CSCIE-275

Guide for Chief Programmers

Serguei Khramtchenko
Apr 2006

1. Preface 3
2. Design walkthrough meeting 4
2.1 Choosing features for an iteration 4
2.2 Preparing design for walkthrough meeting 4
UI design 5
UO design 5
EX design 5
2.3 Publishing design for developers 5
2.4 Conducting the walkthrough meeting 6
2.5 After the meeting 6
3. Design review meeting 7
3.1 Preparing for design review 7
3.2 Conducting the meeting 8
3.3 After the meeting 9
4. Code review meeting 10
4.1 Preparing for code review 10
4.2 Conducting the meeting 10
4.3 After the meeting 11
5. Promote to build process. 12
6. Appendix 1. Using FDDPMA to create workpackage worksheet. 13
7. Appendix 2. Tools to create the documentation for design or code review. 19
7.1 Using java2html converter. 19
7.2 Using MS-Word. 19
7.3 Using Jupiter plug-in for Eclipse 19

1. Preface

This document was created for Harvard University course “CSCIE 275: Software Architecture and Engineering “. This year the course is experimenting with FDD methodology. We use FDD to implement the course project: UML building tool.

This document describes the activities that Chief Programmers (CPs) will have to perform for the course project, as well as some helpful tools. The document assumes that the reader is familiar with Feature Driven Development (FDD). This document’s organization revolves around the meetings that development teams have during FDD iterations: design walkthrough, design review and code review. The following sections discuss what CPs should do before, during and after each of these meetings.

We have two competing teams; each will implement the same functionality. Each team consists of 9-11 developers, split into the three sub-teams. The sub-teams are working on the different layers of the application:

- User Interface (UI): implementation of GUI
- UML Objects (UO): implementation of core business logic of the application
- Externalization (EX): Persistent storage in the database, forward engineering to XML, java code generation.

We decided to have incremental releases for the project. It requires the teams to have a deployable working code at the end of each iteration, so that a customer may start using it early.

2. Design walkthrough meeting

CPs responsibilities include choosing features to be implemented during the iteration, preparing the design for the meeting, publishing it for developers, conducting the meeting, and addressing the issues raised during the meeting. Keep in mind that the CPs are developers. They do write code. However, due to additional responsibilities described in this document they usually have fewer features to implement.

2.1 Choosing features for an iteration

Typically, the project manager and chief programmers chose the features together. The customers may influence the decision, if they want some particular features to be implemented first. When choosing features, take the following into account:

- The chosen features should not depend on any other features that have not been implemented yet;
- The chosen features may depend on other features that will be implemented in the same iteration.
- In order to comply with a requirement to have incremental releases, choose the features that form a complete piece of functionality. This may require coordination between all the CPs in a project team.
- Each CP chooses features that are relevant for his sub-team. In our case the CP of UI team will chose all UI-related features, the CP of externalization team will pick up all the database related features etc. This subset of features forms a workpackage that CP's team will be working on.

2.2 Preparing design for walkthrough meeting

A CP should prepare design for the meeting. The workpackage design is a set of documents, diagrams, descriptions that explain what is to be implemented. The CP creates the design document by going through each feature in a workpackage to record the answers for these questions:

- What the feature is about? In other words, what customer expects to see when we say this feature is implemented? CP may clarify this with project manager/customer. CP should record his/her notes to publish them later.
- How this feature will be implemented? The answer for this question should be detailed enough, so that the developers have clear understanding of steps/algorithms to do during implementation.
- Who is responsible for implementation of the feature? This question is about distributing the work among the developers. Use code ownership to identify the person(s) who will be responsible for the feature. There are may be several developers who work on a feature. However, one developer usually writes unit tests for the feature.

The CPs of different teams (UI, UO, EX) will produce quite different design.

UI design

The design should be based on the mockup screen created during phase 1 of the project. The design for each feature will typically include

- Image of the UI screen/dialog/control that needs to be implemented (hand-drawn or electronic)
- Possible error messages to be displayed to the user
- Identified objects/methods in the UML Objects that will be used to push/pull data. A UI CP has to communicate this information to a CP of UO team.
- Implementing classes checked into CVS. These classes will usually be empty (no java-doc, methods, or attributes) for the first iteration. The following iterations will create some new classes, and may elaborate on existing classes. When creating a new class, assign an owner who will maintain it.
- Defined scope for unit tests

UO design

The design should be based on the project class diagram, created during phase 1. The design for each feature will typically include:

- Implementing classes checked into CVS, class owners are assigned. Unlike in UI team, CP will usually declare stubs for the methods required to implement the feature. Alternatively, CP may include a sequence diagram.
- Optional sequence diagram(s) that explain the sequence of method calls required to implement the feature. The diagram may be omitted if CP discovers that the feature is easy to implement.
- Checked and unchecked exceptions that may be thrown by the methods.
- Identified methods in the Externalization interface that will be used to store/retrieve data. A UIO CP has to communicate this information to a CP of Externalization team.
- Defined scope for unit tests

EX design

The design should be based on the project class diagram, created during phase 1, as well as on input from UO team. The design for each feature will typically include:

- New elements in the database schema (tables, columns etc)
- New methods in the interface for externalization layer, as requested by UO team.
- Exceptions that may be thrown during operation.
- Defined scope for unit tests

2.3 Publishing design for developers

The result of the design activity should be a worksheet for a workpackage. Each developer receives a copy of a worksheet before the walkthrough meeting. The worksheet will be used through out the iteration:

- Developers read it before the walkthrough meeting to get familiar with design
- Developers use it during the meeting to make notes
- Developers use it during implementation as a checklist of things to do.

