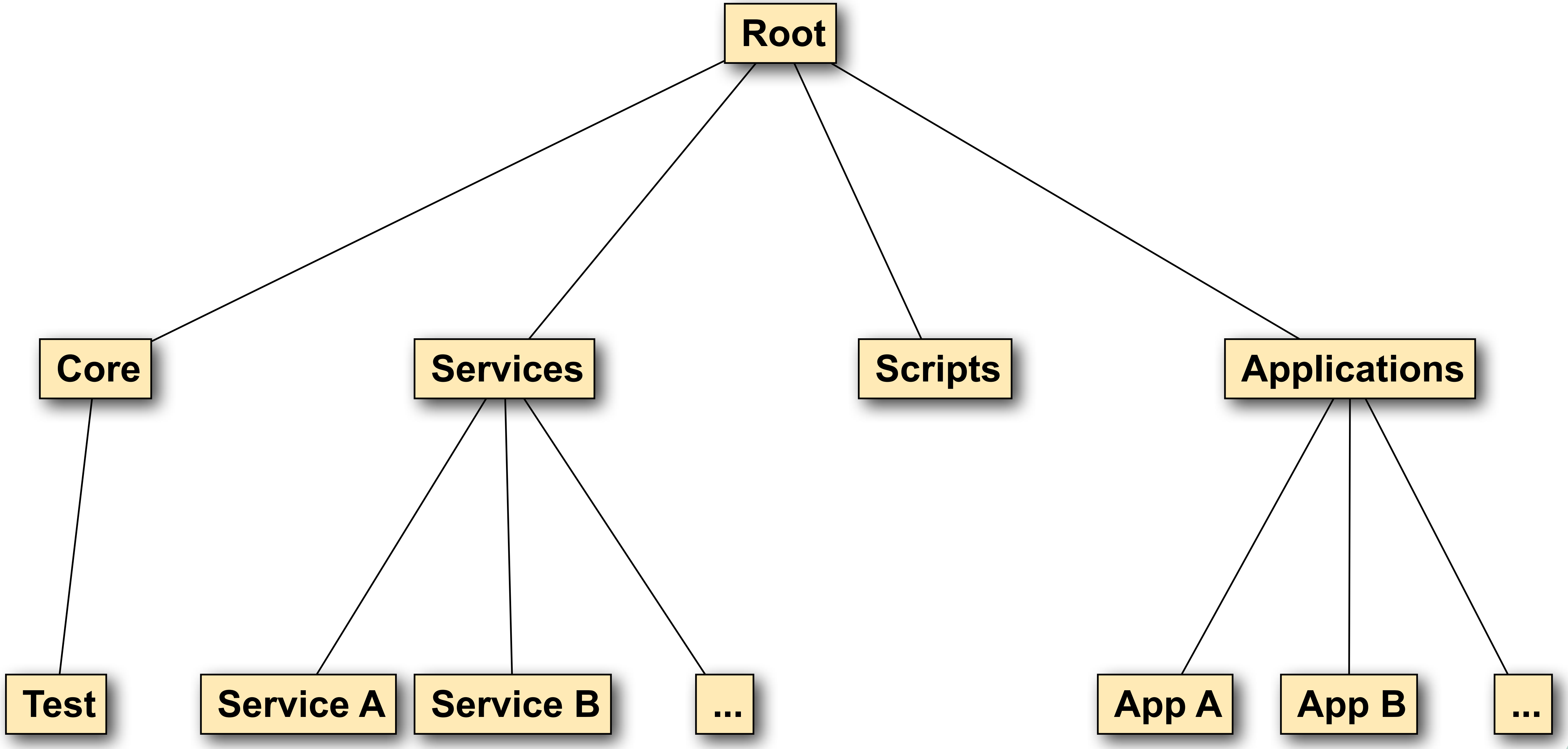
Services File Structure



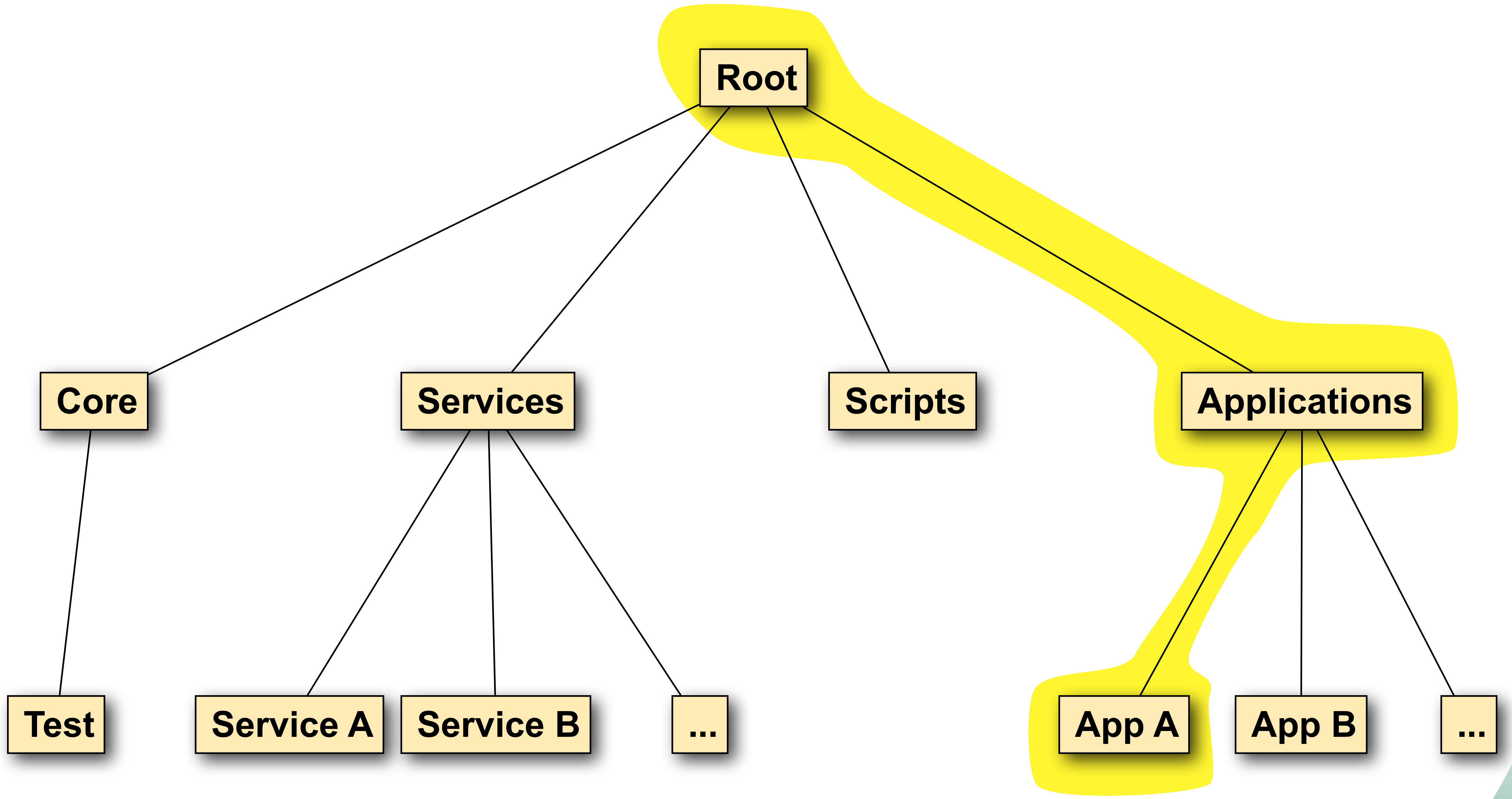
```
services
    build.xml
    common.xml
    pvlogger
        build.xml
        xal
            service
                pvlogger
                    LoggerModel.java
                    LoggerService.java
                    Main.java
                    SessionModel.java
```

