Organisational Aspects of Software Development

Pedro Contreras

Department of Computer Science
Royal Holloway, University of London

February 14, 2007

Introduction

Creating software is a complex task.

- Organising tasks
- Managing people
- Maintaining code
- Controlling the development process
- Ensuring quality
- Estimating economical cost
- etc.

Introduction

We present some useful tools tgether with a few tips that can help at the moment of creating software.

This presentation is separated into two main sections.

- **Development process and collaborative work** deals with tools and techniques to improve communication among Virtual Organisations.
- **Software Engineering Tool** deals with tools and standards to make coding faster and understandable for other programmers.

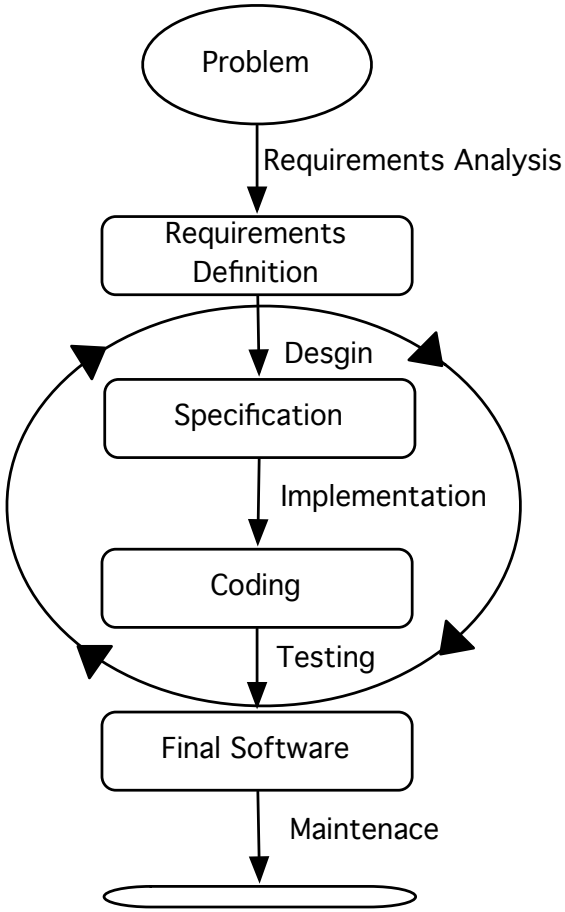
Virtual Organisation

Virtual Organisation (or community) comprises a set of independent individuals that share resources and skills to achieve its mission / goal, but that is not limited to alliance for profit enterprises. The interaction among members of the virtual organization is mainly done through computer networks.

For example a group of people sharing common interest can organise themselves by creating a virtual community through a web portal.

Software Development Process

From the software engineering point of view, a virtual organisation will tackle problems related with the software development process.

