Project Management

WebKit bug reporting guidelines

Ricardo García Fernández

January 31, 2013

1 Questions

Analyze the following bug reporting guidelines from the WebKit project:

`http://www.webkit.org/quality/bugwriting.html`

Pick up another popular FLOSS project/community with similar guidelines and find similarities and differences. After this, can you briefly summarize which are (in your own opinion) the most relevant guidelines for submitting appropriate bug reports to a FLOSS project. What is the role of automated bug reporting systems in this process?

2 WebKit guidelines

1. **Version**: Select WebKit version: 537.11
2. **Component**: Select cause to specific component: CSS, HTML Editing
3. **Platform and OS**: Complete OS field (Ubuntu 12.04)
4. **Priority**: This field will be set up by Q&A member, but you have a guide to choose it at `http://www.webkit.org/quality/bugpriorities.html`
5. **Severity**: the main case is *normal*, but you could choose between *trivial* or *enhancement*. Q&A member will check this field.
6. **URL**: Fill the field with URL that replicates the bug. There is also a guide for this section, Test Case Reduction[1].
7. **Summary**: *One should be able to tell exactly what a bug is about just by reading the summary.* It's the most important part and has to be well redacted and clear for everyone. Replicate the error, URL, is a REGRESSION bug ? if it's a REGRESSION, which bug points to?
8. **Description**: Detailed explanation of the problem. Include only one bug, WebKit version (release, nightly build, etc...), URLs, log messages, backtrace, screenshot for visual bugs. **Be specific**.
9. **Keywords**: Select specific Keywork for tagging the bug. There is a guide for standard Keyword in WebKit Bugzilla[2].
10. **Depends on**: If the fix for this bug depends on another bug fix, attach bug's number.
11. **Blocks**: If this bug blocks another bug, select related bug.

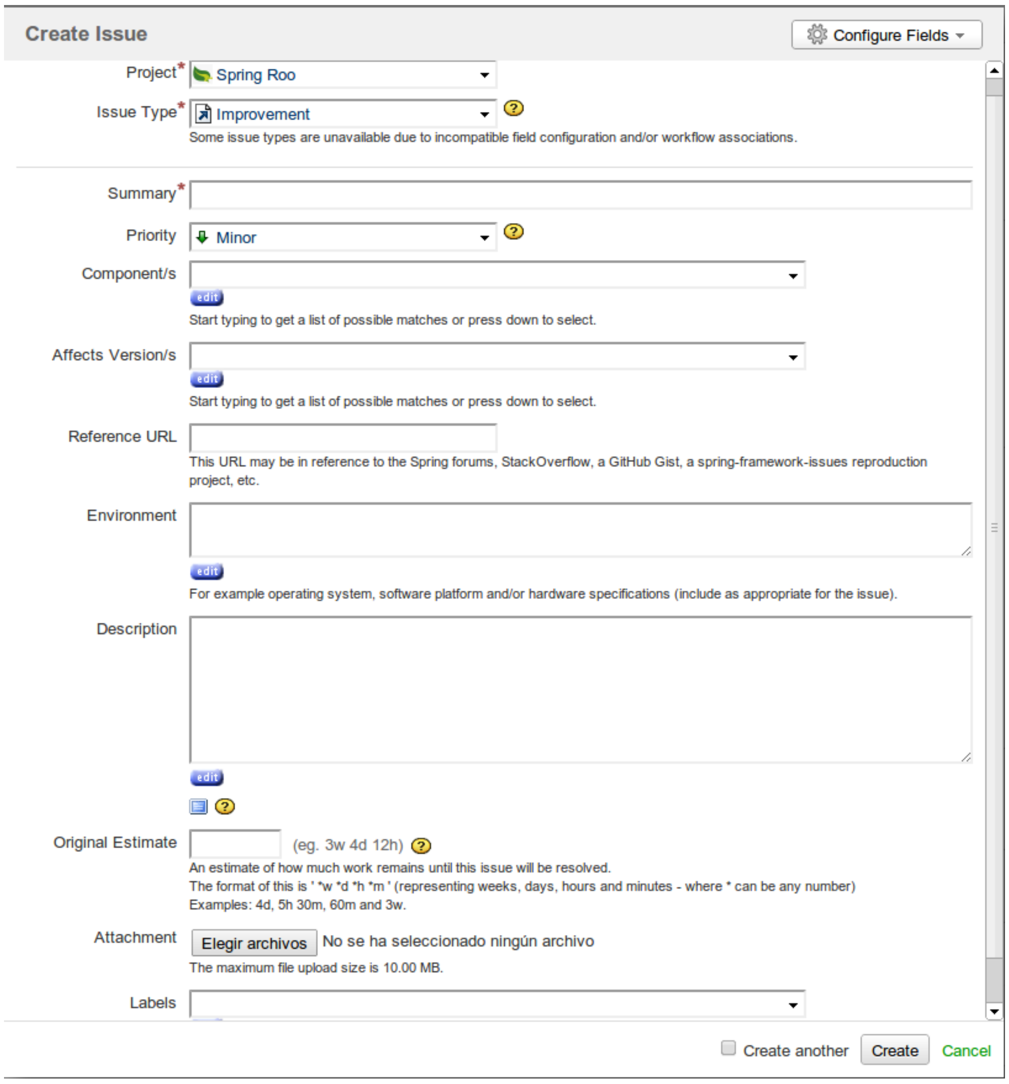
[1] `http://www.webkit.org/quality/reduction.html`
[2] `https://bugs.webkit.org/describekeywords.cgi`

3 Spring-ROO bug tracker

Spring community guide *Get Invloved*[3], aims us to report issues related to each Spring module in Issue Tracking System (Included Bugs).

