Configuration Management

Bernd Schoeller
bernd.schoeller@inf.ethz.ch
Agenda for today

- Motivation
  - What is configuration management?
  - Why do we need configuration management?
- Configuration Management
  - Aspects of Configuration Management
  - Terms and Definitions
  - Problems and Solutions
- Tool Support
  - Available SCM Tools
  - Examples: CVS, Bugzilla, Xenofarm
- Summary
- PART 2: Subversion
About your Future

- You will never work alone.
- Other people will mess up your code.
- You will mess up other peoples code.
- You will never write a (major) program from scratch.
- The software that you will work on was already there when you joined the company.
- The software that you will work on will still be there when you leave the company.
"Configuration management is unique identification, controlled storage, change control, and status reporting of selected intermediate work products, product components, and products during the life of a system."

Anne Mette Jonassen Hass: "Configuration Management; Principles and Practice"
Configuration Management is about the role of **TIME** in software development.
root_class.e
readme.txt
gui/
gui/main_window.e
gui/dialog_window.e
net/
net/ftp_protocol.e
...

A system that stores and organizes documents over space is called a **FILE SYSTEM**.
A system that stores and organizes documents over space and time is called a **REPOSITORY**.
Configuration Management does manage not only source code, but also:

- Binary versions of the software (Builds)
- Requirement and analysis documents
- Design documents, UML-diagrams
- Test results
- System configuration information
- ...
Configuration Management is about ...

- ... knowing who has done what to which document in the past.
- ... enabling different people at different locations to work on the same set of documents at the same time.
- ... going back to an old version of the documents in the case that the path taken was not good.
- ... tracking the quality of the software over time and to stop software regression.
Versioning

- Versions give a unique **time-dependent identification** to each document.
- Deciding for proper version names is the **basis for successful software configuration management**.
- There can be **multiple levels** of versioning schemes within one project. Examples:
  - Versions for written documents:
    - Example: REQDOC-20040103-R4
  - Versions for source code:
    - 1.1, 1.2, 1.3, 1.4, 1.4.1, ...
  - Versions for binary builds:
    - 610, 611, 612, ...
Change Management

- CM has to record
  - WHICH document was changed.
  - WHAT was changed.
  - WHO has done the change.
  - WHEN was the change made.
- The history of the changes should be made visible.
- It should be possible to undo all changes.
- It should be possible to view the version of a document at a certain point in time.
Views

Time

FILE A
1.0 1.1 1.2 1.3
FILE B
1.0 1.1 1.2
FILE C
1.0 1.1 1.2 1.3 1.4 1.5

View at time T
Head = View at time $\infty$
Creating multiple versions of an existing set of documents is called **BRANCHING** (or **FORKING**):
Joining versions that were developed independently for some time is called **MERGING**: 

![Diagram showing version merging process](image_url)
Distributed Development

Developer 1

Repository

Developer 2

Checkout

Commit

Checkout

Commit

Conflict
There are different levels of binary builds:

- Daily (Nightly) Builds
- Integration Builds
- Stable Builds
- Releases
  - Alpha Release
  - Beta Release
  - Release Candidate
  - Official Release

Current, but not reliable.

Not current, but reliable.
Regression Testing

Regression Testing tests features that already worked in old versions of the software, if they also work in new versions of the software.

- A Test Suite is developed together with the software to verify its correctness.
- The Test Suite is run on changed versions of the software to verify that the old parts are still working.
- Regression Test are normally part of the Commit Process or of the Daily Build.
Bug-Tracking

- Bug-Tracking is the term for an *infrastructure* that captures and manages bug-reports in a given project.
- The number and quality of bugs is normally a judgment for the release-quality of software.
CM: Terminology

- Repository, Commit, Update, Checkout, Head
- Branch/Fork, Merge
- Conflicts, Resolving Conflicts
- Tag, Label
- Release, Build
- Test Cases, Test Suite, Regression Tests
Tools for Configuration Management

- Software Configuration Management (SCM) Tools
  - CVS, Subversion, Monotone, ArK, tla (FOSS)
  - ClearCase, BitKeeper, SourceSafe (Commercial)
- Bug-Tracking
  - Bugzilla (FOSS)
- Daily Build
  - Xenofarm (FOSS)
  - Tinderbox (FOSS)
- Most projects have customized software
CVS

- Best known SCM tool.
  - [http://www.cvshome.org](http://www.cvshome.org)
- Key to most open-source development projects and used in many companies.
- Based on RCS (Revision Control System)
- Command-line program (!)
- Graphical-User-Interfaces:
  - WinCVS (Windows) / GCVS (Unix)
  - Integration into Eclipse and many other IDEs
CVS (cont.)