Ant Build Tree

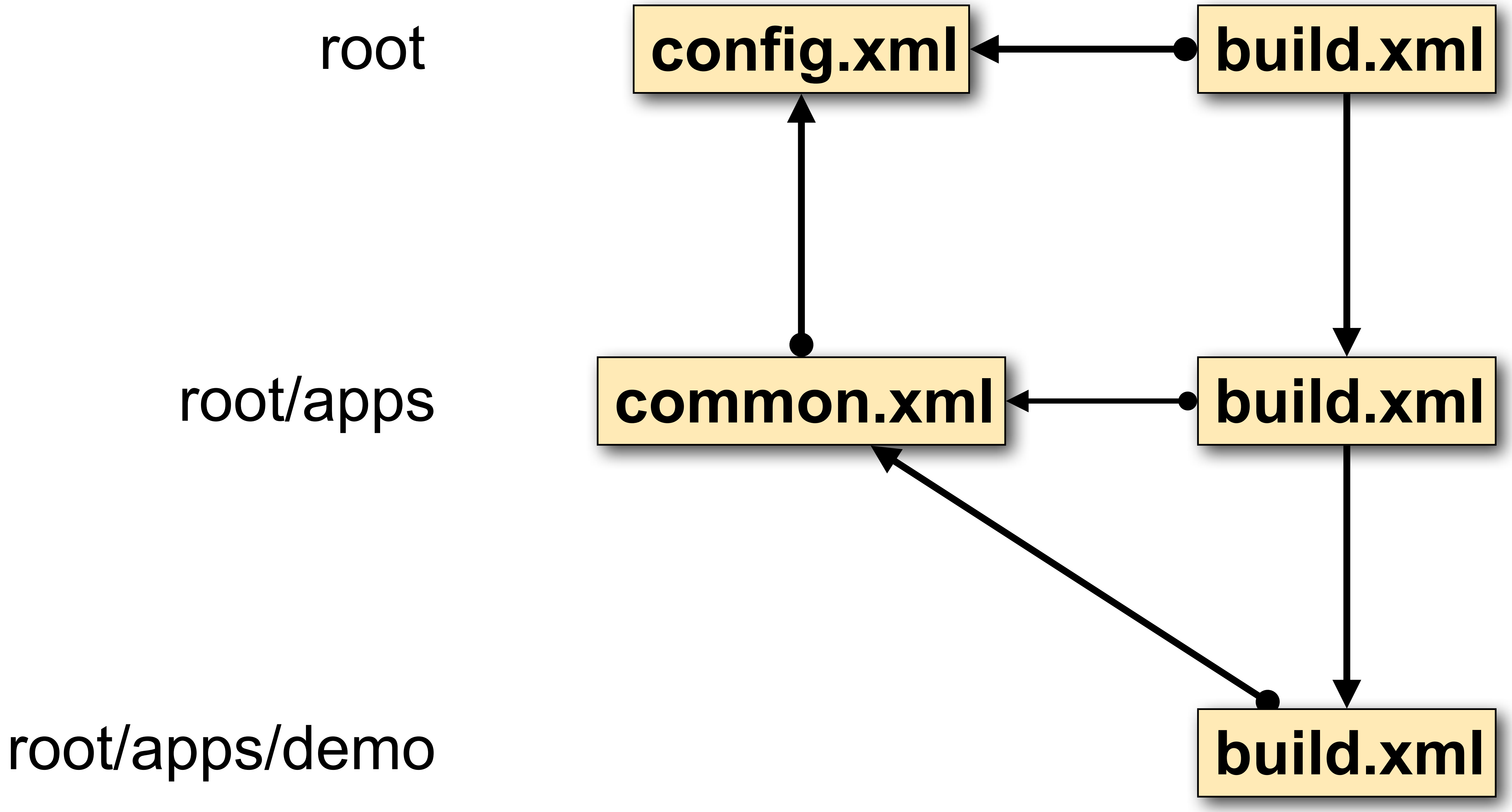
Control Becomes More Detailed



Ant Build Tree
Control Becomes More Detailed



Ant Application Build Hierarchy
“Demo” Application Example



Root Level Build Targets

Target	Description
help	Print help
all	Build core, services, applications; copy scripts
apps	Compile applications and assemble the jar products
clean	Clean compiled classes and build products
core	Compile core classes and assemble the jar products
doc	Build the core javadoc
install	Batch install all build products
install-apps	Batch install applications
install-core	Install the core
install-doc	Install the java documentation
install-scripts	Batch install the scripts
install-services	Batch install services
purge-build	Purge all build products
purge-install	Purge all installed products
run-tests	Build and run unit tests
scripts	Copy scripts to build directory
services	Compile services and assemble jar products
standalone-apps	Batch build applications and assemble jars standalone
standalone-services	Batch build services and assemble jars standalone

Apps Level Build Target

Target	Description
help	Print help
all	Batch compile applications and assemble jar products
all-standalone	Batch compile applications and assemble standalone jar products
force-all	Batch compile all applications ignoring whether they allow batch building
clean	Clean compiled classes and build products
install	Install all build products
purge-install	Purge installed applications

