LECTURES NOTES*
Organisational Aspects of Software Development

Pedro Contreras
Department of Computer Science
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK
pedro@cs.rhul.ac.uk

1. Introduction

Creating software is a complicated task: organising activities, managing people, maintaining code, controlling the development process, assuring quality, estimating economically cost, resource allocation, etc. Software engineering gives an answer to most of these questions. The intention of this document is to discus some useful tools and give a few tips that can help at the moment of creating software.

This document is separated into two main sections. *Development Process and Collaborative Work* deals with tools and techniques to improve communication within Virtual Organisations. *Software Engineering Tool* deals with tools and standards to make coding faster and understandable for other programmers.

2. Organisational Aspects of Software Projects

A Virtual Organisation (or community) comprises a set of independent organisations (or persons) that share resources and skills to achieve its mission/goal, it is not limited to an alliance for commercial gain. The interaction among members of the virtual organisation is mainly done through computer networks. For example a group of people sharing a common interest can organise themselves by creating a virtual community through a web portal.

*Guest Lecture, 14 February 2007

From the software engineering point of view, a virtual organisation will be organised to tackle problems related with the software development process. Figure 1 shows a simplification of the software development process. Thus, in that figure it can be appreciated where some organisational problems may arise when developing software.

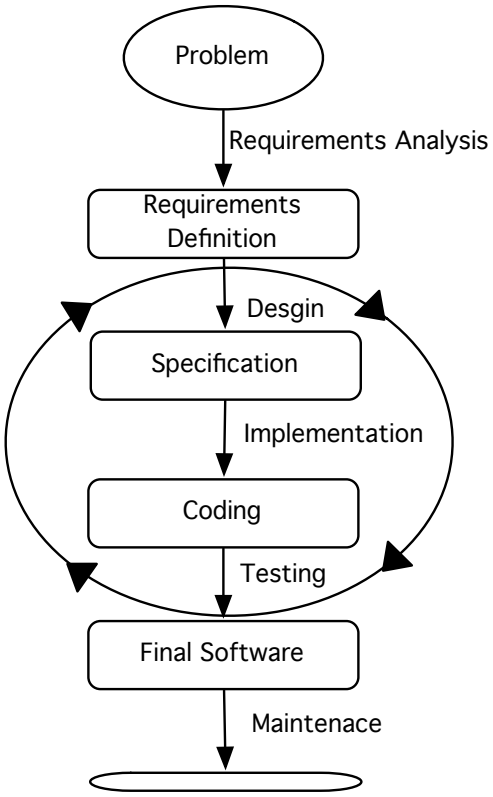

Figure 1. Software development process

The Virtual Organisation philosophy is usually one of an open collaboration. To allow people to work in partnership effectively, a range of tools are required. In collaborative software development projects there are different obstacles to overcome before the collaborative development can become effective. Firstly, good communication is essential in many aspects. Developers need to be aware of information regarding the project as a whole, information regarding specific sections, and they need to be able to directly communicate with other developers.

Adequate facilities for communication must be in place for developers to be able to exchange ideas, and to make, and to respond to queries, so that the development process is as efficient as possible. One important issue in relation to the development process cyclec is development by

second parties (other developers).

As well as communication facilities, there are other ways to ensure that the collaboration is as effective as possible.

Standards play an important role in facilitating code development by second and third parties. Standard software design methods will aid in code integration and coding style standards will help reducing the time required for second party developers to familiarise themselves with the software. Standard forms of code documentation can also greatly aid third party developers.