The workpackage worksheet includes this information:

- Workpackage description
- List of features in the workpackage
- Description for each feature
- Planned and actual milestones for the iteration. The actual dates will be initially empty.
- A list of attached documents: class diagrams, sequence diagrams, memos from the customer etc.

The Appendix 1 describes how to create a workpackage worksheet with FDD project management application (http://www.fddpma.net)

2.4 Conducting the walkthrough meeting

The preconditions for the meeting are:

- CP has created and published workpackage design
- Developers have read the design and have a paper copy of it.

During the meeting, the CP goes through the list of features and describes the design. The developers ask questions and make notes to be used later for implementation. It often happens that developers have difficulties grasping the design for a particular feature. What seemed easy to the CP does not seem easy to the developers. The team may create a sequence diagram, a UI mockup screen, a fragment of DB schema that clarifies the questions. The CP will later attach it to the workpackage.

2.5 After the meeting

- CP updates the worksheet to include the artifacts created during the meeting.
- Developers proceed to design phase.

3. Design review meeting

The design review preparation starts when developers complete designing features for the workpackage. In general, complete design means that the developers created and documented classes, methods and attribute required to implement a feature, but the implementation is not yet provided. The approach for design may vary between the teams. For example, Externalization team may include database schema as part of the design documentation, while UI team may include some screen mockups. The workpackage design is out of scope for this document, the rest of this section assumes that the design is already complete. The design review process may differ depending on whether or not the development team is local or distributed.

For the design-review meeting a CP should ensure that developers have shared their code between team members, and have reviewed the code. The CP is responsible for conducting the meeting itself, and takes a decision whether the design is acceptable or not.

3.1 Preparing for design review

Preparation for the meeting applies to both CPs and developers. Everyone in the team should create a document that includes his or her design work. In the Java development environment, there have the two options to do it: using java-doc, or java code itself; both have pros and cons. The important note applies to both options: the newly created design MUST be highlighted. This allows reviewing only the changed part of the code and saves time in the subsequent workpackages.

- **Creating java-doc documents**.
  Create the documentation by running the javadoc compiler. It will create a set of HTML files. The advantage of this approach is that javadoc compiler verifies that links and tags are correct. Additionally, javadoc does not tempt the developers to dive into coding details during design review. However, javadoc comes without printed line numbers, which is less convenient during design review meeting. It also comes in the form of separate HTML files. The developers may print the HTML in different order, which complicates design review meetings.
- **Using java code for review.**
  The advantage is that the code comes with line numbers. It also gives the ability to look for inline comments. Java-doc usually describes the contract of the method, not its implementation. The developers often describe their intended implementation using inline comments. This comments would be lost should we choose java docs.
  Additionally, java code allows for looking up the previously implemented features. This comes in handy at design review meetings, when there is a need for a reference to the previously done features. This eliminates the need to go back to a computer.

The RECOMMENDED way is to use java code for design review. We agree not to scrutinize the code, if it already exists. However, suggestions that may save someone's time are welcome.

Each developer should prepare a single document that contains all of his/her work. The typical way to create such a document in the local development team is the following:

- Print out your java code
- Take a pen and highlight the portions of the code/java doc that are relevant for the current workpackage.
- Make as many copies of the highlighted code as many developers are in the team.
- Hand the copy over to every developer.

It is more challenging to share the design documents in the distributed development team, which we are. Typically, we need to create a document in the electronic form, or use some tools to perform the remote review. Appendix 2 describes some tools that can help creating such a document. Once the document is created, we can use one of the following means to share it between team members:

- Send it to all team members via email
- Attach it to the workpackage in the FDDPMA tool.
- Check it in a special folder in CVS.

Each team may choose the mean it likes.

The design documents should be distributed well in advance before the design review meeting. Please allow AT LEAST one day for the peers to review your work. While reviewing the design documents, developers MUST mark the questionable parts of the design, and make some notes about the better ways to design it. Depending on the format of the design meeting, the notes may be in the electronic document, or on the printouts (recommended 2 pages per sheet). The developers come to the design review meeting with a copy of their own design document and with the notes for others.

