FUDA EMPIRICAL EVALUATION
Development Experience Questionnaire (Used Templates)

NOTE: This is a fillable PDF file. If you are not using a PDF writer, please make sure to print this file to a PS or PDF file to not lose the information.

Name: S3 Date: 20/08/2008

Concept: ☐ JFace – Context Menu [✓ JFace – Content Assist] ☐ Eclipse – Navigate ☐ Eclipse – Table Viewer

Q.1: Were you able to implement the concept successfully? [✓ Yes] ☐ No

Q.2: How much time did you spend on the concept's implementation? 48min

Q.3: If not successful to implement the concept, what was the main reason in your opinion?

- ☐ Lack of experience.
- ☐ Not a useful template.
- ☐ Not useful sample applications.
- ☐ Complexity of the concept.
- ☐ Other. Please specify: ______

Q.4: Did you refer to the example applications' source code to implement the concept?

☐ No. None of them. [✓ Yes. One of them.] Please specify: JavaEditor ☐ Yes. Both of them.

Q.4.1: If yes, for what program statements and what kind of information?

```
return new char[] { '.', '(' };
from JavaCompletionProcessor.getCompletionProposalAutoActivationCharacters()

ICompletionProposal[] result= new ICompletionProposal[fgProposals.length];
  for (int i= 0; i < fgProposals.length; i++) {
   IContextInformation info= new ContextInformation(fgProposals[i],
MessageFormat.format(JavaEditorMessages.getString("CompletionProcessor.Proposal.ContextInfo.
pattern"), new Object[] { fgProposals[i] }));
   result[i]= new CompletionProposal(fgProposals[i], documentOffset, 0, fgProposals[i].length(), null,
fgProposals[i], info,
MessageFormat.format(JavaEditorMessages.getString("CompletionProcessor.Proposal.hoverinfo.pa
ttern"), new Object[] { fgProposals[i]})); /* FRL_30 */ }
  return result;
from JavaCompletionProcessor.computeCompletionProposals() to see how to construct the
proposals.

IDocument.DEFAULT_CONTENT_TYPE
from JavaSourceViewerConfiguration.getContentAssistant() to see an example value for FRL_16
```

Q.5: Overall, did you find the templates useful? If yes, in what way? If not, why?

Yes, they were useful; however, they could be even more useful if there was some 'implementation strategy'/'suggested use' in the templates tutorial. The recepies would be harder to use without having FRL_n annotations in the example applications.

Q.6: Do you think that the format and structure of the templates are OK? If not, what are the main issues?

A minor issue: when copying the template into the Java editor, the FRL_n comments are disturbing the natural indentation of the code and it takes some time to get organized and understand the structure of the copied code. I suggest putting the comments at the end of the lines.

Q.7: What kinds of information do you think are missing in the templates?

FRL_29 could provide 'initializeEditor()' as a possible callback method for putting the call into. Similarly, FRL_30 could provide 'computeCompletionProposals()'. I needed to navigate to the example apps to figure that out.

Q.8: Overall, in the range of 1-5, how do you rank the provided template in terms of usefulness to implement the concept?

☐ 1 = Not Useful ☐ 2 ☐ 3 [✓ 4] ☐ 5 = Excellent

Q.9: Do you have any additional comments on this experiment?

Navigation to example apps is critical and a frequently performed step. I used it to see example values of method call arguments. Also, when I run the application, I got a null pointer expression because getConfiguredDocumentPartitioning() was returning null (FRL_23). The method does not have to be implemented by the user (the default implementation is sufficient).

Additional Space: