Review Process

In the KIELER project we are conducting regular code reviews of (possibly) all of our sources. To make things easier, we do software assisted reviews in Crucible.

On this Page

- Design Reviews
- Code Reviews
  - After the Review
- Code Tags
  - Design Reviews
  - Code Reviews
  - Review Overview Page
  - Tag Syntax

Design Reviews

Design reviews are scheduled during our various project meetings. For each review, one author and at least two reviewers (though not more than three) are selected. Design reviews take place during special design review meetings. The author's job is to prepare for the meeting by doing the following:

- Choose the classes that are to be reviewed.
- Prepare a meeting page in our Wiki.
- Fill the meeting page with all information necessary for the reviewers to prepare for the code review. That may include class or sequence diagrams. You may also want to take this opportunity to add documentation about the reviewed bits and pieces to the Wiki and link to that documentation from the meeting page.

Now the reviewers look at your meeting page and through the code that is to be reviewed. During the meeting, you will discuss the comments of the reviewers and take notes of all change decisions in the meeting notes. After the review, the author goes through all decisions and gets the changes done, noting so in the notes. The author also adds a tag to the reviewed classes to mark them as design reviewed (see below).

Code Reviews

Code reviews are scheduled during our various project meetings. For each review, one author and at least two reviewers (though not more than three) are selected. The author's job is then to do as follows:

- Choose about four Java classes that are to be reviewed
- Create a Review Task in Jira
- Create a review in Crucible
- Invite the reviewers to the review

Now it is the turn of the reviewers to perform the review:

- Read the source code of the class files in Crucible
- Understand the source code and the class files
- Write a comment if there is something that needs to be improved and mark the comment as defect
- If you want to put more emphasis in your opinion, create a corresponding Jira issue for the defect directly from Crucible
- Write a comment if there is anything that is not understood

What goes into a code review?

If you have never reviewed source code before, you might wonder what to write into the review. Here's a few things you might look out for:

- Is the code understandable as it is? If not, try to find out what is wrong. The problems may be that comments are missing or misleading, that the code has an awkward structure, or that it is even outright wrong.
- Does every class, every field and every method has a Javadoc comment?
- Do the field and variable names make sense? For example, loop variables are often named "i" or "j", while they should actually be called "fooArrayIndex" or something similar.
- Does the code contain logic bugs? For example, dead code that never gets executed or null pointer problems?

Basically, look out for anything that strikes you as odd or problematic.

After the Review

After the review it is up to the author to respond to all comments the reviewers gave.

- The author goes through all comments in Crucible and leaves a note about his decision
- The author may schedule a followup meeting which is only held online to review for example auxiliary classes
- The author adds a tag to the reviewed classes to mark them as reviewed according to the proposed rating (see below)

Code Tags

Design Reviews

When classes are ready for a design review, they should be marked with the following tag:

```
@kieler.design proposed <comment>
```

After the design review has been performed, the same tag can be used to mark the reviewed classes by removing the `proposed` modifier and adapting the comment. The comment should contain the date of the design review (*yyyy-mm-dd*) and the login names of the attendees.

Code Reviews

All classes have initial code review rating *red*. The first code review lifts them to *yellow*, and the second one to *green*. The following tag marks classes that are ready for the first code review:

```
@kieler.rating proposed yellow <comment>
```

After the code review has been performed, the same tag can be used to mark the reviewed classes by removing the `proposed` modifier and adapting the comment. Similarly, the second code review can be marked using the `green` modifier. The comment should contain the date of the code review (*yyyy-mm-dd*), the login names of the attendees, and the ID of the code review in Crucible (such as `KI-15`).

Review Overview Page

The design and code review tags are processed automatically using a custom doclet during Bamboo builds. The result is displayed in a generated HTML page:

http://rtsys.informatik.uni-kiel.de/~kieler/doc/rating/

Tag Syntax

The complete syntax of the available tags is as follows:

```
@kieler.design [proposed] yyyy-mm-dd comment
   Marks a class as design-reviewed or to be design-reviewed. For proposed classes, the
   comment should mention the login name of the person proposing the class to be reviewed.
   For reviewed classes, the comment should include the names of the reviewers.

@kieler.rating [proposed] <yellow|green> yyyy-mm-dd comment
   Marks a class as code-reviewed or to be code-reviewed. For proposed classes, the
   comment should mention the login name of the person proposing the class to be reviewed.
   For reviewed classes, the comment should include the names of the reviewers as well as
   the ID of the review in Crucible (for instance "KI-15").

@kieler.ignore comment
   Marks a class as to be ignored in the code rating statistics. The comment should provide
   an explanation of why the class is ignored.
```

To be compatible with past tags, the doclet is actually quite flexible in terms of what it still accepts. However, it is good practice to follow these recommendations.