3.2 Conducting the meeting

A CP drives the design meeting. The meeting should be short, since all the developers have reviewed each other's design documents. If some comes unprepared the CP should reschedule the meeting. The meeting is about telling the found problems to the code owners. These are the procedures of the meeting:

1) CP chooses the design of one of the developers to be reviewed (or asks for a volunteer)
2) CP goes through the document, loudly saying page numbers: "Page 1, Page 2, Page 3…"
3) Someone, who has notes on the page, may reply: "Wait, I have a comment/question on page 2, line number 127".
4) Team discusses the issue. The outcome can be one of the following:
   4.1 The issue needs fixing. The owner of the code makes a note in his/her copy of the design document.
   4.2 This is not an issue. It may be that whoever made the comment, simply misunderstood the design. It may also happen that newly proposed design has equal merits to the existing one, so no changes are necessary.
5) Repeat steps 1-4 until all developers' designs have been reviewed.

At the end of the meeting CP takes a pass/failed decision regarding each developer. There are three possible outcomes:

- Design is accepted as is;
- Design is accepted with minor problems. The developer has to fix problems after the meeting;
- Design is not accepted. The team chooses the date for the next design review meeting.

The RECOMMENDED format for the design review meeting is the live meeting. It is much more productive when everyone is in the same room. For distributed teams it is acceptable to perform the design review during a conference call. It is counter-productive to discuss the design issues via email, or using instant messaging without voice support.

3.3 After the meeting

The developers should modify the code/schemas/mockups etc to fix the found problems. A CP should enter the date of the meeting into the FDD tracking tool.

4. Code review meeting

The code review meeting is very similar to that of design review, described in the previous chapter. This section only describes the processes that defer from the design review.

For the code-review meeting a CP should ensure that developers have shared their code between the team members, and have reviewed the code. The CP is responsible for conducting the meeting itself, and takes a decision whether the code is acceptable or not.

4.1 Preparing for code review

The preparation starts when developers have completed the coding efforts. Typically, we review the code itself, database scripts, jsp pages, Hibernate's mapping files etc. In other words, we review all the artifacts created in an iteration.

During first couple of iterations we also review the code for unit tests. This ensures that all the team members develop the best approach for unit testing. It is up to each team to decide whether the unit tests must be running error-free for the code review meeting. In our project schedule, we have rather long gap between the code review meeting and start of the next iteration. The teams may decide to utilize this gap to make sure unit tests are running. In any case, it is useful to review the unit test code to see the intended testing scenarios. It is up to the CP to decide if the unit test code should be included into the review process.

In the design review, we had an option of what to use for review: java doc, or java code. For the code review, we always use java code. See Appendix 2 for the list of tools that may help sharing the code between the developers.

4.2 Conducting the meeting

The meeting procedures for code review are the same as for design review. Here are a couple of extra hints about the process

1) We should expect to have people of different level of expertise in the team. It may happen that some developers do not get any comments, as their code is nearly perfect. Less experienced members may get tons of comments. Be merciful to each other. Use correct language. Do not take comments personally.

   An example of negative comment: "Who in their right mind would write such a code? It is a piece of garbage!"
   An example of positive comment: "This code will be shorter and more efficient if rewritten like this…"

   Do positive comments. Remember, our goal is to produce quality software, not to upset someone.

2) The code review process is where we learn from each other. It plays the same role as pair programming in “eXtream Programming” method. Please utilize this unique opportunity to learn the best programming practices, usage of patterns etc.
3) During 3-week long iteration, the developers may produce a lot of code. It is quite common to see 20-30 pages of code for every developer. If a team consists of only three developers, we have 60 to 90 pages to review. This is rather tough goal for the 45 minutes that we have for code review meetings. Please focus on the goal.

4.3 After the meeting

The developers should modify their code to fix the found problems. If the code was accepted, a CP should enter the date of the meeting into the FDD tracking tool. If the code was rejected, a CP should reschedule the meeting.

5. Promote to build process.

This section describes the last activity for an FDD iteration: promotion to build. The Chief Programmer promotes the code to build. The developers do not participate in the activity, unless unit tests reveal problems in their code.

The process consists of the two parts:

1) Make sure that all the unit tests are running. This includes all unit tests: those written for previous iterations and for the current one, those written by other sub-teams, and by the sub-team that completes its iteration. If some unit tests fail, CPs may either fix the problems themselves, or find the responsible developer whose code is failing. Quite often, the CPs of different teams have to communicate to figure out the source of failure.
2) Promote the code to build. This process has to ensure that the promoted code is readily available at any time in the future. It is quite common that the start of next iteration may break some existing code. However, the promoted code should be available for the customer to play with.

   It is quite easy to promote the code using code repositories. For example, in CVS one would create a tag with a predefined name, such as “BUILD-15MAY2006”. Should the need arise, the team may return to this build by checking out code using “sticky tag BUILD-15MAY2006”

This concludes the FDD iteration. Time to celebrate!

