

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. Move the parser to the reconciler thread
2. Automated testing using SWTBot

## 2. Architecture & Design

We made changes in the following packages:

1. Project name: **org.eclipse.photran.cdtinterface.vpg**

Package name: **org.eclipse.photran.internal.core.model**

File modified: **FortranModelBuilder.java**

The Model Builder contains the code to parse the source code and build the AST and VPG. It is a time consuming task that was previously run on the main thread. We improved the responsiveness of the editor by preventing the main thread from running this function, and made the Reconciler thread run it instead.

2. Project name: **org.eclipse.photran.ui.vpg**

Package name: **org.eclipse.photran.internal.ui.editor\_vpg**

File modified: **FortranVPGReconcilingStrategy.java**

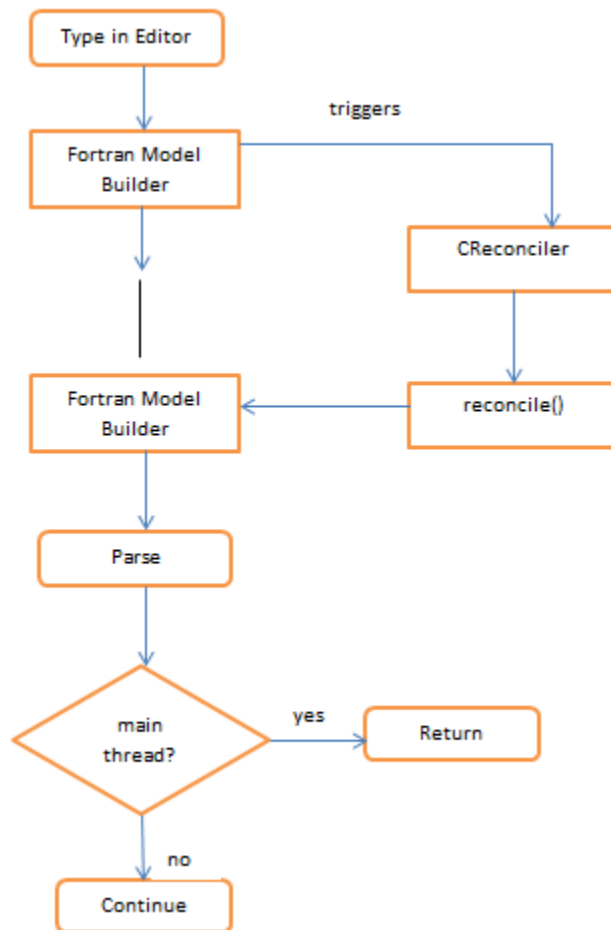
This is where the reconciler thread will call the reconcile function. It will result in the Outline View to be updated. In our solution, we call the working copy translation unit to reconcile and compute the AST before updating the Outline View.

3. Project name: **org.eclipse.photran.ui**

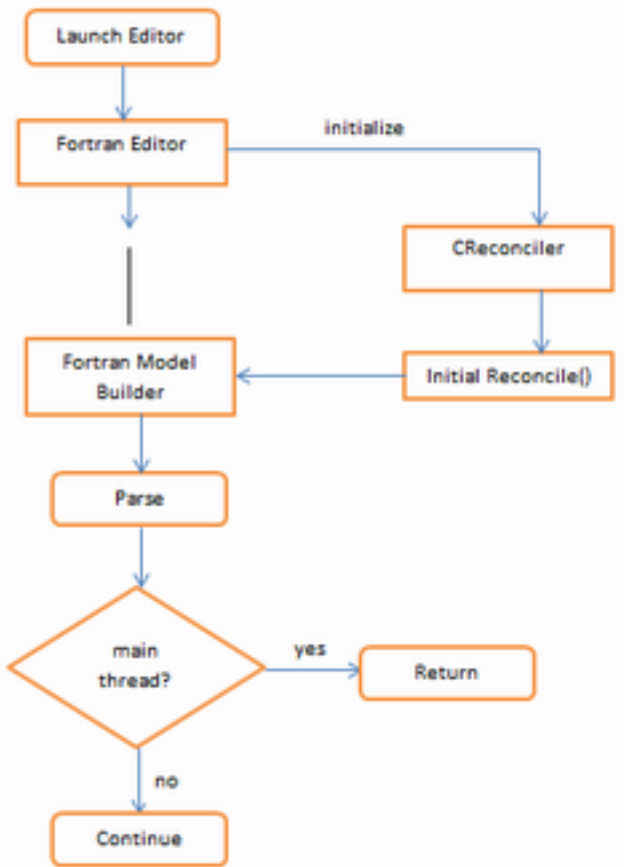
Automated test cases were added to this package, under the folder **swttests**

## Flow Diagrams:

When launching a text editor:



When typing:



### 3. 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. Try to load a large file seamlessly.
3. Test more thoroughly, because there is always a requirement of more testing in any system.
4. 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.
5. We also want to explore alternative testing strategies.

