Coding Management

Steve Guest (RAL)

Coding Standards

- For Java, the HCSS coding standards should be used, see "Java Coding Standard and Guidelines for the Herschel Common Science System" Issue 2.0 HSCDT/TN009.
  - Based on Java guidelines from Sun
- A draft document exists for Jython. This is still under discussion, but:
  - Use the same naming conventions as for Java.
  - Calling Java from Jython is transparent and easy.
  - Call Jython from Java - while possible - is less so. This will have an impact on which things are written in each language.

Naming Conventions #1

- All names should use US spelling.
  - Mixing spellings can result in unexpected behaviour e.g. method overload vs override.
  - The Java libraries use US spelling.
- Java package names are all lower case.
  - They should start with herschel.spire
- Class names start with an initial upper-case letter and the first letter of each subsequent word capitalised.
  - Class names should normally be nouns, e.g. CalibrationTable
  - Java Interfaces use the same conventions
- Constants (in Java, static final variables) are in upper-case with words separated by underscores, e.g. MAX_VALUE.

Naming Conventions #2

- Method names begin with a lower-case letter. If there is more than one, subsequent ones are capitalised.
  - The first word should normally be a verb, e.g. calibrateTable, though not always, e.g. length.
  - Use the same convention for Jython functions
- Follow Java Bean conventions for certain methods:
  - setAttribute, getAttribute, isSomething, hasSomething etc
  - This will allow some nice shortcut syntax from Jython.
- Variable names follow the same capitalisation conventions as method names.
  - In larger Java classes, prepend '_' to distinguish class and instance variables from local variables at a glance, e.g. _instance.

Some Notes on Jython Conventions

- Python has no privacy mechanism in the Java sense, but there are some conventions:
  - A variable starting with single underscore is internal to the class and serves as a warning not to use it.
  - A variable starting with a double underscore is a stronger warning that it is private to the class, Jython helps here by "mangling" the name to make it more awkward to use.
  - This is therefore different to the recommendation to use underscores for all class variables in the HCSS coding standards.
- There are a number of “hooks” you can put in Java classes to make them “nice-to-use” from Jython.
  - Using Java Bean conventions is a nice start, but there are other things that can be done, e.g. you can overload operators.

Jython Privacy Example

```
>>> class Xy:
...   def __init__(self):
...     self._x = "xxx"
...     self.__y = "yyy"
...
>>> c = Xy()
>>> print c._x
xxx
>>> print c.__y
Traceback (innermost last):
  File "<console>", line 1, in ?
AttributeError: instance of 'Xy' has no attribute '__y
>>> print c._Xy__y
yyy
```

Configuration Control #1

- CVS is used as a configuration control tool.
  - It is set up to use a central server at ESTEC.
  - SPIRE has a general read-only account.
  - Developers need their own account with write privilege.
  - The adminstrator is Rob Zondag, who will provide a username and password on request.
  - See “Herschel Science Centre Software Configuration Procedures (SCCP)”, version 1.1 HSC/DOC/0212.
- The CVS repository can be accessed using a number of clients:
  - “Classic” command line interface.
  - GUI tools such as WinCVS
  - Browsed using the CVSWeb link at the ESTEC site

Configuration Control #2

- All configured text files should contain - as a minimum - the CVS Id tag in the header, see example.
- Releases of packages and larger systems are made by marking them with a CVS *tag*.
  - Package versions should be tagged as D_SPIRE_<package-name>-<major-version>_<minor-version>. This will include it an automatic build at ESTEC (and, if enabled, running of test harnesses). For example, D_SPIRE_DEGLITCH_0_4.
  - Releases of systems (eg SPIRE_IA) have a different type of tag. There is currently no general scheme but anything can be used, e.g. SPIRE_IA_1_2_3 for example.
- There are separate “main”, “prototype”, and “user-private” areas in the repository.

Example Java Header

```
/* $Id$
 * Copyright (c) 2003 CCLRC
 */
package herschel.spire.qla;

import javax.swing.event.ChangeEvent;
import herschel.ccm.api.Product;

/**
 * DataEvent is used to notify interested parties that the available data has changed. The event also
 * supplies information on the current detector array.
 * @see DataListener
 * @author   S.Guest         RAL/SPIRE
 * @version  November 2003    2.0
 *
 * Change Log:
 * May      2002  first stable version
 * July     2003  change Product import
 * November 2003  SCR-0255: added getArrayType & arrayChanged
 */
```

SPR/SCR System

- The SPR/SCR system is also run centrally at ESTEC, via a web site.
  - A user name and password are required to access the site, but these are well known….
  - One area of the system is used for the HCSS.
  - Instruments have their own areas.
- SPRs and SCRs should only be raised on software that is in the CVS system.
- We will need an ICC CCB to prioritise and disposition.
  - Currently this function is performed by the ISDT.
  - The OBST should also be involved.
  - CALT and OPST should be involved later.

Documentation

- Document Java code with Javadoc. Note that Javadoc comments are written in HTML format.
  - Jython files should contain __doc__ comments.
- Livelink versus Web site
  - Which documents should be where?
- Technical notes should normally be provided to describe each IA module in general and algorithmic terms.
- User guide/on-line help
  - Documentation of GUI components should be suitable for both printed documentation and online help => HTML.
- Design documentation
  - Always? Or just when there is some complexity in the design?

Test Harnesses

- Packages that become a part of the HCSS distribution (like IA) need to have a test harness.
  - It is good practice anyway.
  - They will be automatically run as part of the HCSS build.
  - They should normally go in a “test” subpackage.
  - Provide an AllTests class to run everything.
- Unit tests are written using the JUnit tool.
  - It is prohibitively difficult to write them for GUI components, don’t attempt it.
  - Do however, try to decouple the GUI from the underlying logic.
- It’s tempting to make them an afterthought, but:
  - They are much easier to write at the same time as the code.