6. Appendix 1. Using FDDPMA to create workpackage worksheet.

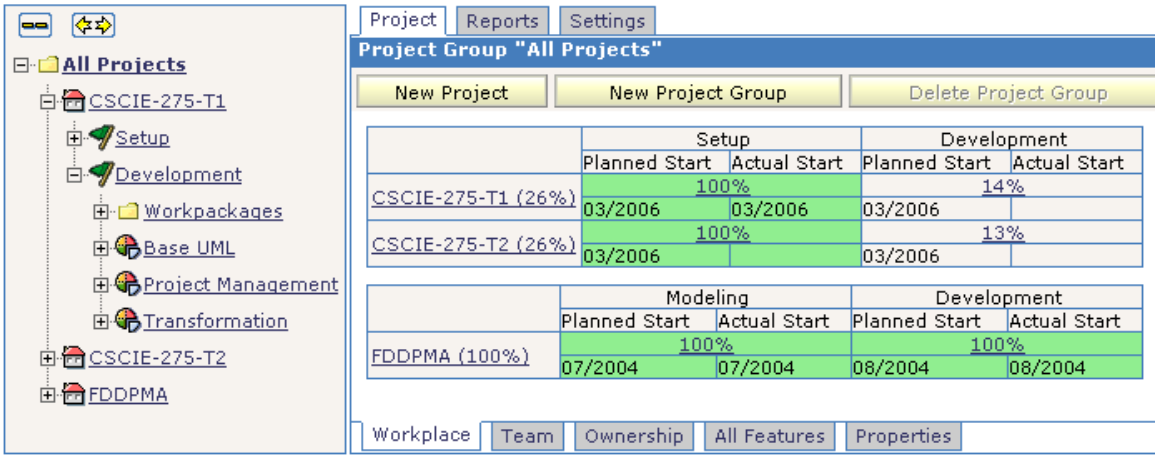
This section describes how to create a workpackage using FDD Project Management Application (FDDPMA). The application is web-based, therefore publishing the workpackage's worksheet comes for free.

This section has several prerequisites:

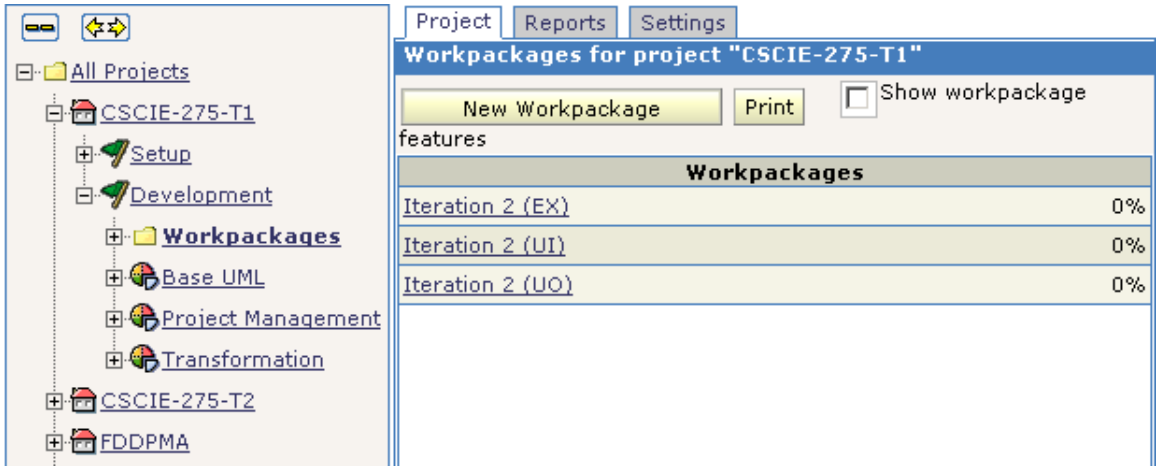
- The reader is familiar with FDD
- The reader has read FDDPMA user guide
- The project setup is complete. This means that a project was created in the FDDPMA application, a project has features assigned to subject areas and business activities.
- The list of features for current iteration is known.

The rest of this section is a sequence of steps required to create a workpackage.

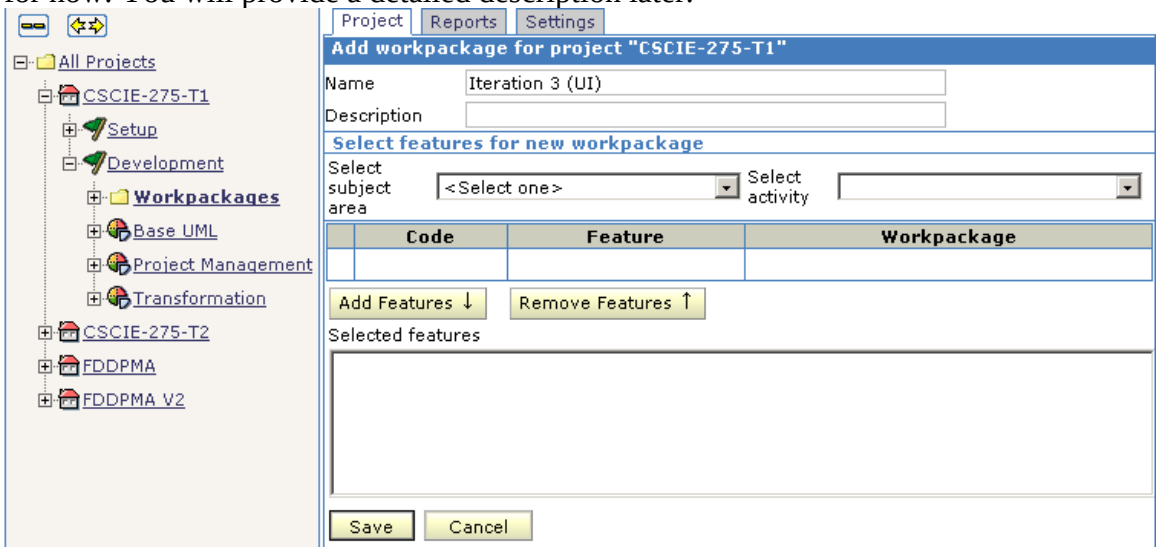
1. Login to FDDPMA application.
2. Expand nodes in the navigation tree for the project, until "Workpackages" node is visible:



