Minutes of Meeting of the DP Documentation Working Group #4

Hassan Siddiqui

1. General

Document ID	HERSCHEL-MOM-0680
Date of Meeting	3 November 2005
Venue	Telecon

2. Present

- Rik Huygen (RH)
- Hassan Siddiqui (HS)
- Steve Guest (SG)
- Javier Correles Garcia (JCG)

3. Introduction

The ordering of sections in these minutes do not necessary correspond to the chronological order at which they were discussed during the telecon.

4. Previous Action Items

Outstanding actions from previous meetings were reviewed. It appears that only one action item was outstanding:

AI 150205/4

HS to revise DP documentation/code release schedule

The DP-DOC-WG agreed to *close* the above action, based on the following information HS provided prior to the telecon:

1. 2 weeks before HCSS code freeze: UM and FAQ updated
2. 1 week before HCSS code freeze: DP internal code freeze. Documentation and ia.demo updates allowed only
3. HCSS code freeze in place; no further updates
4. HCSS system tests complete. Code freeze lifted. DP iteration x released.

5. Hiding private documentation of classes

and methods from Javadoc

This issue has been tracked via SCR-0454: "javadoc should hide documentation which is not part of public API" (see this link), and a DP-CCB action item:

AI 140605/1

RH, as part of the DP documentation group, to make a counter proposal how documentation for private classes, methods, etc, can be hidden in javadoc

RH presented his analysis of the situation. This is best summarised in these minutes by an excerpt of his recent addition to the analysis section of in SCR-0454, provided below. The final agreement by the DP-DOC-WG is to introduce the yDoc tool into HCSS builds. HS promised to request this to be placed in the system.

The question is how to exclude certain information from the javadoc at package level, but also classes, methods and even fields.

1. Excluding packages: This is possible using the javadocexclude directive in the jake.config files. You can specify any (sub-) package that should be excluded from the javadoc generation process.

2. Excluding classes: The exclusion of classes should normally be done by assigning proper modifiers to your classes. You can make your classes package friendly (meaning the class can be accessed only from within the package) by *not* specifying any modifier. Unfortunately, the guidelines for package structure in the HCSS contradict this rule since public APIs should be placed inside a sub-package called 'api', while implementations should be put into the sub-package 'impl'. In order to make this work, classes in both packages should have public access state. A sub-package 'impl' could however be excluded using the javadocexclude as described above.

3. Excluding methods and fields: For methods and fields the same guideline applies to use the normal Java access modifiers to define the access level of your methods. This is however very restricting since we don't want to see e.g. all jython hooks that are build into our code. Part of this can be solved by extracting interfaces for these methods, but there are surely occasions where we can not do this in a reasonable way. There is a need to exclude methods from the documentation even if they have public access state.

The first recommendation is that developers should re-visit their packages and classes and check if sub-packages and classes can be made non-public in order to remove them from the javadoc. Second we need a mechanism to make methods and field disappear from the javadoc. Several options were investigated:

1. Write our own Doclet/Taglet/Filter classes to exclude parts of the javadoc. This is a work package that needs considerable effort, while (see option 3) it has been done for us by others.

2. Use the Java 1.5 annotation mechanism We still need to write our own classes to cope with these annotations.

3. Use a third party library, like yDoc. yDoc was purchased and (again) investigated for this purpose. The (single user) license costs 138 EURO. This product implements some useful Doclets/Taglets and Filters for javadoc. One of the filters implements exactly the request made by this SCR, i.e. if you put the tag @y.exclude anywhere in your javadoc (for class, method or fields) that piece of javadoc will not be used during generation. yDoc therefore fulfills the requirement. It is however not needed for all developers to have a yDoc license since the standard javadoc tool will simple raise a warning and ignore the unknown tag. In addition, yDoc provides for each class in the javadoc a class diagram with dependencies and associations, showing inheritance etc... One thing I did not investigate (amongst many others :-) is the performance of this tool against normal javadoc.

6. Document Structure for Instrument-Specific Needs

Peer Zaal in an email exchange on 3 Nov reported concerns related to the organisation of documents visible from the online help facility provided by JIDE. For his HIFI users, he needs a different structure to that presented by the core DP documentation framework. This issue is related to SCR-1687: "It should be possible to move the HCSS main entries of help" (link)

HS mentioned that this issue was raised in the earlier Editorial Board telecon at the end of October, and the conclusion was that this issue should be delayed until after the DP-ALL effort of getting a consolidated build of documentation, software and data is complete.

However, RH felt that a lot of Peer's immediate problems could be solved by making modifications to the core DP documentation framework at HIFI. RH has made similar modifications for the PCSS. RH agreed to discuss this issue directly with Peer at an upcoming visit to Groningen. This resulted in the action:

AI 031105/1

RH to discuss with Peer Zaal HIFIs problems with organising their documentation from the JIDE help facility. Due 30 Nov 2005.