6. We found out that spell-checking is slow when there are a lot of constant strings. We want to explore this option to see if it gives any improvement over our solution.
7. We will try to find out why the AST and VPG tasks are not running.

### **Personal reflection:**

#### **Lionel:**

The Editor Scalability project attempts to address the issue with the Photran editor where the editor does not scale well with the size of the file. The process of reverse engineering the Photran Editor is very challenging. It was difficult to know where to start as the whole project is huge and consists of many functions. Some components run on different threads, and the Photran Development Guide didn't really help us too much in our project, as it didn't have much documentation on the classes and components that pertain to our development. The XP process is particularly useful. Working in pair programming fosters us to think together and get everyone involved in a project as large as the Photran Editor. I find the testing process critical to our development process as it helps redefine our solution.

#### **Srikanth:**

This project helped me in gaining insights about how to approach the problems related to scalability in heavy weight applications like eclipse. It also taught me the importance of how debugging can be helpful in reverse engineering. The existing documentation is not really up-to-date and didn't really help us in reverse engineering. There aren't really many GUI testing frameworks for editors. And it is not really difficult for us to narrow down on using SWTBot. However, writing tests is not really straight forward as the tests involve asserting on appearance of widgets and refreshing editors. It also taught me about how to co-ordinate and finish the tasks while working in larger groups.

**Nisha:** This project changed my impression about testing and made me realize how significant testing is. I learned to write automated GUI testing using SWTBot and it felt good to see the green bar and UI running automatically. The most time consuming and challenging part was the reverse engineering where we had absolutely no clue initially where to start. The project served as a good means to boost my confidence to work on big size projects next time.

**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.

**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.

#### **4. Testing**

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:

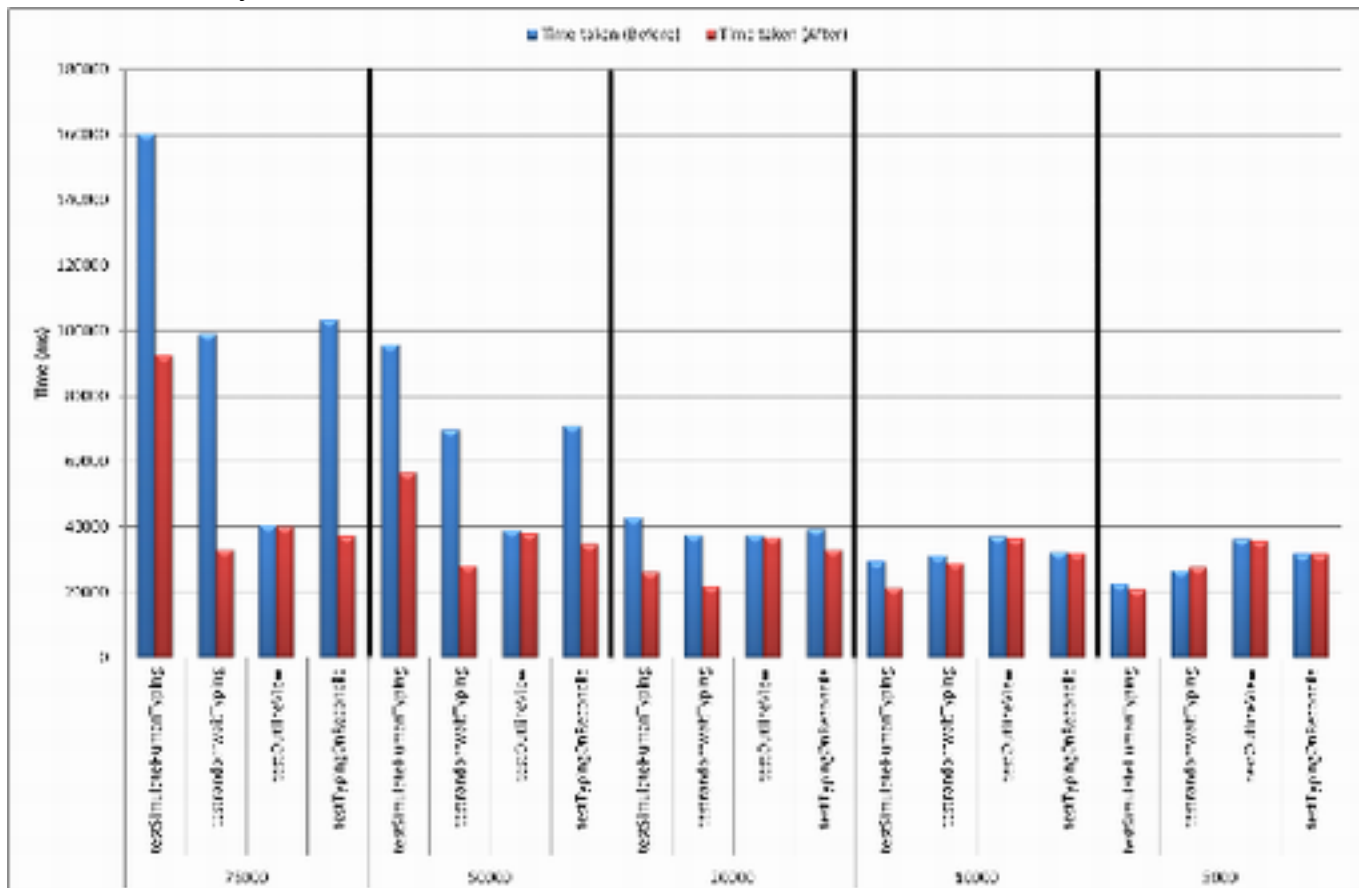
<http://wiki.eclipse.org/SWTBot/UsersGuide>