App Level Build Target

Target	Description
help	Print help
build	Compile application and assemble jar product
build-standalone	Compile applications and assemble standalone jar product
clean	Clean compiled classes and build product
compile	Compile reporting all recommended warnings
compile-warn-all	Compile reporting all recommended warnings
compile-warn-mandatory	Compile reporting only mandatory warnings
install	Install the application
purge-install	Purge installed application

Root Batch Applications Build

- ✓ **Rules based build**
- ✓ **Maintenance Free**

```
<!-- Build all the applications which allow batch building. -->
<target name="apps" depends="core">
   <subant target="all">
      <fileset dir=".">
         <include name="apps/build.xml" />
      </fileset>
   </subant>
</target>
```

Application Build
"Demo" Application Example

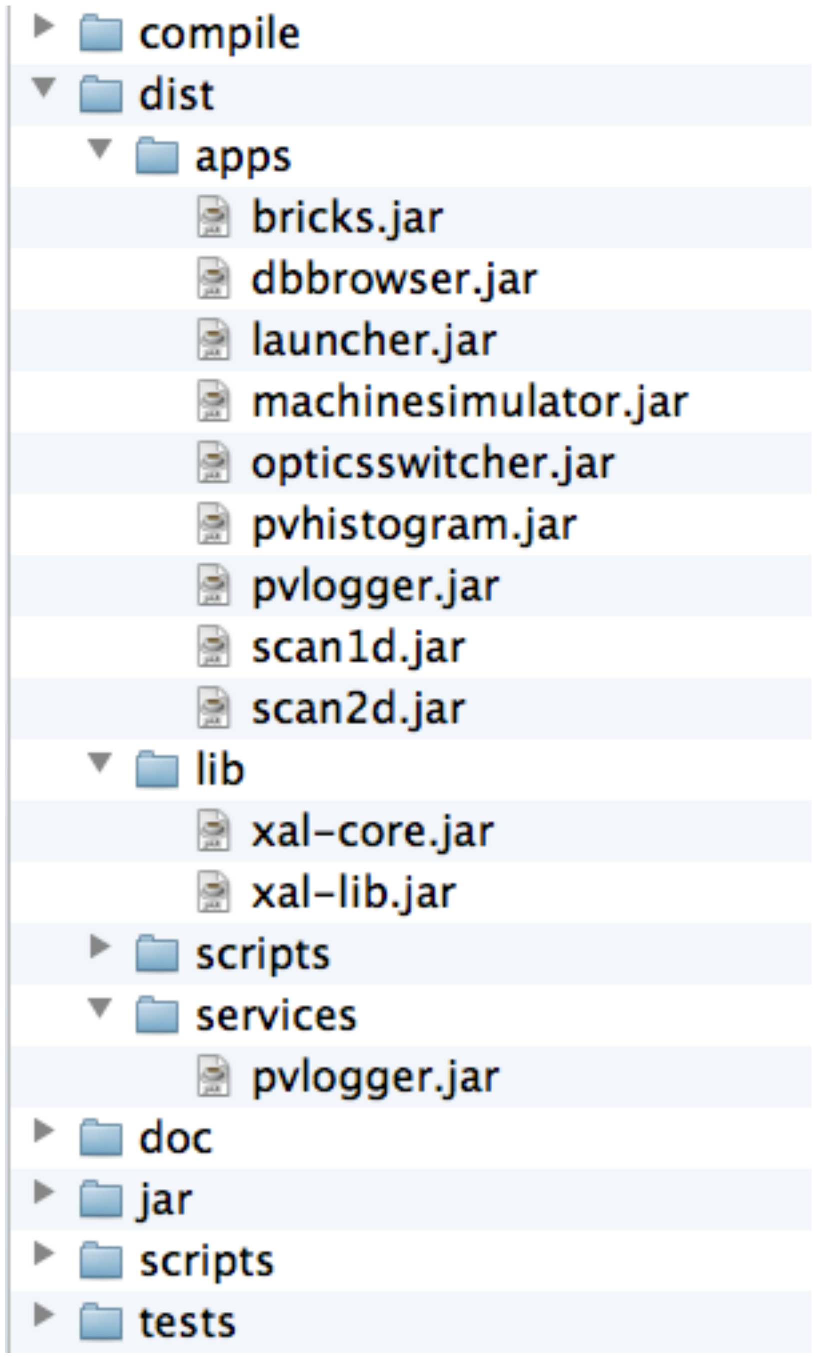
- ✓ Simple build file
- ✓ Option to exclude batch building
- ✓ Option to exclude install