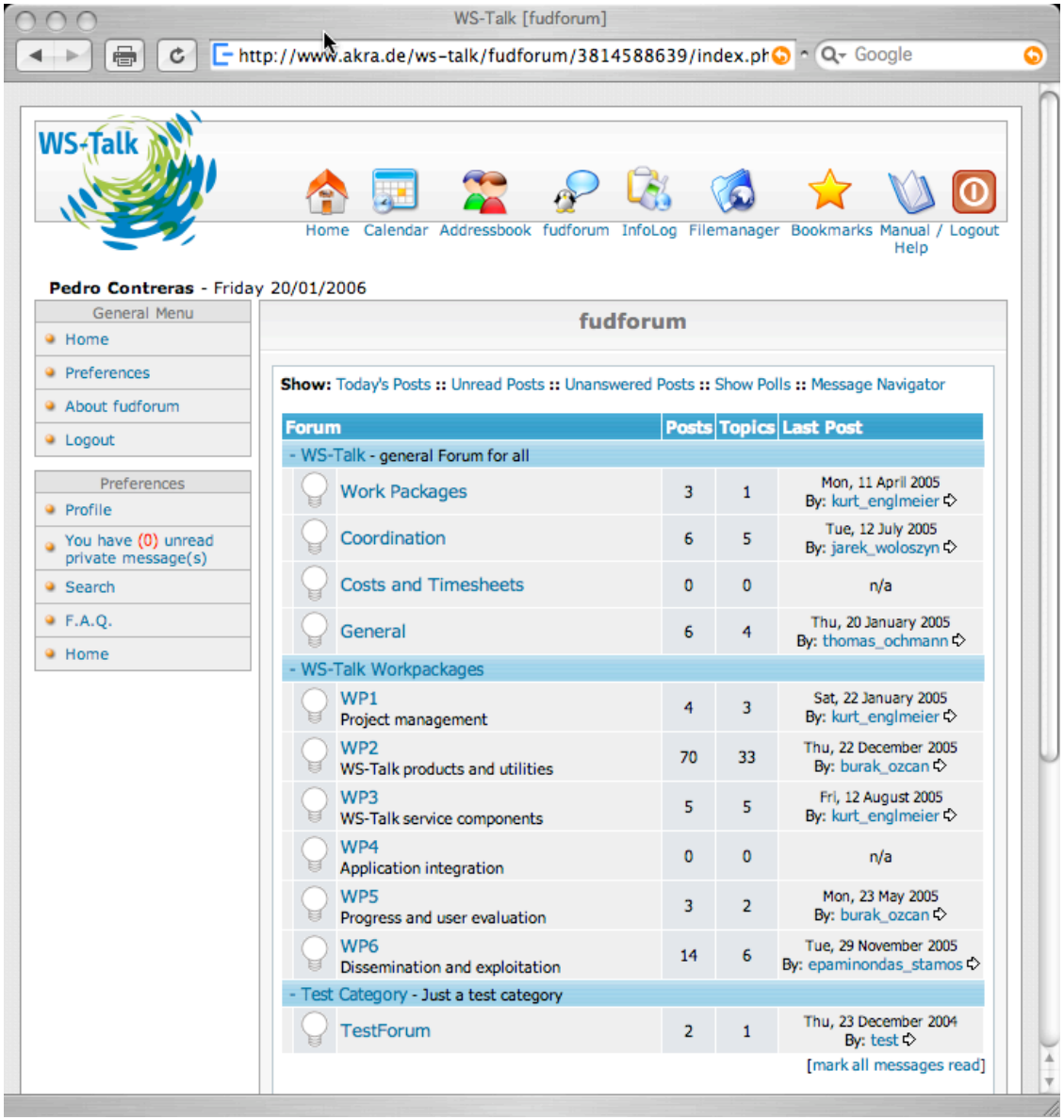


Collaborative Work and Tools

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

Wiki example

Forum example



Coding standards & Java

In the same way that any language has it own writing standards and protocols, coding in Java has some basic guide lines. This has the propose of making the code more readable and understandable for other developers. The following section describes some simple principles when writing code in Java.

Software Engineering Tools

- Adhere to the style of the original
- Adhere to the principle of least astonishment
- Do it right at first
- Document any deviation

Java coding conventions

- Formatting
- Naming
- Documentation
- Packaging code

Java coding conventions

- **Formatting:** This includes 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 as complete chaos in another-
- **Naming:** Usually Java Software Development Kit convention from Sun Microsystems are used, which includes some of the following: how to name classes, variables, methods, and constants, and when Capitalise.