SG felt that, in spite of the DP-ALL effort, the DP documentation framework was very inflexible and needs to be made more flexible to address future needs by the ICCs. JCG and RH took an action to see ways of improving the flexibility:

AI 031105/2

JCG/RH to look into ways of improving the flexibility of the DP documentation framework with respect to the organisation of documentation as seen by users through eg the JIDE help facility. Due: 31 Dec 2005.

7. Referencing javadoc using jtags

This issue has been tracked via SCR-1509: "help framework should allow @see and @link usage within jtags" (see this link).

It was agreed in the DP-DOC-WG telecon on 29 June 2005 that there can be references from the Users Reference Manual (URM) to developer documentation such as javadoc, and such references can be facilitated using jtags (jtags enable user documentation to be embedded into source code).

What had not been clarified in that telecon, and the main reason why SCR-1509 remains open, is whether developer documentation can reference a section of a user document such as the Users Manual or the URM. HS preferred that user documentation is not referenced by developer documentation, but this was not agreed by SG or RH. RH pointed out that it might be difficult to reference user documentation from javadoc if the user documentation is automatically generated by an XSLT processor. Such a processor splits a document in individual files for each chapter and section. The files are named using the chapter and section numbers. When the author of a document reshuffles the sections or chapters there is no way to update the links in the javadoc. RH proposed that references from javadoc to user documentation are made only to the document as a whole, not to individual sections. There might however be a possibility to use the ID attribute in a chapter xml tag. This is because if a chapter has an ID attribute, this ID will be used as the filename.

It was agreed that referencing of user documentation from javadoc would only be to the entry point of the user document in question. For example, for the User's Manual, the entry file in HTML is the file um.html . For the URM, it is the file urm.html .

8. Auto-generating the Product Definitions Document

The Products Definitions Group, in their meeting on 14 Oct, proposed a Product Definitions Document to specify the structure of all Products used within the DP systems. They felt that a lot of the contents of the document could be auto-generated, and asked Javier to investigate.

HS assumed that the Product Definitions Document is really an ICD, defining the structure of all Products within DP. Therefore it is a matter of defining a schema (eg an XML schema) under which the structure of all Products could be defined. Product specifications written in accordance to this schema (eg XML documents) could then be used to generate the user-readable documentation.

SG, as member of the Product Definitions group, pointed out that the Product Definitions Document is really intended to be a reference guide for users accessing products. So such a document would need to specify the methods that can be used to access the data, in addition to specifying the structure for each product. Ideally, the method documentation would be auto-generated from the code itself.

HS felt that it was no longer clear what the Product Definitions group wishes were, and would seek clarification from the HPDG chair.

9. DP document framework: status

HS provided a short status report of missing functionality still to be included in the DP documentation framework. These include:

- Instrument documentation: how to use/adapt the existing DP framework to cater for ICCs (see Section 6, “Document Structure for Instrument-Specific Needs”)
- The generation of the DP developers documentation from constituent contributions from package custodians.
- A means of including code snippets into documentation
- Other categorizations of functions in the URM, beyond a simple alphabetical order

RH pointed out on this subject issues related to javahelp. He mentioned that there are problems related to full text searches, and context sensitive help problems. He will discuss these issues offline with JCG.

10. Developers Manual

HS provided the current status of the developers manual: still a lot of work to do. He mentioned that the original plan of provided instructions for developers in raising and analysing SxRs will be moved instead to a separate guide provided by the HSCDT QA personnel.

It was asked whether Rik's DocBook Authors guide will be made available. This resulted in the action:

AI 031105/3

HS to provide a link to the DocBook Authors Guide from the DP web pages. Due 30 Nov 2005.

11. Doc Framework Review

HS, following a request by Nicola DeCandussion asked the group whether they felt a full review of

the document framework is needed. The response was fairly lukewarm. SG felt that it could have value, but only if the recommendations made during that review were followed. RH and JCG were not sure if it is necessary.

The conclusion was not to conduct a formal review.

12. AOB

Date of next meeting: Thu 15th December 2005, at 2pm CET.

13. Post-Meeting Notes

On 24 Nov, RH reported that AI 031105/1 (RH to discuss with Peer Zaal HIFIs problems with organising their documentation from the JIDE help facility) closed with the following input:

During my visit to SRON 21-23 Nov I discussed this issue with Peer and explained how he can adopt the javaHelp system files (jhelpset, jhelpmap, jhelptoc) to statically change the structure of the HIFI on-line documentation. We worked out a simple example for the QLA user manual.