3. Select "Workpackages" nodes. FDDPMA will display a list of existing workpackages. Click "New Workpackage" to create a new one.

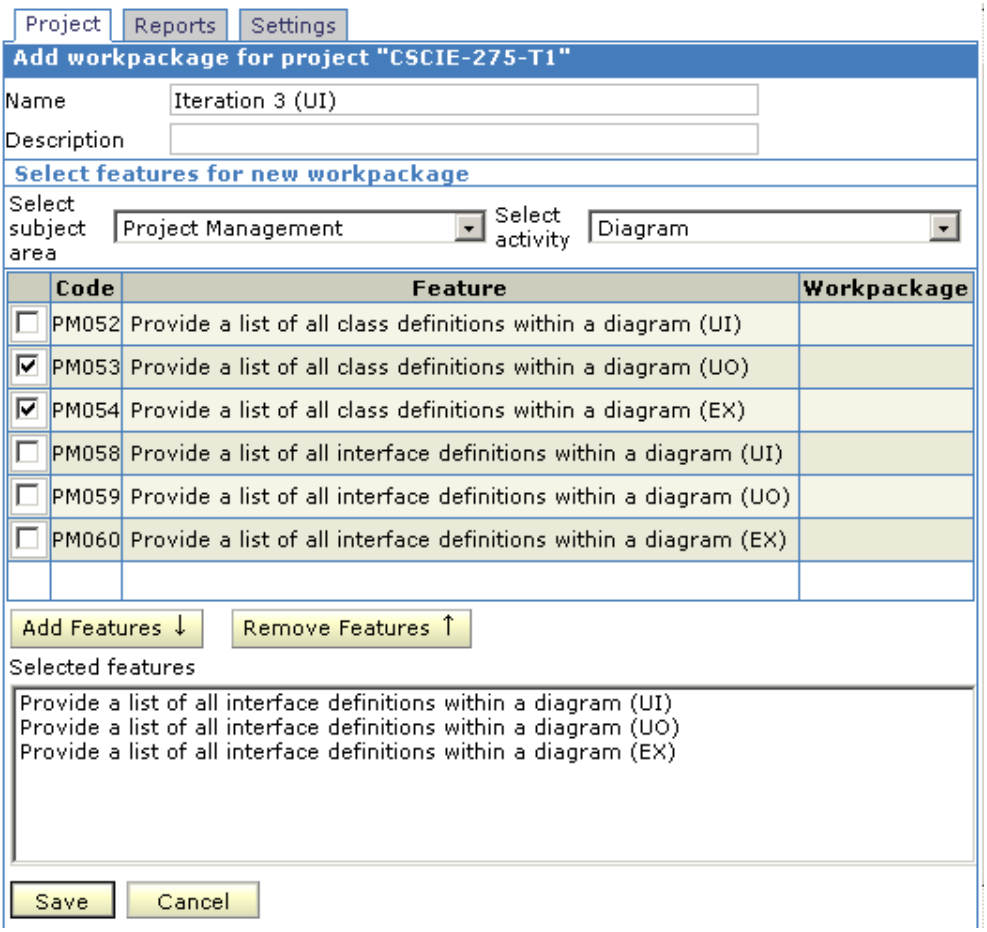


4. Provide a unique name for a new workpackage. Leave the description field empty for now. You will provide a detailed description later.