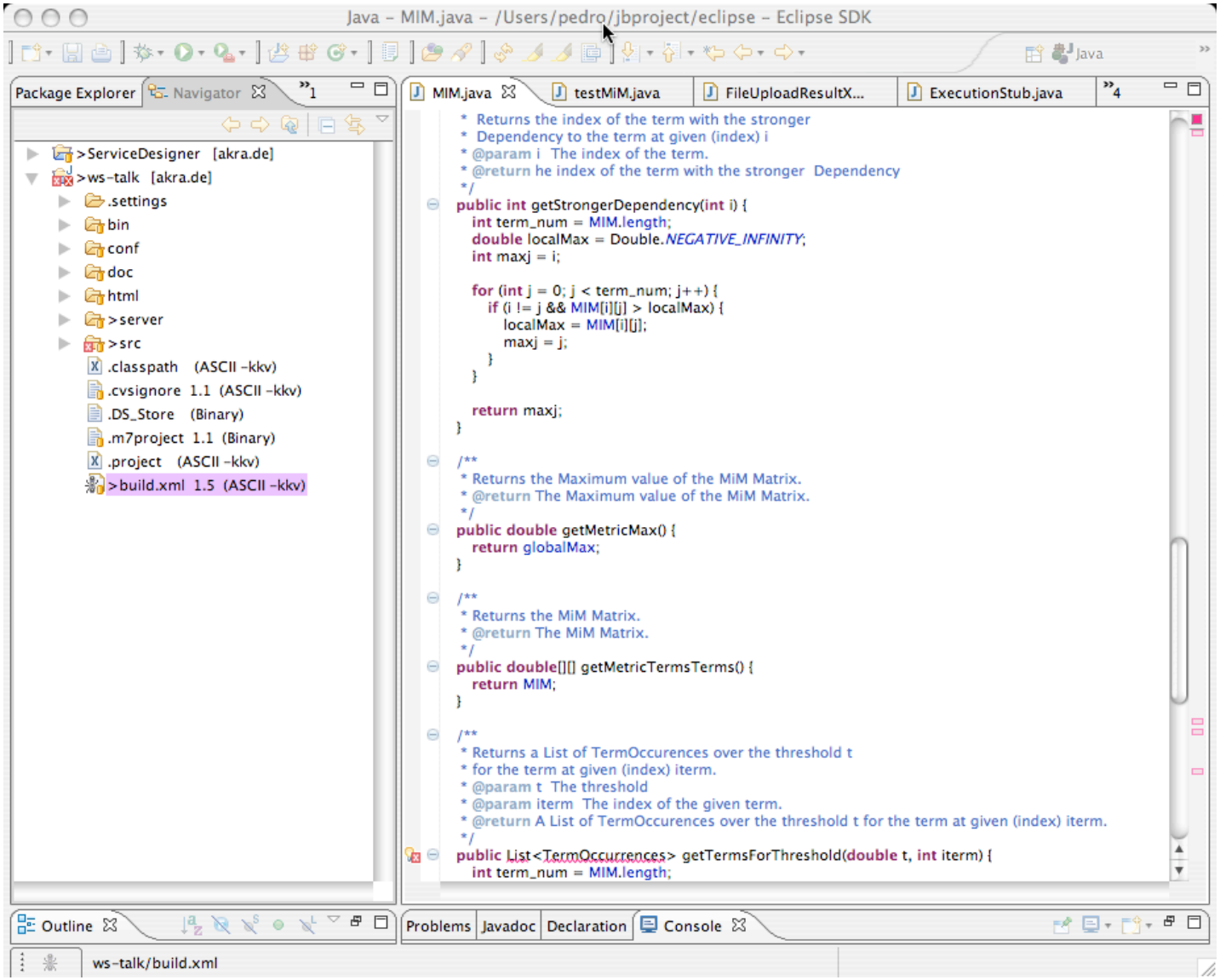
Software Engineering Tools

- **Documentation:** Write documentation for those who will be using the software as well as the people that will maintain it. Documenting Java code for other programmers can be done by means of using comments and using Java documentation package].
- **Packaging code:** Java code organises classes in packages, making at sensible to use this help when reusing code. For example, when creating a new package include only related classes, since when using the package it needs to be imported. Therefore if packages are not well organised the software becomes inefficient.