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- Build the application whose name matches that of the project. -->
<project name="demo" basedir="." default="all">
    <!--<property name="exclude.batch.build" value="true" />-->
    <property name="exclude.install" value="true" />

  <import file="../common.xml"/>
</project>
```

Build Directory
Default Configuration



```
▶ compile
▼ dist
    ▼ apps
        bricks.jar
        dbbrowser.jar
        launcher.jar
        machinesimulator.jar
        opticsswitcher.jar
        pvhistogram.jar
        pvlogger.jar
        scan1d.jar
        scan2d.jar
    ▼ lib
        xal-core.jar
        xal-lib.jar
    ▶ scripts
    ▼ services
        pvlogger.jar
▶ doc
▶ jar
▶ scripts
▶ tests
```

Build Configuration Options

Environment Variable	Default Value	Description
JUNIT_HOME	/usr/share/java	Location of junit.jar
XAL_BUILD_ROOT	${xal.home}/build	Location of intermediate and final build products
XAL_INSTALL_ROOT	${build.root}/dist	Location to install products for distribution
XAL_JAVADOC_ROOT	${install.root}/doc	Location to install Javadocs

XAL Service Architecture

- Core Protocols
  - XML-RPC
    - Communication Protocol
    - Transport supports few classes
    - Error on services that are shutdown by call
    - Slow WebServer dependence which has been deprecated and later restored and discouraged
    - Outdated Apache 1.1 with unknown patches
    - Not easily portable to newest version
  - JmDNS
    - Dynamic Registration and Discovery
    - Old version 1.0 plus patch 1473279
    - Incompatible with latest version
- Service interface in service's package

Open XAL Service Architecture

- Core Protocols
  - JSON-RPC
    - Communication Protocol
    - Custom implementation in core
    - Transport supports large number of classes and easily extensible
    - Supports Oneway calls using Java annotation
  - JmDNS
    - Dynamic Registration and Discovery
    - Version 3.4.1 (latest)
- Service Interface in core under xal.service
- Remote Data Cache simplifies synchronization calls

Example Service Interface

```
package xal.service.worker;

import xal.tools.services.OneWay;

import java.util.Date;


/**
 * Demo service interface.
 * @author  tap
 */
public interface Working {
    /** add two numbers */
    public double add( final double summand, final double addend );


    /** get the launch time */
    public Date getLaunchTime();


    /** shutdown the service */
    @OneWay
    public void shutdown( final int code );
}
```

Example Service Implementation

```java
package xal.service.worker;

import java.util.Date;


/**
 * Demo service providing demo work.
 * @author  tap
 */
public class WorkService implements Working {
    /** add two numbers */
    public double add( final double summand, final double addend ) {
        return summand + addend;
    }


    /** get the launch time */
    public Date getLaunchTime() {
        return Main.getLaunchTime();
    }


    /** shutdown the service */
    public void shutdown( final int code ) {
        Main.shutdown( code );
    }
}
```

Example Service Startup and Shutdown

```
/** run the service */
protected void run() {
    ServiceDirectory.defaultDirectory().registerService( Working.class, "Worker", new WorkService() );
}


/**
 * Main entry point to the service.  Run the service.
 * @param args The launch arguments to the service.
 */
static public void main( final String[] args ) {
    new Main().run();
}


/** Shutdown the application */
static public void shutdown( final int code ) {
    ServiceDirectory.defaultDirectory().dispose();
    System.exit( code );
}
```

Example Service Client

```
include Java

import java.lang.System

include_class 'xal.tools.services.ServiceDirectory'
include_class 'xal.application.ApplicationStatus'

services = ServiceDirectory.defaultDirectory.findServicesWithType( Java::JavaClass.for_name( "xal.service.worker.Working" ), 5000 )
puts "#{services.length } services found"
if ( services.length > 0 )
    proxy = ServiceDirectory.defaultDirectory.getProxy( Java::JavaClass.for_name( "xal.service.worker.Working" ), services[0] )

    sum = proxy.add( 5.3, 2.4 )
    puts "Sum: #{sum}"

    sum = proxy.add( -34.2, 75.8 )
    puts "Sum: #{sum}"

    launch_time = proxy.getLaunchTime
    puts "Launch Time: #{launch_time}"

    puts "Shutting down the service..."
    proxy.shutdown 0
    puts "Sent shutdown request..."
end