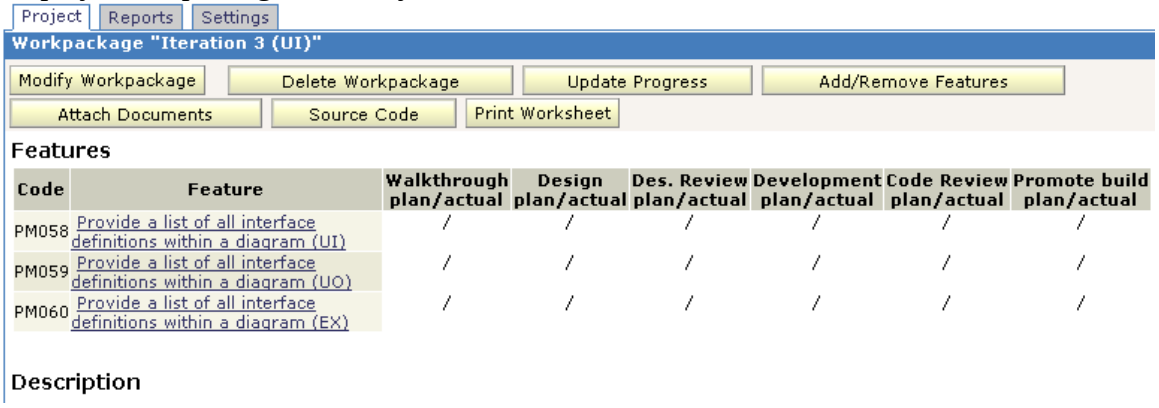


5. Now select features for the new workpackage. Select subject area, then business activity. The application will display a list of all features in the activity. Select those that should be included into the workpackage, then click “Add Features” button. The selected features will be added to the workpackage. Repeat steps

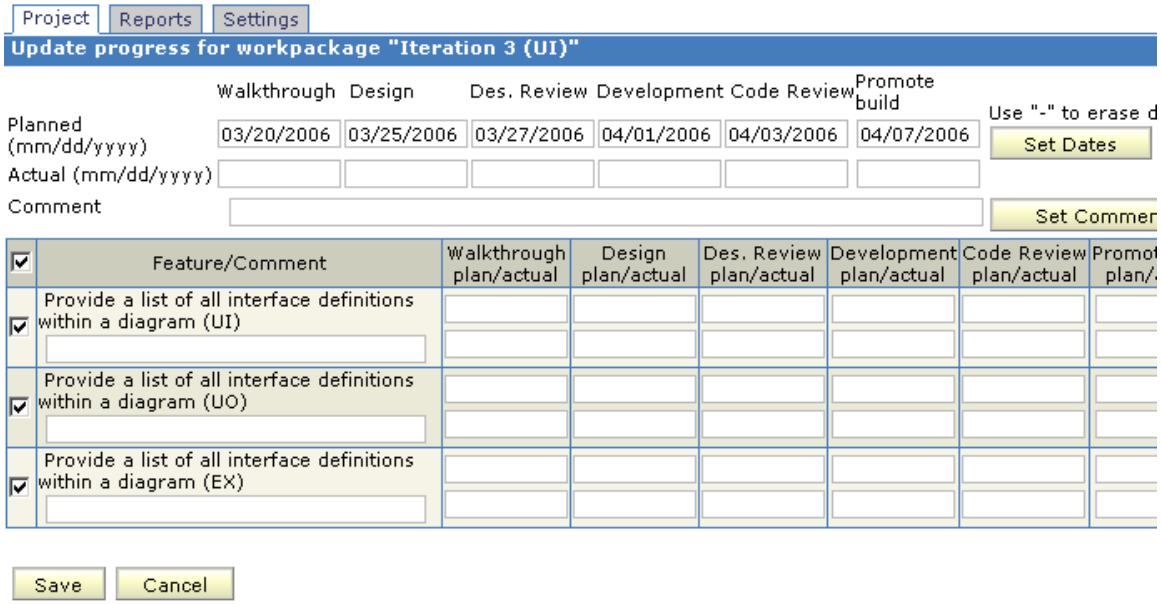
described in this item until all the features are included.