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

#### **Test Cases**

For each of the test case, we measure the time taken. The start variable initialized before typing starts. The end variable takes the time after typing ends. The difference in time is stored. These test cases were run first without the solution and the values were taken and populated in the test cases so that they can be compared with time take for the test cases with the solution. The assert statements are true if the new time taken is less than old time. There is a delay set after every character insertion to simulate typing.

- **testOutlineView** - Inserts a module in the large file and checks if the module is updated in the outline
- **testTypingOnReconcile** - Creates a delay such that reconcile is hit and then types again.
- **testSimulateHumanTyping** - Generates random line numbers, to insert text in the large file.
- **testrandomwaitTyping** - Generates random line numbers and random delays after every character insertion.



Size of file	Test Name	Time taken (Before)	Time taken (After)
75000	testSimulateHumanTyping	160318	92395

	testrandomwaitTyping	99219	32934
	testOutlineView	40586	40185
	testTypingOnReconcile	103495	37181
50000	testSimulateHumanTyping	95916	56627
	testrandomwaitTyping	69752	28060
	testOutlineView	39152	37986
	testTypingOnReconcile	71023	34670
20000	testSimulateHumanTyping	42842	26090
	testrandomwaitTyping	37559	21832
	testOutlineView	37551	36755
	testTypingOnReconcile	39333	32922
10000	testSimulateHumanTyping	29319	21293
	testrandomwaitTyping	31342	28849
	testOutlineView	37060	36517
	testTypingOnReconcile	32201	31960
5000	testSimulateHumanTyping	22490	20738
	testrandomwaitTyping	26613	27655
	testOutlineView	36217	35875
	testTypingOnReconcile	31825	31818

## 5. Appendix

### i. Installation instructions:

- Download the latest code from repository.

Repository location:

[https://subversion.ews.illinois.edu/svn/fa12-cs427/\\_projects/G22/](https://subversion.ews.illinois.edu/svn/fa12-cs427/_projects/G22/)



Packages to be downloaded:

1. org.eclipse.photran.ui (for automated tests)
2. org.eclipse.photran.cdtinterface.vpg
3. org. eclipse.photran.ui.vpg

**ii. Steps to run the program:**

No other changes need to be made. Editor can be used as before.

**iii. Steps to run the tests:**

1. Navigate to the following location:  
org.eclipse.photran.ui/trunk/src/org/eclipse/photran/ui/swttests/
2. Right click on the test files individually - > Run As -> SWTBot test