University of Toronto
Department of Computer Science

CSC444F – Software Engineering I

Instructor: Matt Medland
November 3, 2015

Assignment 3: Requirements, Plan, Implement & Test

Due Date: Tuesday, December 1, 2015 at the start of lecture

This assignment counts for 10% of the final grade.

Select a few issues from the Mozilla Bugzilla database for your group to implement. Alternatively, if your group has a really good idea for a substantial Mozilla Firefox add-on/extension, you can pitch your idea to the instructor for approval. You should break down your chosen feature(s) into smaller, implementable, sub-features and apply a sizing to them. Feel free to use whatever issue tracking software you like to manage your feature(s).

The analysis of your features will include user requirements, a domain model to map out domain entities (if applicable), a set of use cases describing how users will interact with your features, and optional UML sequence diagrams to help describe the use cases. Make a development plan (a.k.a. Agile Horizon Plan) for your chosen features, as described in section 3.4 of the Penny book. You will also provide a brief risk analysis associated with the plan.

As your team is implementing the chosen features, track your time and progress on the plan. Try to keep capacity and requirement in balance at all times and make note of any significant changes (ex. feature work happening faster than expected (unlikely), or unanticipated extra effort (very likely)). Produce a burn-down chart that tracks all capacity and requirement changes throughout the plan.

You are required to write test cases and demonstrate the code coverage of those tests. Ideally, the tests should run the same way as other Mozilla automated test cases do, leveraging their preexisting framework.

This assignment is to be carried out in your teams. Each team will submit one report and present their work on the last day of class.

Doing the Assignment:

This assignment has ten (10) steps. They are:

1. Bring your group's mercurial repository up to date with the master Mozilla-central repo located at:
   https://hg.mozilla.org/mozilla-central/

This should be done as early as possible, and certainly before any new code is committed for your a3 features. Document what you did to bring your repo up to date and if there were any problems (ex. UNIX permissions issues, auto-merge failures, etc.) and how you resolved them.

2. Split up your feature(s) into smaller, implementable chunks. Play planning-poker with your teammates to come up with initial effort estimates. When estimating, use real time units rather than something abstract like points, or a low, med, high scale. Make an initial release (Agile Horizon) plan where the end date is the due date for the assignment. Include a brief qualitative risk assessment with the associated plan that outlines possible undesirable outcomes, likelihood of their occurrence, and potential mitigating actions.

3. Complete a requirements analysis for your selected feature(s). Your requirements analysis must identify why users want the requested feature(s) or add-on/extension. It should identify relevant assumptions about the problem domain. At the end of the analysis, you should have a detailed list of specific functions and quality requirements, along with a rationale for how the functions will allow the users to meet their goals.

4. Generate detailed use cases (3 max.) for the selected feature(s). Create a use case diagram to give an overview of the set of use cases, and create individual descriptions for each use case, including pre and post-conditions, exceptional behavior, and alternative paths for each use case. Your use case descriptions should be written in a style that typical users can understand, so that, for example, they could form the basis for a user manual. Generate sequence diagrams, as necessary, to clarify and provide more detail to the use cases.

5. Implement your chosen feature(s) or add-on/extension. Decide on which development process you will use and how you will track your implementation efforts. Assign the (broken-down sub-)features to individuals and re-estimate as needed. Try to use existing (open-source) libraries to help you where appropriate. If you make a bad choice of library to use it will show poor judgment and hurt your mark.

6. Track your plan as you develop. Your estimates from step 2 will make up the initial requirement for the plan (F) and developer time available will make up your team's capacity (N×T). Your release date is the due date of this assignment. Along with the plan, you will maintain a burn-down graph showing the historical progress of the plan. Whenever you update estimates or log work completed you should update the plan and the burn-down graph. Track your team's velocity on a weekly basis.

7. Create unit (white-box) tests, and automate running the tests, for your feature(s). Depending on your development strategy, you may write your test cases before your features (TDD), as you go (other agile methods), or after your features are complete (most traditional methods). Discuss the level of code coverage of your test suite in your report.

8. Write a report documenting your re-synch with the main Mozilla-central repo, feature break-down and sizing exercise, development process, release plan, burn-down graph, and unit tests. Include a "post-mortem" section describing

what went well, what did not go well, and anything unexpected that occurred. For example, were there any feature estimates that were off by a significant amount? Did you have to adjust the plan by dropping any feature items, etc?

