GSoC 2013 – Proposal

Organization name: The Eclipse Software Foundation
Project topic: Usage Data Collection for Code Completion
Bug ID: (Bug 401851)
Project: Code Recommenders
Mentor: Marcel Bruch

Student name: Timur Achmetow
Time zone: CET (UTC+1), summer: CEST (UTC+2)
University: Georg Simon Ohm University Nuremberg (Germany)
Course of studies: Software Engineering (Master)

Abstract

Extend the Eclipse Code Recommenders project by a specific dialog, which shows user specific statistics about the using with the Eclipse code completion and something else. This data will be graphical presented in an own dialog. Eclipse developers thus have the possibility to see how effectively they work every day with their IDE.

Detailed Information

Last year I have already participated in the Google summer of code 2012. I’ve learned a lot of the Eclipse Foundation through this great experience. I’ve participated in the Mylyn project and have extended the Mylyn Task-Editor for useful activity tracing features which are based on tasks/bugs (Traceability, for example reviews, builds, commits). Working with the amazing community has made a lot of fun. So I had the possibility to meet very interesting and great people. Personally, I could refine myself through the GSoC 2012 or rather the Eclipse Foundation very well. I got to know the entire process of the Eclipse Foundation and know how software can be developed in a professional environment. That was a very great experience for me!

Through this awesome project „Usage Data Collection for Code Recommenders“ by Marcel Bruch (Code Recommenders), other Eclipse developers have the possibility to learn a little more about their daily work in dealing with Eclipse. We want to display various metrics and charts regarding the work with the code completion engine and generally with Eclipse, for example, which Eclipse shortcuts are used and which other shortcuts maybe interesting for the developer. Another example could be, the number of times how much code completion was triggered, concluded or aborted, how much keystrokes saved by using code completion and much more. In my opinion these are very interesting data for every involved developer. This data can also be upgraded graphically, where different metrics are presented in the form of diagrams (see mockup).

I’m developing software since I was 20 years old, so software development is for me not only my job, but rather my hobby and I love it. And that’s great! At my first industrial placement I have learned C and C++, other languages with which I have worked: C#, Perl and Ruby. Since 2009 I work exclusive with Java and the development of Eclipse-RCP applications. At these work experience I have learned a lot of the Eclipse Rich Client Platform and another Eclipse technologies like SWT/JFace, SWTBot, plugin development etc. I think the Eclipse ecosystem is an awesome thing, and it fascinates me again and again. I’m just thrilled how well the whole Eclipse system works and lives.
Project Description

Description from mentor: Do you know how much time code completion saves you per day? No? Well, this proposal should address that by (locally) collecting usage data that can answer this and more sophisticated questions. Moreover, the collected data can then be used to make recommendations on the most effective use of your IDE, e.g., recommending a keyboard shortcut that you don't use (and presumably don't even know about).

With this project various metrics could be created, which provide the developer interesting facts about his work. For example, we could show how many keystrokes are saved from a developer through the code completion engine that would have to type manually without this engine. With this additional data we could also evaluate the developer activity. On which days/weeks is the developer more productive and show this interesting information in a chart (or block diagram). So we can see which days/weeks are the real working days/weeks and the other one. This stats center could extend through other Eclipse data, like the Eclipse shortcuts and something else. The following data we could show about the using of the code completion engine in a specific dialog:

- Number of times code completion triggered, concluded and aborted
- Number of saved keystrokes by using the code completion
- Types on which the code completion was used (fields, local variables etc.)
- In which classes code completion was used
- How much time code completion saves

Mockup

![Code Completion Statistics](image-url)
The dialog for this additional information could be split in several tabs. The first tab contains all metric data about the use with code completion. The next tab “Charts” could contain a few graphical analyses about the use of the code completion engine based on days or weeks. The “Conclusion” tab contains some summarized data about the developer himself and some suggestions how developer could be more productive with the Eclipse IDE. For this use case we could analyse in the background the used Eclipse shortcuts. With this data we know, which shortcuts are often used and which one are not used. With this information we could suggest which other common shortcuts are used by another proficient developers. This could help some Eclipse newbies, because they can learn very fast, which is the common way to develop productive software. But also real Eclipse experts will hopefully get some good tips.

![Graphical analysis of code completion statistics]

**Project Background**

I have already checked out the whole Eclipse Code Recommenders plugins from GIT and the project compiles now in my IDE. The next step would be to study some Code Recommenders tutorials, before I can really start with coding. Another task is to optimize the UI mockup together with the community through the Code Recommenders mailing list or BugZilla. The same applies to the architecture of the entire project at this stage; this activity has also to be done.

At first, I would probably need some help from my mentor, where the relevant code are placed and is interesting for this project. Thereafter, the entire UI and the business logic could be implemented. Also several unit tests and UI tests, if we need it.
**Deliverables**

In the final stage, of course, the implemented plugins should have been tested and checked out for their functionality (through the mentor). For this project I have planned the following milestones:

- **Milestone 1 (25. May):**
  - Create UI Mockup with the community
  - Study the documentation and the code base

- **Milestone 2 (20. June):**
  - Integrate various metrics and implement the infrastructure for this project
  - Pimp the dialog, e.g. with ui.forms and some cool icons etc.

- **Milestone 3 (15. July):**
  - Integrate the charts and conclusion tabs in the dialog

- **Milestone 4 (10. August):**
  - Create tests for this project, if possible I will do this parallel
  - Maintain the wiki and document the new created code

**Public Checkpoints:**

- **Checkpoint 1 (31. May):**
  - Announce the UI Mockup in a blog post or Wiki

- **Checkpoint 2 (30. June):**
  - Announce the dialog with the first tab (contains metrics)

- **Checkpoint 3 (31. July):**
  - Announce the chart and conclusion tab

**Tech-Spec**

Here are some tools or frameworks are listed, which I would to use:

- Attractive look & feel in the dialog
  - SWT, JFace und UI.Forms

- Persist user data
  - JSON: persist the java objects on the filesystem
    Plugin: com.google.gson.Gson
• Create charts
  o JFreeChart (Open Source under LGPL)
    http://www.jfree.org/jfreechart/

• Tests
  o JUnit, JUnit Plugin Tests and SWTBot tests

Expectations
I want to minimize the coaching time from my mentor as much as possible. Through my working experience I have some experience in Java and plugin development. After have a look at the code base I could start to develop the first features. Maybe I need some GIT support, because I don’t work a lot of with this SCM. But I think the EGit wiki is therefor a very good start.

Bio / About Me
I am Timur Achmetow (28 year old), a student from Germany. In October 2010, I'll be starting the Software Engineering course at Georg Simon Ohm University at Nuremberg (Germany), currently just finishing my Master Thesis. This is my second year applying for GSoC and I think the Google summer of code is a great thing for the organizations and students.

My technical skills:
• Java experience: a lot (5 years’ experience)
• Eclipse: a lot (5 years’ experience)
• Version Control Systems: SVN
• Build Tools: Maven (Tycho) and Ant
• Software Platforms: Eclipse RCP (5 years’ experience)
• UI Toolkits: SWT and JFace
• Test Frameworks: JUnit and SWTBot
• Static Code Analysis: PMD, FindBugs, CheckStyle
• Continuous Integration: Hudson