2.1. Communication Tools

A central portal can provide a common access point to the various communication tools available. For example Wiki pages can provide support by means of posting relevant documents. Wiki is a leading-edge, web-based collaboration platform targeting the corporate Intranet world. TWiki [7] fosters information flow within an organisation. Some Wiki features are: web browser, auto links, text formatting, search, e-mail notification, structured content, attachment, templates and skins, statistics and plug-in resources. Further to the Wiki pages a portal can also provide a forum facility to encourage grouped discussion.
In this case the groups deal with site-related topics, project-specific topics and general areas such as design, project organisations, and technical activities.
To allow developers to be kept up to date with various information about articles posted, items of news, events and polls, a news section also can be accessible. The idea of a central portal providing access to the various communication platforms is essential to effective collaboration. There is no confusion over how to contact other members as all contacts can be seen through the use of one central site. Further to these web-base community-oriented communication tools, one-to-one contact is necessary in order to ensure that developers can carry out their task efficiently. For example, in the case of second party development the ability to contact a previous developer to query some aspect of the work greatly reduces re-development time. To facilitate direct communications between project members, these can keep close contact through more traditional tools such as: e-mail, telephone conferences, chat programs and regular meetings.

A web portal can include one or more of the following tools:

- Wiki
- Internet Forum

- Blogs
- News
- e-mail (e-mail lists)
- Chat & instant messaging programs (e.g. MSN)
- Others: telephone, meeting, video conference

3. Software Engineering Tools

3.1 Coding standards & Java

The core task when developing software is writing the code itself. In the same way that any language has its own writing standards and protocols, coding in Java has some basic guidelines. This is with the propose of making the code more readable and understandable for other developers.//
The following describes some simple principles when writing code in Java.

- **Adhere to the style of the original:** When modifying existing code do not change the code's style, and do not try to change the old code to make it match a new style.
- **Adhere to the principle of least astonishment:** This suggests to avoid doing things that will surprise the software's users.
- **Do it right at first**
- **Document any deviation:** No standard is perfect and universally applicable. Thus when a situation that does not fit the norm arises, or when it is decided that a certain style should not be applied, then document why you have done so.

Following these simple principles may help producing better quality software. From the coding point of view this is not enough and additional conventions are needed for the following:

- **Formatting:** This include things such as indentation for block statements, breaking up long lines, and use of white spaces instead of "hard tabs". what looks perfectly formatted in one environment can look completely chaotic in another.
- **Naming:** Usually *Java Software Development Kit* conventions from Sun Microsystems are used (in Java case), which includes some of the following: how to name *classes, variables, methods, and constants*, and when to capitalise.

- **Documentation:** Write documentation for those who will be using the software as well as the people who will maintaining it. Documenting Java code for other programmers can be done by means of using comments and using the Java documentation package [5].

- **Packaging code:** Java code organises classes in packages, making sensible use of this help at the moment of reusing code. For example, when creating a new package include only related classes, since when using the package they need to be imported, and therefore if packages are not well organised the software becomes inefficient.

For complete the Java style guide refer to Vermeulen et al [8].

3.2 Interactive Development Environment Tools

Interactive Development Environment (IDE) is an integrated system to assist in writing software. Usually this system includes tools to help with code editing, graphical design, compiling and running programs, and debugging. Some examples of IDE tools include Visual C++ and Visual Basic (both from Microsoft Corporation). In the Java world there are plenty of good and free IDE systems, such as:

- Eclipse, `http://www.eclipse.org/`

- JBuilder Foundation, `http://www.borland.com/us/productsjbuilder/index.html#foundation`

- NetBeans IDE, `http://www.netbeans.org/`

3.3 Version Control System

A version Control System (CVS) [3] is usually a centralised place where files can be stored, and accessed from any machine with an Internet connection. Also it provides a way to store different versions of a document. If a specific document version needs to be recovered it can be done easily.
Figure 2 shows a typical CVS repository.