ServiceDirectory.defaultDirectory.dispose

exit 0
```

Online Model

- Major changes to online model engine
- Scenario API improvements
- Online Model Synchronization performance boost
- Chris will give more details

Online Model Live Synchronization

- 10 to 20 times speedup versus XAL
  - Uses Batch Get Request
  - From seconds in XAL to few hundred milliseconds in Open XAL without existing connection
  - Just few milliseconds in Open XAL with existing connection
- Dramatically simpler code in Open XAL versus XAL

Open XAL Current Status

- Java 7 Mostly Ready
  - It Works
  - No compiler warnings (see TODO: tags)
  - Need to add newly supported generics (e.g. JList)
  - Warning about bootstrap class path
- Core is complete except for formula parser
- 9 deployable Open XAL applications replacing 11 XAL applications
- 1 service
  - PV Logger

Open XAL Roadmap

Target Date	Task	Progress
Oct 31, 2010	Project Creation	100%
Dec 31, 2010	Website Development	100%
Feb 15, 2011	Application Framework Migration	100%
Apr 30, 2011	New Online Model Implementation	100%
Sep 30, 2011	Fix Compiler Lint Warnings	100%
Feb 28, 2012	JSON Framework Implementation	100%
Feb 28, 2012	Common Package Migration	100%
Oct 31, 2012	New Service Implementation	100%
Dec 31, 2012	Common Services Migration	100%
May 31, 2013	Replace Lattice Generator	25%
Dec 31, 2013	Common Applications Migration	10%
Dec 31, 2013	Test Suite Development	1%

Application Porting from XAL
Strategies

Benefit	Straight Port	Port with Modifications	Rewrite from Scratch
New Packages	✔	✔	✔
Fix Compiler Warnings	✔	✔	✔
Fix Bugs		✔	✔
Future Proofing		✔	✔
Feature Enhancement		✔	✔
Eliminate Obsolete Code			✔
Effort	**Low**	**Medium**	**High**

Current Applications

Application	Notes	Status
Bricks	GUI Builder	Complete
DB Browser	Browse database schema	Complete
Launcher	Launch applications and manage live apps	Complete
Machine Simulator	Run online model simulations. Replaces MPX from scratch.	Progress
Optics Switcher	Switch default optics	Complete
PV Histogram	Display histogram of a channel's live values	Complete
Scan 1D	Scan one channel and record others	Complete
Scan 2D	Scan two channels and record others	Complete
PV Logger	Manage PV Logger service and browse archived data	Complete

Special Application Ports

- MPX application replaced with new Machine Simulation application
- SCORE becomes a service with a client front end

Current Activities

- Tasks
  - Port Applications
  - Implement Formula Parser
  - Fix Javadoc warnings
  - Unit Testing for applications and services
  - Unit Test Cases
  - Automated testing
- Project management
  - Issue Tracking
  - Communication

Opportunities to Help

- Bug Fixing
- Unit Tests
- Porting Applications

Code Collaboration

- Source Forge Git Repository
  - Coding Standards
  - Don’t add external jars that have incompatible licenses
  - Abide by third party licenses for code
- Limit master branch to core group
  - We need a process
  - Code must compile without warnings or error and pass unit tests
- Use namespace rules for branching
  - site.sns.master

Coding Standards

http://xaldev.sourceforge.net/specs/SoftwareManagementGuide.pdf

- Self documenting code including verbose, unambiguous symbol names
- Internal and public API documentation
- Standard symbol naming conventions: variables, constants, methods and classes
- Literals should be assigned to and referenced through a symbol
- Follow standard software design patterns
  - Code encapsulation, Adaptor, Model-View-Controller
- Minimize third party libraries

Release Management

Release	When to Increment
Major	Referencing APIs break or major structural changes
Minor	New features that don't break referencing APIs
Patch	Bug fixes for existing features

Release Management Versioning

http://xaldev.sourceforge.net/specs/Versioning.pdf

- Branch prefixed with "release"
- Branches for each major/minor version
  - *release.major.minor* (e.g. release.2.1)
- Tag for patch releases
  - *release.major.minor.patch* (e.g. release.2.1.3)

Project Website
Documentation, Status, Source

http://xaldev.sourceforge.net