XAL Workshop 2010

Open Discussion

- Start from scratch with parallel migration
- Fresh repository (SourceForge)
- Source Code Control (Subversion)
  - Structure of repository
  - Core and Site Specific Sub-projects (Devices, Applications)
  - Individual Doorkeepers for packages and applications
  - Branches for Core only (Development, Pre-release and Production)
  - Mailing List and Blog
- Localization Support (localized resources)
- Quality Control and Regression Testing
- Coding standards document (documentation, formatting)
  - Sample files
  - Online Help
- Task Management (feature requests, active projects)
- Accelerator Optics File
  - XML Accelerator Schema for validating XML
  - Stylesheet for rendering Optics File
- Database Schema Variations
  - driver (configuration file)
  - table and column naming (site specific name mapping)
  - structure (site specific SQL adaptor)
  - declare the required database schema
  - post the current database schema used by XAL
- Current to Field Mapping?
- Alignment
- Organization of the source files
- Package Naming (preface package name with “xal”)
- Project Name: (Open XAL)
- Applications should allow for site variations in database structure
- Easy for physicists to adopt
  - Math toolbox
  - Easy access to modeling protocol
  - Efficient machine experiment support
  - Realistic machine simulation
  - Full event reconstruction offline

XAL Core

- Application Framework
- Tools (possibly split)
- XAL Online Model
- Machine Hierarchy (site specific versions of devices)
- XAL Tools (Lattice Generator, Widgets)
- Channel Access
- Ant Build System

Coding Standards

- Modern Java Convention
  - http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html
  - Constants upper case with underscores separating words
  - Variables and Methods use mixed case (first letter lower case)
  - Packages should be strict lower case
  - Files organization should match package organization
  - Two blank lines between methods, three blank lines between classes
- Code should be readable (descriptive variable and method names)
  - Self documenting
  - Descriptive variable and method names
  - Use line and character spacing for clarity
- Java Bean accessors and setters
- Literals used more than once should be assigned to a constant
- Access should be as hidden (private) as possible
- API should use interfaces (use List not ArrayList) as possible
- All packages, classes and methods should have Java Doc
- No generated source code (javacc?)
- No IDE specific files or dependencies
- Help files either in plain text or HTML + CSS
- Header should identify author/care taker contact