University of Toronto

CSC408F/CSC2105F Software Engineering **Fall 2004/2005**

Project Software (50% of course mark) – OmniEditor package

1 Introduction and Motivation

Text editors are the best friends of programmers. There are, however, multiple choices of editors that may confuse a new programmer once he or she is used to one of them. It wastes the programmer's time to study a new set of key bindings as the time can be spent on more creative tasks.
For example, VIM (developed in C) and Emacs/XEmacs (developed in Lisp/C) were long seen as the best of breed in the programming world. They have different key bindings that lock the programmer in the pro/con camps. Yet another set of editors catches up with a huge set of new features, such as Eclipse and jEdit (developed in Java).
As software engineers, we want to bridge the gap between the different editors and help the programmer to use any text editor through familiar key bindings.
These open-source editors have been engineered for years. In addition, they are good quality editors that have been widely used as integrated development environments (IDE). Unlike commercial editors such as Microsoft Word, Notepad, Wordpad, UltraEdit etc., open-source editors can be studied and modified to satisfy end-user's needs.
In software engineering practice, a program with more than 75,000 lines of code (LOC) can be considered large. The following table lists some statistics of the above mentioned open-source editors. Note that XEmacs is a graphical user interface (GUI) variant of Emacs.

Table 1: LOC of popular open-source editors

editor	version	lines of code	language
VIM	6.3	260,623	C
Emacs	21.3	306,746 263,178	C Lisp
XEmacs	21.5.4	353,045 140,830	C Lisp
jEdit	4.1	114,117	Java
Eclipse	3.0	1,585,873	Java

The project involves software reengineering, testing and maintenance. The spirit of team work is very important to the success of software engineering.

Requirements

The easiest way of sharing information between different editors is through a file interface. However, it is inconvenient because it serializes the development, in other words, one editor can load a file

only after another editor has saved it. Another way is to share by Copy-Cut-Paste operations on a shared clipboard. Remote cooperation is limited because the clipboard is local. Therefore we would like to communicate the editing events across editors through an interoperatable interface, such as web services.
The idea of sharing editing skills among different IDEs is not new, for example, a VIM editing session can be associated with the NetBeans IDE (the one that accompanied with JDK) or Visual Studio. However, it is preferred to have a general design that any IDE can communicate with each other through a common interface.
The objective of the project is to bridge different editors through web services. For example, a programmer familiar with VIM can edit a text file while at the same time the change can be reflected in Eclipse, on a remote machine.
It is not required to reinvent the wheel, i.e., only need to use one legacy editor that exist for years. The students are not recommended to spend lots of time to write a new text editor. Instead should understand the existing one, and extend it.

Functional requirements

To be more specific, the end user of the target system should be able to edit in a local text editor while invoking text editing services provided by at least one remote editor to perform some tasks that helps the group. The cooperating text editors can be the same, both are VIM for example, but must reside on different machines.
The aim of this project is to find a subset of the smallest common divisors of editors such that together, the editors can fulfill useful editing tasks, such as

1. UPLOAD: inform the remote service to open a channel with the editing state, such as the content of the local text buffer and the position of the cursor in the buffer;
2. DOWNLOAD: query the remote service about the editing state of potentially other editors, such as the content of a certain channel and the position of the cursor; [1]
3. MOVE: move the cursor to up/down/left/right in the text buffer
4. MARK: mark the cursor for selection
5. SELECT: select a region from the mark to the cursor, if the mark is not set, the selection is the character at the cursor
6. TEXTSELECTED: return the string of the current selection
7. INSERT: insert a string at the cursor
8. DELETE: delete a current selection
9. FIND: search a string and move the current to the found occurrence, if it does not exist, then do not move the cursor and return false

[1] We may call the ”UPLOAD/DOWNLOAD” operations ”LOAD/SAVE” from the perspective of the service provider.

Features 1-9 are basic editing operations that are required to be implemented as a web service such that they can be programmably invoked by any remote programs. Each team should finish at least one of the following three sets of operations $\{1, 2, 3, 4, 5\}$, $\{1, 2, 6, 7, 8\}$, $\{1, 2, 9\}$.

Non-functional requirements

The correctness and reliability must be tested for the selected features. The reliability, the interoperability and extensibility are required too.
The responsiveness and usability are desired, but not required.

Deliverables

Phase A deliverables

Phase A involves requirement engineering and design recovery. The deliverable should be a document covers requirement specification, feasibility analysis, alternative selections and risk analysis, architecture design and test plan.

Phase B, C deliverables

Phase B involves the development of the followings.

1. Design web service for an individual editor. Each team must choose a programming language either C or Java, then design a web service based on either VIM (C) or Eclipse (Java). Specify your web service using a WSDL description.

2. Test your own web service and submit your test cases and their verifiable scripts;

3. Publish your web service with its description to the classmates (we will announce the official URL for each team).

The deliverable for Phase B includes the design documents and source code. The changes must be managed and the released functionality must be unit tested.
Phase C involves an invocation of the web services developed by other teams from the local editor developed by the home team. Each team is now responsible to integrate at least two services such that they can be used to do useful synchronized tasks (detail requirements will be announced later). The team must study the other's document and validate the other's services. The communication across teams must settle the defects and proceed. Each team must maintain the developed web-service too.