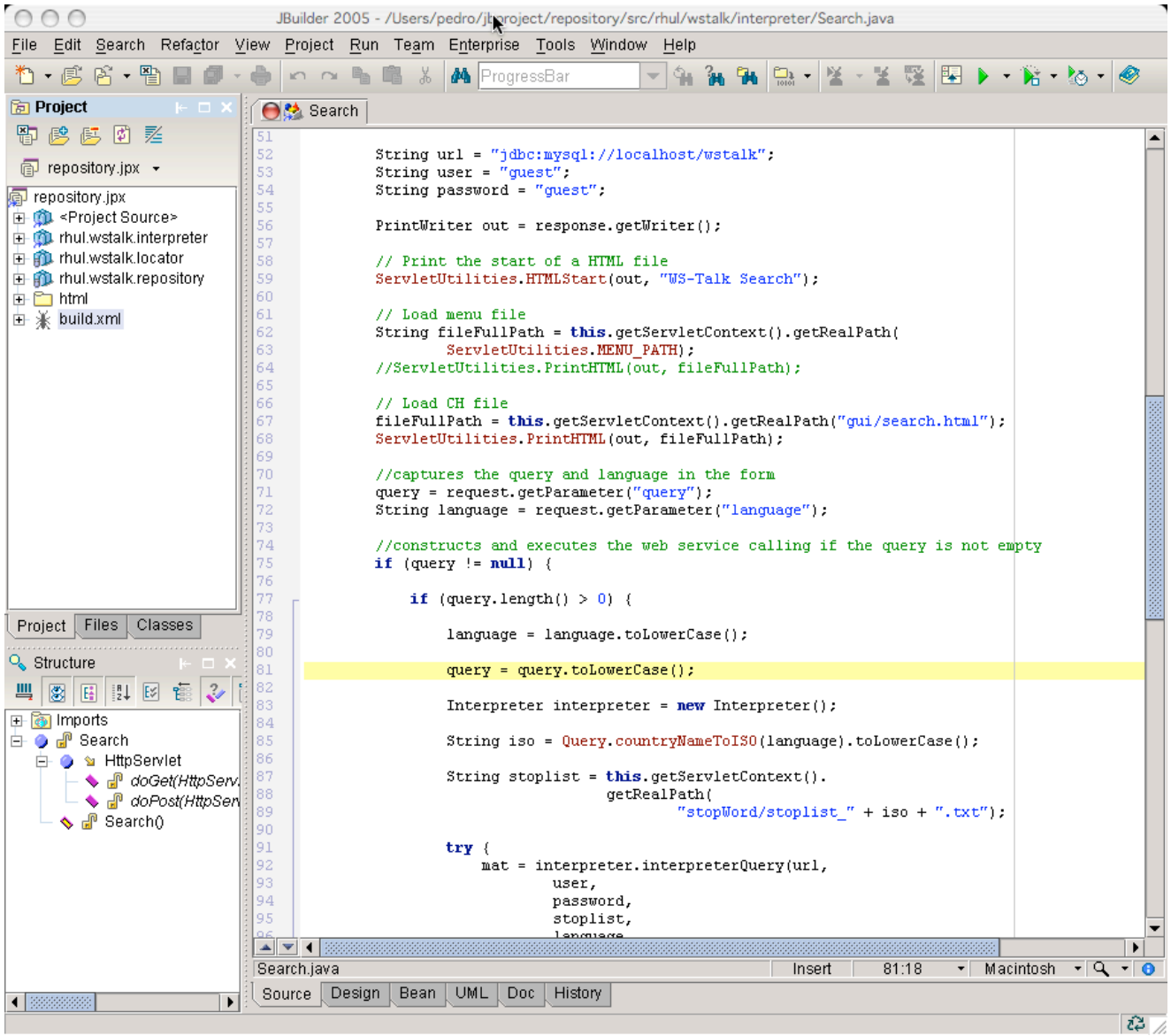
IDE Tools

Interactive Development Environment (IDE) is an integrated system to assist in the software writing; usually such systems includes tools to help with code editing, graphical design, compiling and running programs, and debugging

IDE Eclipse example



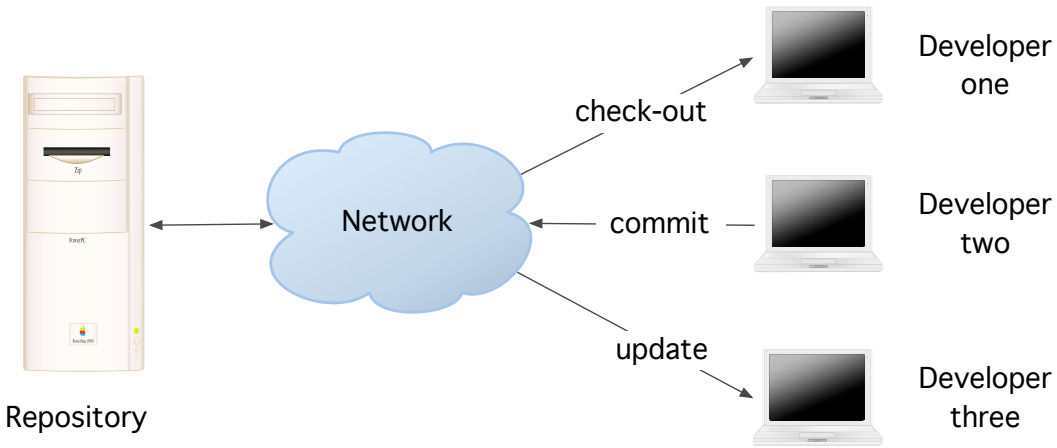
IDE JBuilder example



Version Control System

A version Control System is a centralised place where files can be stored, and accessible from any machine with Internet connection. Also it provides a way to store different versions of a document. Then if any version needs to be recovered it can be done easily.