- Features of CVS:
  - Central repository
  - Checkout creates a copy of the files on the local machine.
  - File-based versioning
  - Distributed work over the Internet with PSERVER (insecure) or SSH (secure) protocols.
  - Can work with binary and ASCII encoded files, but is not very effective with binary files.
  - Can do some primitive conflict resolving.
Bugzilla

- Open Source Bug-Tracking Tool
  - http://www.bugzilla.org/
- Developed for the Mozilla Browser Suite
- Web-Based Interface
Bugzilla (cont.)

![Bugzilla Interface](image)

### Bugzilla Advanced Search Features

- **Summary:**
  - `contains all of the words/strings`
  - `contains the string`
  - `contains all of the words/strings`
  - `contains all of the keywords`

- **Product:**
  - CSS
  - MailNews
  - Marketing
  - Minimo
  - Mozilla Localizations

- **Component:**
  - Sidebar
  - Simple MAPI
  - Skinability
  - Software Update
  - Spanish

- **Version:**
  - 1.0 Branch
  - 1.1
  - 1.2
  - 1.3

- **Target:**
  - Future
  - 3.0
  - Jan
  - M1

- **Status:**
  - UNCONFIRMED
  - NEW
  - ASSIGNED
  - REOPENED

- **Resolution:**
  - FIXED
  - INVALID
  - WONTFIX
  - DUPLICATE

- **Severity:**
  - blocker
  - critical
  - major
  - normal

- **Priority:**
  - P1
  - P2
  - P3

- **Hardware:**
  - All
  - DEC
  - HP
  - Macintosh

- **OS:**
  - All
  - Windows 3.1
  - Windows 95
  - Windows 98
Xenofarm

- System for a „Daily Build“.  
  - http://xenofarm.tigris.org/
- Developed for projects that target many platforms.
- Used in:
  - ArgoUML
  - Pike
  - Python
- Automated scripts will distribute the source code to build machines.
- Will run the compilation process and a set of test-cases.
Summary

- I have only given a glimpse at the topic of Configuration Management.
- Every software project should use it.
- Every software project should use it from the start.
- Having a large and long-term software project without it is foolish.
- Configuration Managements can be supported by proper tools.

But the key to successful configuration management lies in the discipline of the developers to use it.
Part II: Subversion
What is wrong with CVS?

- File based management
  - Rename file
  - Changing directory structure
- No atomic commits
- Network protocols are problematic
  - PSERVER is not encrypted
  - RSH is obsolete
  - SSH lacks anonymous access
  - Too much communication between client and server
- Inefficient storage of binary files
Subversion (also known as SVN)

- Available at
  - http://subversion.tigris.org/
- Full book on Subversion available at
  - http://svnbook.red-bean.com

“Subversion is meant to be a better CVS, so it has most of CVS's features. Generally, Subversion's interface to a particular feature is similar to CVS's, except where there's a compelling reason to do otherwise.”

Subversion Homepage
- Subversion uses a client server architecture
- Every developer work on his/her “working copy”
Version numbering is per “commit”

Time

FILE A
1 3 6 8

FILE B
1 5 8

FILE C
1 2 3 4 7 9

Head = View at time $\infty$
Version numbering in Subversion

Version numbering is per "commit"

Time

FILE A

1 2 3 4 5 6 7 8 9

FILE B

1 2 3 4 5 6 7 8 9

FILE C

1 2 3 4 5 6 7 8 9

Head = View at time $\infty$

HEAD
HEAD = revision 9
BASE = revision 7
COMMITTED = revision 6
PREV = revision 5

Working Copy taken at revision 7
Subversion Commands

- `svn checkout`
- `svn update / svn revert`
- `svn commit`
- `svn info / svn log / svn status`
- `svn add / svn delete / svn move / svn mkdir`
- `svn copy`
- `svn diff`
- `svn merge / svn resolved`
- `svn cat / svn list / svn blame`
- `svn export / svn import / svn switch`
Eclipse Plugin subclipse (http://subclipse.tigris.org)
Even more ...

Windows Explorer Integration: http://tortoisesvn.tigris.org
Using the command line, I will:
1. Create a repository
2. Import some files
3. Checkout a working copy
4. Modify and commit
5. Rename a file
6. Create a branch
7. Modify branch and trunk
8. Merge branch back into trunk