University of Toronto
Department of Computer Science

CSC444F – Software Engineering I

Instructor: Matt Medland
October 13, 2015

Assignment 2: Implementing Change Requests

Due Date: Tuesday, November 3, 2015 at the start of lecture

This assignment counts for 7.5% of the final grade.

Analyze the issues in the Mozilla bugzilla project at **bugzilla.mozilla.org**. There are instructions on the **course web page tutorial section** on how to pick a good first bug to work on. Limit your search to the **Core** and/or **Firefox** products in bugzilla. You do not have to limit yourself only to issues labeled as "good first bug," but it's a good place to start looking at issues.

Using an appropriate software development process, select at least two issues (bugs and/or small features), complete the implementation of them in your repos on the EECG lab. Provide test cases suitable to demonstrate that the changes have been correctly implemented. Note that it is possible that a real life mozilla committer may be simultaneously working on the same issue in the real mercurial project. If this happens, please resist the urge to look at the code they commit and focus only on your team's implementation.

This assignment requires you to use your judgment about what software process to use, and which bugzilla issues to select for implementation. There is no "correct" choice – you will be given credit for selecting changes that can be implemented correctly in the time available, and which are most likely to satisfy the users. Issues that involve manipulating code are strongly preferred. Note that users are more likely to value simple fixes that work reliably over more ambitious features that are incomplete.

This assignment is to be carried out in your teams. Each team will submit one report.

Doing the Assignment:

This assignment has 8 required steps, and one optional step. They are:

1. Examine the **Firefox** and **Core** product issue lists in bugzilla. It is recommended that you get started by looking at issues labeled "good first bug." From the issue list, select a handful of interesting bugs or features to examine further. This is your issue shortlist – list all the issues selected in your report.

2. Draw one or more use case diagrams illustrating all the use cases relevant to the items on your shortlist. Use an appropriate UML drawing tool to draw these diagrams. You can use the same tools you used for assignment #1, or choose different tools.

3. Select an appropriate software process model to guide you through the remainder of this assignment. Use any of the software process models presented in class (e.g. SCRUM, XP, or one of the more "traditional" methods), or any other software process that you are familiar with. You will need to consider how to adapt the process to your particular needs.

4. Select at least two items from your shortlist to implement and test. Use your chosen process model to guide you in the selection process. This may involve documenting the use cases in more detail. Estimate the effort required to implement each change, and identify any anticipated risks. You will also need to determine which team members are allocated to work on which of the tasks.

5. Implement your selected issues. Ensure your mercurial server is running, then individually, clone your team's repository to a location where you will be working on the code. Your EECG home directories have disk quota limits which may prevent you from cloning it there. **When you commit changes (locally) use the --user argument to the hg command and specify your EECG account name**. This is important so that your TA will know which team member made a given change when examining the mercurial log. Be sure push your changes back to your group's repo on EECG when you are finished (actually, you should push/pull & update frequently while you work). Your TA may clone your repository and run your changes, or look at mercurial logs, while marking this assignment.

6. Write test cases to demonstrate that the changes have been implemented correctly. Design these as "user acceptance tests" (UATs) – i.e. a description of the steps a user needs to carry out to check that the items were implemented correctly as requested/reported.

7. Write a report that describes the steps you went through to select and implement the issues you worked on in this assignment. Document your development process, and comment on how well the process worked for you.

8. Document your teamwork by completing the group evaluation form, which is linked on the course webpage. Submit your printed review form in person to your TA at the beginning of the standup meeting following the due date.

9. [optional] before making changes you can pull and merge the master Firefox mercurial repo with your team's repo on EECG. This will bring your group's repo up to date w.r.t. the master repo and the bugzilla issue list. This is not strictly required for this assignment. As of the time of writing this handout, there have been 3,817 changes committed since creating the group repos.

What to Submit:

Submit your report by email to the instructor before class on the due date. The report should not exceed fifteen (15) pages (not counting cover pages, appendices, and group evaluation forms). It should include the following items:

1. A brief description of the software development process you used, including the reasons you selected this process, and any steps you took to adapt the process to your needs.

2. A use case diagram showing use cases relevant to the list of bugs/features on your shortlist, plus any other documentation you produced to describe use cases and/or change requests in more detail. Make sure to identify each issue with its issue ID in bugzilla.

3. A brief description of your implementation plan. Describe the rationale you used for selecting the items you chose to work on, and any risks you identified when you developed the plan. Write a brief technical commentary on how the changes affect the design and/or the code of Firefox. List all relevant Firefox source code files that were added, modified, or removed as part of your implementation work.

4. A set of user acceptance tests (UATs), described in a form that would allow any user to execute the tests using Firefox, and determine that the software works correctly. You can assume that the user has a basic working knowledge of the Firefox browser, and web terminology in general.

5. A review of lessons learned in carrying out this assignment, including commentary on how the chosen process helped or hindered you, and any problems you encountered, technical or otherwise.

Be sure to include a cover page indicating the name of your team, the names and student numbers of all team members, title of work, course, and date. Assignments will be judged on the basis of appearance, the grammatical correctness and quality of writing, and the visual appearance and readability of models and diagrams, as well as their content. Please make sure that the text of your report is well structured, using paragraphs, full sentences, and other features of a well-written presentation. Use itemized lists of points where appropriate. Text font size should be either 10 or 12 point.

Marking Scheme:

Marking of your assignment will be handled by your TAs. If you have questions about a marked assignment, you should first ask your TA before/after a tutorial or by email. If you don't get satisfactory answers, you should talk to your instructor.

Marks for this assignment will depend on the following factors:

- Description of your process (15%): Did you identify and evaluate a suitable development process? Does your choice take into account the circumstances of this project, including project size, team experience and schedule? Did you clearly describe how you adapted the process to your specific team's needs?

Did you understand how to apply the process, and did you follow it? Did you describe how well the process worked, and identify lessons learned?

- Description of use case analysis (15%): Did you draw up a shortlist of candidate issues? Did you identify an appropriate set of use cases for the items on your shortlist? Did you draw use case diagrams? Are your use cases written from the user's perspective? (sometimes the user, or actor, is not a human). Did you provide use case diagrams as well as detailed descriptions?

- Your implementation plan (20%): Did you select a manageable subset (but at least two) of the issues on your shortlist for implementation? Did your plan take into account user's likely priorities, as well as the time and effort available? Did you clearly state the rational you used for this selection? Did you identify the major risks associated with your plan? Did you follow the plan, making any adjustments to the plan as needed? Did you describe what the changes were, and how they affected the code? Did you track and document each team member's time spent on implementation?

- Working application and test cases (20%): Can the new version of your software be cloned from your group's mercurial repository hosted on EECG? Does it build and run? Are your test cases clearly described? Can a user execute the test cases without any of your team members' help? Do the changes work the way they should?

- Presentation (15%): The style of your presentation, including language, grammar, clarity of the presentation, layout and legibility of the diagrams, etc.

- Standup meeting (15%): Did you present your work well during the standup meeting? Did you work effectively as a team? Did you demonstrate active involvement of all team members in the project? If your teams had problems, did you address them effectively and in a timely manner?