9. Prepare a 5-minute feature demo that will be presented during the last class. One or two slides and a brief live demo will be sufficient.

10. 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.

What to Submit:

Submit your report by email to the instructor before class on the due date. The report should not exceed twenty (20) pages (not counting cover pages, appendices, and group evaluation forms). It should include the following items:

1. A brief description of the feature(s)/extension your group chose to implement. The rationale behind your choice.

2. Discussion of the re-synch process of your group's repo with Mozilla-central as outlined in item 1 in the "Doing the Assignment" section above.

3. Requirements analysis for the selected feature/extension. Your analysis should clearly distinguish between user requirements (e.g. goals) and the specific functions to be implemented, and indicate any assumptions needed to ensure that these functions will indeed satisfy the requirements. You should also include your quality requirements here.

4. A complete set of use cases for the selected feature/extension. Your use cases should show alternative paths and exceptional behaviors, and document pre and post-conditions for each use case. Draw a high-level use case diagram for the feature/extension. Write detailed descriptions for each use case, use further diagrams only to add clarity (ex. sequence diagrams). List domain entities (if applicable) as well.

5. A discussion of your feature breakdown and estimation activities playing "planning-poker." A detailed release plan, containing the features & estimates, as it looked at the beginning of the development effort, and the resulting plan at the end of the effort. Include the final burn-down graph, showing important events on the graph like capacity constraint violations, major feature resizing (up or down), and rebalancing events like cutting, or adding, features. A discussion of your chosen development methodology and its (in)effectiveness in helping carry out the plan.

6. A brief risk analysis associated with the initial development plan, including speculation on mitigating actions that could be taken.

7. A discussion of your automated unit tests, and their associated code coverage.

8. A general "post-mortem" discussion of what went well, what unexpected things came up during the development effort, and any adjustments you had

to make to the plan as a result. Also, what you would have done differently if you had it to do over again.

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. Reports 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:

Your TAs will handle marking of the assignments. 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 feature(s) or add-on/extension (10%):** Did you clearly describe appropriate criteria used for making your decision? Did you investigate existing resources (e.g. libraries, open-source code, etc.) that you could use to help implement the features? Did you use appropriate tools such as estimation and risk analysis to help make the decision?

- **Re-synching your project with Mozilla-central (5%):** Did you list the main steps you took? Did any issues come up that caused you trouble? (ex. merge problems) Did you describe them well and how you handled them?

- **Description of your Requirements Analysis (15%)**: Did you analyze the requested feature(s)/extensions from the user's perspective? Did you clearly distinguish between the user's goals and the functions that the software will provide? Is your list of required functions complete and appropriate? Did you set out a clear argument for why the specified functions will meet the user's goals? Did you identify any domain assumptions used in these analyses?

- **Description of your use cases (15%):** Did you identify an appropriate set of use cases for the requested feature(s)/extensions? Did you draw a use case diagram? Are your use cases written from the users' perspective? Did you clearly document the normal and alternative paths for each use case & identify exceptions? Did you identify pre and post-conditions for each use case? Did you identify relationships between use cases (e.g. "uses", "extends") where appropriate?

- **Development plan (25%):** Did you describe the process of breaking your feature down into implementable chunks of work? Did you construct an initial version of the development plan that was balanced (i.e. $F \leq N \times T$)? Did you use an appropriate tool to track your features? Did you adjust your estimates as work progressed and as you learned new information? Did you re-balance your release plan (if necessary) and justify your changes? Did you keep a burn-down graph updated as you made changes to the release plan? How did

the burn-down graph help you make decisions regarding how to adjust the plan?

- **Automated Testing (10%):** Did you write an appropriate set of unit tests? Did you automate the execution of the tests effectively? Did you provide an analysis of the code coverage of your unit tests on your code?

- **Professional quality report (5%):** The style of your report, including language, grammar, clarity of the presentation, layout and legibility of the diagrams, charts, graphs etc. must be of high quality.

- **Demo & class presentation (10%):** Was your demo presentation clear and to the point? Was it obvious to the audience what was the new functionality? Was it clear why this new functionality is valuable to users?

- **Standup meeting (5%):** 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?