6. Click save button. FDDPMA confirms that workpackage was created and displays workpackage summary view.

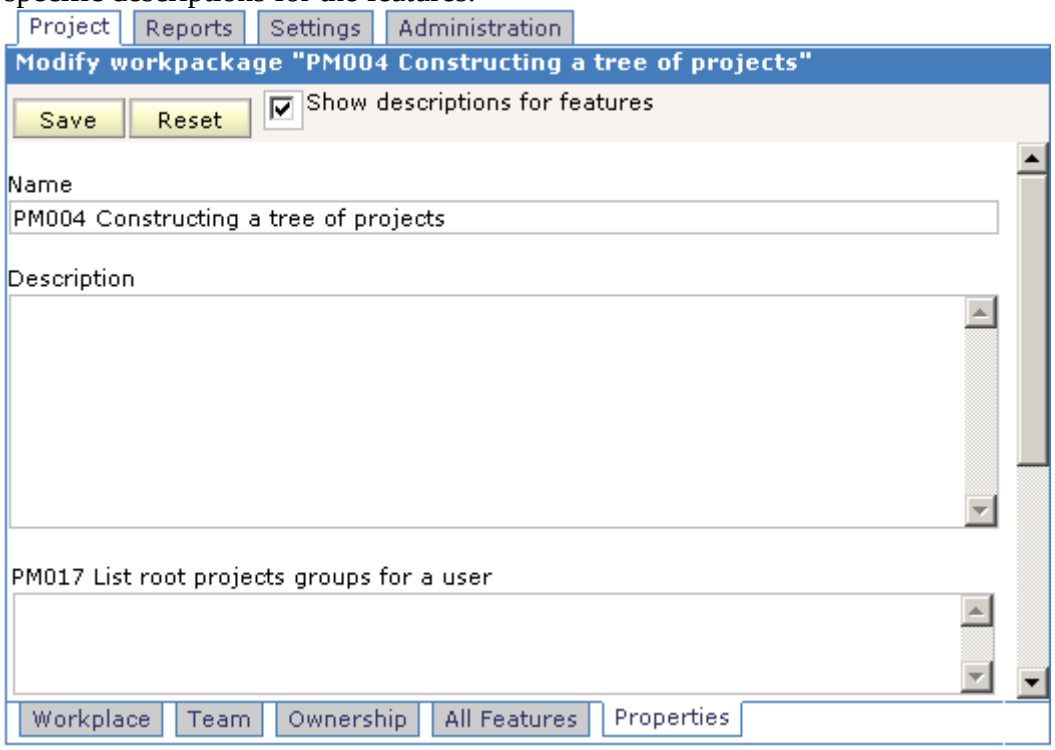


7. Click "Update Progress" button to provide milestone dates for the workpackage. The application displays a screen that allows to set milestones for all the features in the workpackage. Select all the features using the checkboxes on the left. Provide planned dates in the top row of dates. Click “Set Dates” button.



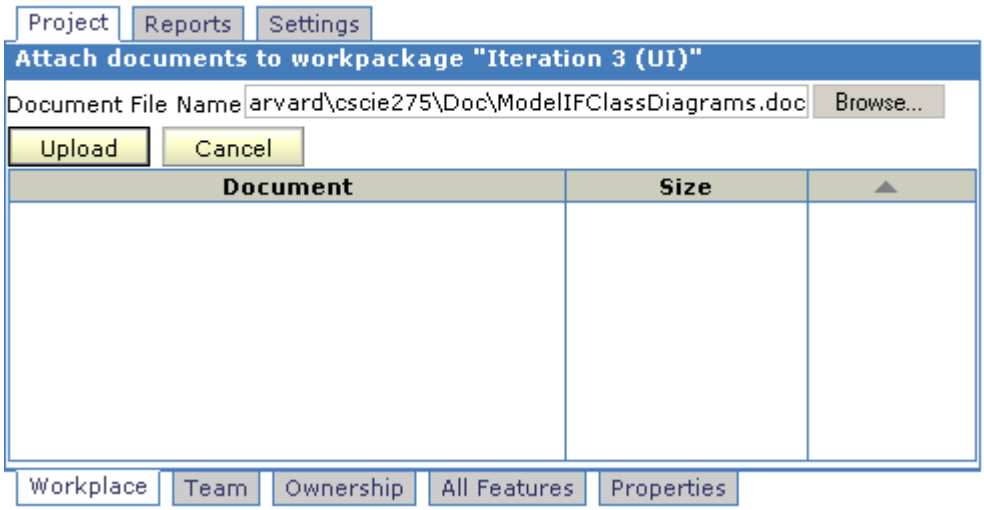
Application will propagate the dates to all the features. Click “Save” button to save the changes and return to workpackage summary view.

8. Click “Modify Workpackage” button to provide the description for a workpackage. The application will switch to the “Properties” sub-tab and display a screen to modify the name and description. Check off “Show descriptions for features” checkbox and the application will allow to edit feature descriptions at the same time.” Type in or paste the common description for workpackage and specific descriptions for the features.



Click “Save”, then click on “Workplace” sub-tab to return to workpackage summary view.

9. Attach the documents that clarify the design (sequence diagrams, mockup screens etc.) Click “Attach documents” button. The application will display a screen to upload the documents



Click “Browse” button to select a document, then click “Upload” button. Repeat this process until all the required documents are attached. Click “Cancel” button to return to workpackage summary view.

10. Click “Print Worksheet” button. The application will generate and display a workpackage worksheet in the form of PDF document. It includes all the information entered for the workpackage.

Workpackage Iteration 2 (UI)

Features

	Walkthrough	Design	Des. Review	Coding	Code Review	Promote build
BU001 Create a class definition (UI)						
Planned	20-Mar-2006	25-Mar-2006	27-Mar-2006	01-Apr-2006	03-Apr-2006	07-Apr-2006
Actual	20-Mar-2006	25-Mar-2006				
1. User opens project menu 2. User clicks 'Add class' 3. Class Properties dialog opens for new class definition 4. The class definition wizard will ask the user for a class name (maybe package). 5. The wizard verifies that the class name is unique among classes and interfaces. 6. Offer the user the choice of adding attributes to the class. 7. Offer the user the choice of adding operations to the class. 7. Offer the user the choice of adding associations to the class.						
BU007 Create an interface definition (UI)						
Planned	20-Mar-2006	25-Mar-2006	27-Mar-2006	01-Apr-2006	03-Apr-2006	07-Apr-2006
Actual	20-Mar-2006	25-Mar-2006				
1. User opens project menu						

Let the developers know that the workpackage design is ready. The developers will login into the application and print the workpackage worksheet.