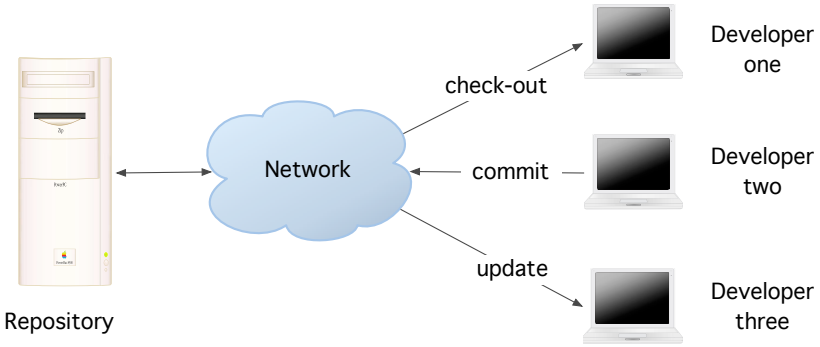

Figure 2. CVS working

A Version Control System includes the following sub-systems:

- The repository.
- Workspaces and file manipulation.
- Project, modules and files.
- Tags.
- Branches.

3.4 Bug Tracking System - Bugzilla

A key stage in the software life cycle is code maintenance and bug tracking. Bugzilla [2] is a free *Defect Tracking System* that allows the individual or groups of developers to keep track of outstanding bugs in their software effectively.
Bugzilla creates a web-based central repository to maintain a running list of reported defects and it status. Some of the following tasks can be carried out with Bugzilla:

- Track bugs and code changes
- Communicate with team-mates
- Submit and review patches
- Manage quality assurance (QA)

3.5 Apache Ant

Apache Ant [1] is an open source Java-based build tool. Some of the advantages of using Ant instead of other widely used building tools such as *make, gnumake, nmake or jam* are the following.

- **Ant build files are platform independent.** Ant resolves any platform dependencies such as Operating System (OS) commands (e.g. create directories) and how to format correctly the Java classpath.
- **Ant tracks files dependencies.** *javac* compiler is only invoked when source files have been changed. Thus when compiling files just the changes are recompiled and not everything.
- **Ant Java-based tasks.** Ant includes a wide range of *tasks*, that are very helpful for customising processes. For example Ant includes a task for running JUnit tests. Also Ant can be extended by writing custom tasks.

3.6 Software Testing, Unit Test

There are a lots of different kinds of test that can be performed on a software project. In some cases testing requires extensive feedback from the end users; other form of testing may required Quality Assurance team, or other extensive resources. Unit test, and more specifically JUnit [4] is a piece of code dedicated to exercise a very small, and specific functionality of the code to be tested. Basically it allows us to write a piece of code to test the software's specific sections in different situations.

3.7 File Access and Security

One problem not mentioned in the above sections is security and the identification of which information is publicly accessible and which is not. In open source projects almost every document and piece of code is publicly accessible through the Internet. Sometimes one exception is the code, this from the point of view of who is allowed to modify what or who is allowed to update code into a given repository. In any case usually downloading code is free and with no restriction (depending of course on the kind of licence we are dealing with, and the kind of project). Then there are restrictions in place as to who is authorised to upload and integrate code. In this case the SSH [6] protocol can be used to allow secure communication between developers and the server repository.

3.7.1 Secure Shell, SSH

SSH is a program for logging and executing commands into a remote machine. It provides secure encrypted communications between two untrusted hosts over an insecure network. SSH can use different authentication methods such as RSA (public cryptography) keys. Though remote log-in is the primary use of SSH, the protocol can be used as a general purpose cryptographic tunnel, capable of copying files, encrypting e-mail connections, and triggering remote execution of programs.

References

[1] Ant. Apache Ant, 2004. http://ant.apache.org/.

[2] BugZilla. Defect tracking system, 2005. http://www.bugzilla.org/.

[3] CVS. Concurrent Versions System, 2005. https://www.cvshome.org/.

[4] JUnit. Unit Testing, 2005. http://www.junit.org/.

[5] S. Microsystems. Java Documentation Specification, 2005. http://java.sun.com/docs/.

[6] SSH. Secure Shell, 2005. http://www.ssh.com/support/downloads/secureshellwks/non-commercial.html.

[7] TWiki. Enterprise Collaboration Platform, 2005. http://www.twiki.org/.

[8] A. Vermeulen, S. Ambler, G. Bumgardner, E. Metz, T. Misfell, J. Shur, and P. Thompson. *The Elements of Java™ Style*. Cambridge University Press, UK, 2000.