ECE450 – Software Engineering II

Winter 2007

Assignment 1 – Projects Selection and Exploration

Objectives

This assignment has three objectives:

1. To select the software projects with which your team will work for the rest of the term.
2. To familiarize yourself, if you weren't familiar already, with software development mailing lists, bug trackers, and repositories of software projects.
3. To perform an initial exploration of the technical and social aspects of your selected projects.

Due Date

This assignment is due at 5:00pm on February 1st, 2007.

Description

Throughout the term, you will be working with 2-3 open source software projects. For assignment 2, you will perform a comparison of the architectural structure and design of two similar projects (e.g. two RSS readers, two image viewers, etc.). For assignment 3, you will make a contribution to an open source project. For this contribution *you may want to choose to work with one of the two software projects that you will be comparing anyway*, instead of picking a third alternative to familiarize yourself with.

Your first task for this assignment is to select the projects that you will work with during the term. The open source projects of your choice need to cover the following requisites:

a. They should be relatively mature. Look for projects with an active development history of over 3 years. Younger projects *might* be acceptable, if they have a strong developer community.
b. They must have at least 3 developers actively contributing to it.
c. You and your teammates must be newcomers to the projects.
d. The project teams must have an active mailing list.

Your second task is to explore the work dynamics, software toolkits, documentation, and decision making processes of the projects you selected. Browse their bug databases and their mail archives. Download, install, and use their software. Log in to one of their chat sessions. This information will help you navigate through assignments 2 and 3 more easily, and will give you an idea of what you're up against.

You should summarize your findings in a report, including the following elements:

An **executive summary** that mentions the projects you selected, their nature, and the address where they are hosted.

An **introduction** that briefly summarizes your team's work processes, decisions, rationale for those decisions, your selected projects, and a few key bits of information about them (use your judgment).

A **section for each of your selected software projects**. In each project section, make sure to cover, at the very least, information such as:

- What is the objective of the project?
- What is the history of the project?
- What competing projects are out there?
- What is its organization?
- How active is its developer community?
- What software toolkit does it use (programming language, version control, bug database, etc.)?
- How are decisions made and conflicts resolved?
- How easy was to install or use their software?
- What were your impressions when using their software?

You may want to cover additional information that conveys a better idea on how the project's software and its development community work.

A **contribution proposal** for the project to which you wish to contribute. Remember that this project may be one of the two projects you will compare anyway, or it may be a third project. Valid contributions are code, tests, and/or documentation. You must demonstrate the need for your proposed contribution. *Do not overreach!* You have a limited amount of time, and you will not be able to do as much as it might appear at this point. Good judgment in selecting an appropriate chunk of work will be a component of your grade.

Finally, you may include **other sections** and **appendices** as needed.

Deliverables

You must submit an electronic report in HTML format as an attachment to your instructor's email account. Your report should be viewable in Firefox. Name your file: "[team name]-Assig1.html". The subject line of your email should be "[ece450] [team name] Assignment 1". The report should not exceed 3,500 words, excluding appendices. Treat this upper limit as a limit, not as a guideline. Your report will be made available to the rest of the group in a password-protected section of the website.

Marking Criteria

You will be marked on the quality of the descriptions of each of the projects you selected, on the adequateness of your contribution proposal, and on the quality of your writing.

Regarding the descriptions of each project, try to convey all essential information as if to an outsider of the project. Do not add superfluous details. Show that you understand the nature of the project, its challenges, and its culture. When the situation calls for it, be precise (e.g., when talking about the software toolkit of the project). Ensure that your

description of the software installation process is accurate. Show that you're thinking critically about the performance and usability of the software.

Regarding the adequateness of your contribution proposal, find the right balance between risk, conservativeness, and relevance: Trying to submit a patch to the Linux kernel is ridiculously hard; fixing the typos in a user manual is trivial; adding Esperanto support for a cooking recipes database is useless. If in doubt, ask your instructor first (but ask early!). Make sure you describe the contribution proposal precisely, unambiguously. Show that you have estimated this is an appropriate chunk of work. Show that the development community is interested in having this issue fixed.

Finally, regarding the quality of your writing, consider submitting your report to a writing centre if this is not one of your skills. A proper table of contents, labelled figures, documented references, and other elements of good writing style are required.

Suggestions

Start early. Seriously.

If you don't know where to start looking for open source software projects, try visiting Sourceforge (http://sourceforge.net/). Thousands of open source projects are registered and hosted there.

A good way to find the level of maturity of a project is to browse their bug database. Few bugs does not mean better software –it means few users and/or lack of interest. Tons of documented bugs suggest an active community of developers and users constantly trying to improve the code. No bug database suggests you should run away and don't look back.

You do not have to contribute with code to a project to get a good grade in Assignment 3. First, far too many projects suffer from deficient documentation, inexistent HOW-TO guides, lack of translations, etc. If programming is not the forte of your team, consider contributing with documentation. Second, thorough testers are often badly needed. Filing helpful and precise bug reports or generating a test suite can be useful contributions.

Make sure you can download, install, and execute the software of the projects you choose. You will be working with these projects –better make sure you can actually get them running, and that you don't hate working with them.

The best place to find project-related information is often on the project's mailing list. If the project is mature, the mail archive will likely be long and intimidating. As a favour to everyone, do not ask questions to the list without searching its archives first. If you need to send email to the mailing list, make it brief, unambiguous, and friendly. Developers are always willing to help if you ask the right way.

Use your team blog to discuss project-related matters. It's a good idea to leave a trail of evidence that you care about your projects, in case things go wrong later in the term.