7. Appendix 2. Tools to create the documentation for design or code review.

7.1 Using java2html converter.

The "java to html" converter is available here: http://www.java2html.de/. The converter performs java-language highlighting and provides line numbers.
There are three ways to use the converter:

- As a plug-in for Eclipse IDE. A user may select a number of classes in the navigation tree, and convert them into a single HTML file.
- As an on-line tool at http://www.java2html.de/applet.html. Remember to check off the "line numbers" checkbox. Copy and paste the source code of your java classes into the source window. The output window will contain HTML code for a single output document. Copy the generated HTML code and paste it into HTML document.
- As a command-line utility. A user may convert a single file, or a directory with sub-directories containing many java files. This approach creates many HTML files, which further need to be merged into a single document.

Once the document is created, open it in MS-Word and highlight the portions that are relevant for the current workpackage.

7.2 Using MS-Word.

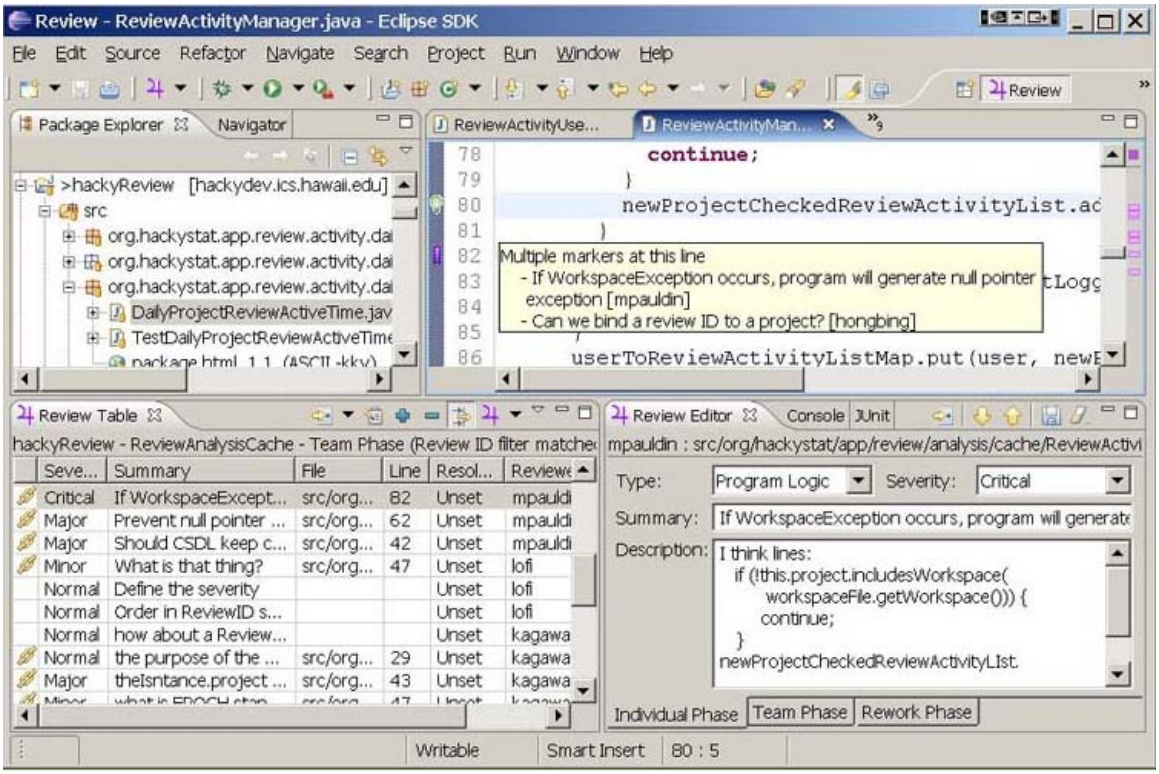
MS-Word can insert line numbers into the document.

1) Create a new blank document.
2) Copy and paste source code of all you classes into the document.
3) Select the whole document.
4) Select "File->Page Setup" menu
5) Select Layout tab.
6) Click "Line Numbers" , check "Add line numbering" checkbox, pick "Continuous" option, click "Ok"
7) Highlight the text that is relevant for the current workpackage.

The resulting document will not preserve java-language highlighting and is slightly harder to read.

7.3 Using Jupiter plug-in for Eclipse

Jupiter is a pug-in for Eclipse was created in the University of Hawaii: http://csdl.ics.hawaii.edu/Tools/Jupiter/. Its purpose is to facilitate design and code reviews.



The prerequisite is that all the team members install the Eclipse IDE and Jupiter. The Jupiter plug-in creates a set of XML files, which may be shared via CVS. Theoretically, the tool may eliminate the need to conduct the live review meetings. However, my personal experience shows that it is counter-productive. While Jupiter may be used to prepare design document for the meeting, the live/phone meetings still work better. The detailed comments are available upon request.