Project Report: Editor Scalability

CS 427 Software Engineering, Fall 2012

Team Bagyamn

1. **Brief description of project**

The current Photran editor behaves badly for large files. For very large files with several thousand lines of code, the editor is not able to keep up with typing. It freezes for several seconds before the user can see the characters that he typed. It also takes a long time to load a large file into the editor.

The main goal of our project is to improve the performance of the Photran editor for large files. In order to do this, we need tests to measure the performance of the original editor, and tests to show the improvement in performance after implementing our solutions.

This project is primarily about reverse engineering the system and comprehensive testing. Our implemented solutions are as follows:

1. Threaded lexer and parser
2. Delayed lexer and parser
3. Delayed reconciler
4. Automated testing using SWTBot
5. **Architecture & Design**

This section is divided into 2 parts:

1. **UML diagrams:**
2. **Testing framework:**

In order to automate the UI testing for our project, we used the SWTBot framework. SWTBot is an open-source Java based functional testing tool for testing SWT and Eclipse based applications. More information about SWTBot can be found at the following links:

<<add class names, some technical details>>

<http://wiki.eclipse.org/SWTBot/UsersGuide>

<http://www.vogella.com/articles/SWTBot/article.html>

1. **Future plans**

We plan to upload our code to github and continue working on it through the winter vacation. There were some tasks on our agenda that could not be completed due to time constraints. We plan to work on these and submit our code to Jeff Overbey for inclusion into the plugin.

We will be specifically focusing on the following:

1. Include a GUI option to implement the scalability option.
2. Test more thoroughly, because there is always a requirement of more testing in any system.
3. We used SWT bot for testing and noticed some inconsistencies in its behavior. It gives the testing time readings in a range of +/- 20 for the same file. We plan to get to the root of these deviations observed in its behavior.
4. We also want to explore alternative testing strategies.
5. One of the concerns from milestone 2 - Spell-checking is slow when there are a lot of constant strings – is this solved or can it be included in future plan??
6. Second concern from milestone 2 - Why isn't the AST or VPG tasks running? Is this solved??

* **Personal reflection:**
* Lionel:
* Sam: Trying to identify issues with editor scalability was a task, which at first, didn’t seem like much of a project. This was the complete opposite. Getting an understanding of what the project demanded from us took a long time. Our initial solutions were shut down due to the lack of understanding. After revising our plan and solution, a more substantial solution was proposed and implemented. A prominent issue with this project is the lack of guidance with the framework of our solutions. I feel a little more explanation as to what and how we should go about solving the issues, especially what not to consider solutions. For example, the possible solution listed in the project description is a solution that was turned down by the TA. Finding a good and usable GUI testing framework is another issue. Trying to look for a GUI tester, and then implement it with our project was another difficult but surmountable task. There is a lack of GUI testers, and some of them are actually proprietary. This increased the time it took to create decent automated tests because of the steep learning curve and lack of documentation for the few open source testers. Another difficult task was reverse engineering. This was especially hard because the documentation on the editor and its various components is severely lacking. Took copious amounts of time to identify the various aspects of code we needed and the various features it affected. Overall the project was hard, but beneficial in acclimating me to reverse engineering.
* Srikanth:
* Nisha:
* Sugandha: This project taught me how to ramp up on a new area quickly and understand the basic workings without reading every line of code. It helped me develop the skill to read somebody else’s code intuitively. The continuous emphasis on testing helped us ensure a good quality final product. We often underestimate the importance of testing as a coder, but this project helped to make continuous testing a habit.
* Rucha: Working on this project gave me a flavor of working on a large scale project. I realized that software engineering is not always about writing new code; understanding code written by someone else is an equally important skill. The project helped me improve my reverse engineering skills. I learned how to quickly navigate to different parts of the project, in order to understand the flow of logic between the different components of the system.

1. **Testing**
2. **Appendix**
   * 1. **Installation instructions:**
     2. **Steps to run the program:**
     3. **Steps to run the tests:**