So I choose Spring-ROO Issues/Bug tracker[4] to compare with WebKit BTS -*Bug Tracking System*-. Spring-ROO uses JIRA[5] instead of Bugzilla[6] that is used by WebKit.

Go to Spring-ROO JIRA web, and create new bug. JIRA shows this popup:



Project*	Spring Roo
Issue Type*	Improvement
	Some issue types are unavailable due to incompatible field configuration and/or workflow associations.
Summary*	
Priority	Minor
Component/s	
	Start typing to get a list of possible matches or press down to select.
Affects Version/s	
	Start typing to get a list of possible matches or press down to select.
Reference URL	
	This URL may be in reference to the Spring forums, StackOverflow, a GitHub Gist, a spring-framework-issues reproduction project, etc.
Environment	
	For example operating system, software platform and/or hardware specifications (include as appropriate for the issue).
Description	
Original Estimate	(eg. 3w 4d 12h)
	An estimate of how much work remains until this issue will be resolved. The format of this is ' *w *d *h *m ' (representing weeks, days, hours and minutes - where * can be any number) Examples: 4d, 5h 30m, 60m and 3w.
Attachment	No se ha seleccionado ningún archivo
	The maximum file upload size is 10.00 MB.
Labels	

[3]http://www.springsource.org/get-involved
[4]https://jira.springsource.org/browse/ROO
[5]http://www.atlassian.com/en/software/jira/overview
[6]http://www.bugzilla.org/

After open it, we can go to Bug Guideline at `https://jira.springsource.org/secure/ShowConstantsHelp.jspa?decorator=popup#IssueTypes` and read Spring bug reporting statements:

1. *the issue type*: show issues type list to choose.
2. *a summary*: Brief summary of the bug.
3. *a description of the issue*: More complete description about the bug you are reporting. **Be clear and specific**. How to replicate the Issue.
4. *the project which the issue belongs to*: In this case Spring-ROO.
5. *components within a project which are associated with this issue*: Which component/s in project is/are affected by this bug.
6. *versions of the project which are affected by this issue*: Spring- ROO Version to replicate the bug.
7. *versions of the project which will resolve the issue*: This is for Spring-ROO team.
8. *the environment in which it occurs*: Environment, OS, platform, related libraries,...
9. *URL*: Forum url for comments, reproductions, project, etc.
10. *Original Estimate*: Estimate of how much work until its resolved.
11. *Attachments*: Files to help to resolve the issue.
12. *Labels*: Tags for the issue. This helps to find related issues and help.
13. *Issues links*: Related issues that could depend or block this work.
14. *a priority for being fixed*: You can set the priority, but will be tested by developers to give a more realistic value.
15. *an assigned developer to work on the task*: This is for Developers Team, after read the Bug/Issue.
16. *a reporter* - the user who entered the issue into the system: Filled automatically with your Spring-ROO JIRA user data.
17. *the current status of the issue*: Status changed by Developers.
18. *a full history log of all field changes that have occurred*: Log history for each issue.
19. *a comment trail added by users*: Log history comments.
20. *if the issue is resolved - the resolution*: Final result explained by developer.

4 Similarities and Differences

We can see the similarities outweigh the differences.

Bug-Trackers Both make great stress upon definition versions of their projects. The environment where the bug has been found and as you can reply, accompanied by a clear and specific description of how we got the bug. The section which specifies the component where the bug is found and the tag that you assign. Also if there is a related bug that may affect the new bug.

One similarity to note is the reference for regarding the error logs. The logs are written by the application, so they are clearer and there is not a different interpretation by each person. It is a way to avoid the 'human error' and certify the error.

Above all we can get the idea that the most important thing is to make a clear description of the bug is and how to replicate it.

More technical data developers are responsible, such as the priority, the resolution time and severity.

Furthermore, the few differences that can be found in the different *BTS* are related to the collection of information. By *Spring-ROO*, emphasis is on linking the Bug with other sources of data, such as for forums, websites to unify the information in one source and not waste any source.

5 Most relevant

In my opinion, most relevant guidelines in a *BTS* are in common with *WebKit* and *Spring-ROO*.

- Write a **clear and specific summary and description** about the bug.
- Information about the environment and version affected.
- Describe how to replicate the bug, step by step and which tools you used (version, etc. . .).
- Log information, traces.

6 Automated Bug Reporting Systems

The BRS attempt, through a basic standard of information, create a bug, without disturbing the user. That is, bugs are sometimes lost by the mere fact that a person has to attach the error that happened in the corresponding BTS. This person may not know what a BTS, not feel like registering at another site, the error does not stop you keep doing your job, you had a bad experience in other BTS, invest time in it, does not know how to report bugs, anything related to the 'human behaviour' of each of us.

Therefore, the automatic transmission bugs, try to gather information when there is a bug in the project automatically, environment, version, related projects, record logs, user, through a friendly interface that the person has the option sending the error, ie, know what you are doing without having to invest almost no time. This will collect more bugs and you can get more information and different statistics of bugs replicated in different environments.

And if the person wants to track closer you are given the option to get involved in solving the bug, but that has to look for information. For thus user collaboration more accurate and show their level of involvement. So bugs will be more accurate and better drafted.