Example: Concurrent Version System



Bug Tracking System - Bugzilla

Bugzilla is a free Defect Tracking Systems that allows 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 their status.

Bug Tracking System - Bugzilla

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)

Bugzilla engine example

Apache Ant

Apache Ant is an open source Java-based build tool

- Ant build files are platform independent
- Ant tracks files dependencies
- Ant Java-based tasks

Apache Ant

Ant build files are platform independent

Ant resolve any platform dependencies such as Operating System (OS) commands (e.g. create directories) and how to format correctly the Java classpath.

Apache Ant

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 task for running JUnit tests. Also Ant can be extends by writing custom tasks.

Apache Ant Example

build.xml

```
<project name="alwaysontop" default="make_jar" basedir="..">

  <!-- IMPORTANT VARIABLE HERE -->
  <property name="build.dir" value="C:/Projects_LocalCVS/DevDaily/AlwaysOnTopTest"/>
  <path id="class.path">
    <fileset dir="lib">
      <include name="**/*.jar"/>
      <include name="**/*.zip"/>
    </fileset>
  </path>

  <target name="init">
     <property name="project_name"   value="alwaysontop"/>
     <property name="jar"            value="${build.dir}/jar/${project_name}.jar"/>
     <property name="mainclass"      value="com.devdaily.alwaysontoptest.Main"/>
     <property name="sampleDataDir"  value="${build.dir}/data"/>
     <property name="nameOfDataDir"  value="data"/>
    <tstamp/>
  </target>

  <target name="create_classes_dir" depends="init">
    <mkdir dir="${build.dir}/classes-ant"/>
  </target>

  <!-- CLEAN TARGET -->
  <target name="clean">
    <delete dir="${build.dir}/classes-ant"/>
  </target>

  <!-- COMPILE TARGET -->
  <target name="compile" depends="clean,create_classes_dir">
    <javac destdir="${build.dir}/classes-ant" source="1.4" >
      <src path="src"/>
      <exclude name="**/_*.java"/>
      <classpath refid="class.path"/>
    </javac>
    <copy todir="${build.dir}/classes-ant">
      <fileset dir="${build.dir}/src">
        <include name="**/*.gif"/>
        <include name="**/*.jpg"/>
        <include name="**/*.png"/>
      </fileset>
      <fileset dir="${build.dir}">
        <include name="reports/**/*.*"/>
      </fileset>
    </copy>
  </target>

</project>
```

Software Testing, Unit Test

There are a lots of different kinds of testing that can be performed on a software project. In some cases testing requires extensive feedback from the end users; other testing form may require a dedicated Quality Assurance teams, or other extensive resources. Unit test, and more specifically JUnit, is a piece of code dedicated to exercise a very small, and specific functionality of the code to be tested

Secure Shell, SSH

SSH is a program for logging and executing commands into a remote machine. It provides secure encrypted communications between two non-trusted hosts over an insecure network. SSH can use different authentication methods such as RSA keys (algorithm for public-key encryption). 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

- Ant. Apache Ant, 2006. `http://ant.apache.org/`
- CVS. Concurrent Versions System, 2006. `https://www.cvshome.org/`
- JUnit. Unit Testing, 2006. `http://www.junit.org/`
- Sun Microsystems. Java Documentation Specification, 2007. `http://java.sun.com/docs/`
- SSH. Secure Shell, 2006. `http://www.ssh.com/support/downloads/secureshellwks/non-commercial.html`
- TWiki. Enterprise Collaboration Platform, 2006. `http://www.twiki.org/`
- A. Vermeulen, S. Ambler, G. Bumgardner, E. Metz, T. Misfell, J. Shur, and P. Thompson. The Elements of JavaTMStyle. Cambridge University Press, 2000.