

Configuration Management*

*From <http://www.cs.otago.ac.nz/cosc345/lectures.html>

Configuration Management

Change is constant
teams need communication
Poor communication = confusion

Goals of configuration management:
Minimize confusion so more work can be done.
Maximize productivity by minimizing mistakes.

Why Is It Needed?

- Problems with large projects:
 - multiple developers,
 - shared data,
 - double maintenance,
 - simultaneous updates.

Multiple Developers



One communication path



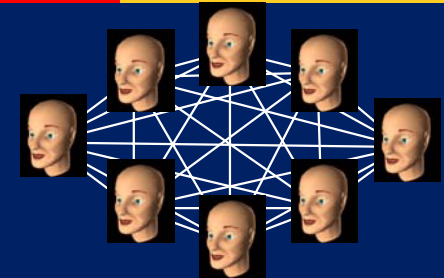
- takes a copy of files,
- makes changes,
- copies changed files

- takes a copy of files,
- makes changes,
- copies changed files,
- integrates changes

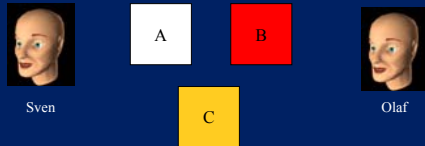
Multiple Developers

- Shared Data
 - each developer has their own copy of files,
 - “Who Changed What and When?”
- Double Maintenance
 - How do we make sure that these multiple copies are *identical*?
- Simultaneous Update
 - developers change same file at same time

Group Communication

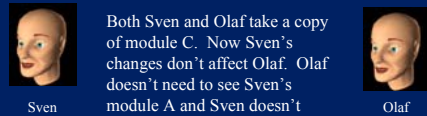


Shared Data



Sven changes module A and has to make a change to shared module C.
Olaf's code in module B also uses module C.
Sven's change breaks Olaf's code.
Olaf doesn't know what has changed and when.

Fixing this problem

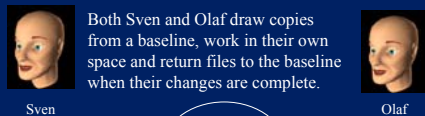


Both Sven and Olaf take a copy of module C. Now Sven's changes don't affect Olaf. Olaf doesn't need to see Sven's module A and Sven doesn't need to see Olaf's module B.

A new problem

- Now Sven and Olaf have multiple copies of module C,
- Sven's copy of module C is different.
- But module C is supposed to be shared.
- What happens when we integrate the project?
- This is *Double Maintenance*

Double Maintenance

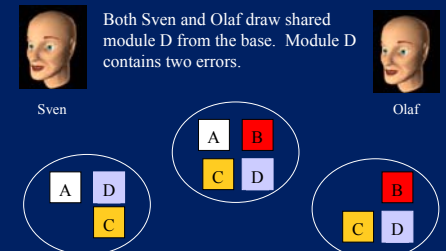


Both Sven and Olaf draw copies from a baseline, work in their own space and return files to the baseline when their changes are complete.

Baselining

- The baseline defines known modules,
- When Sven makes a change, it is logged.
- Olaf can draw Sven's module C from the base,
 - it will still break his module B,
 - he will have to fix his module B,
 - but at least he knows why!

Simultaneous Update



Both Sven and Olaf draw shared module D from the base. Module D contains two errors.

Simultaneous Update

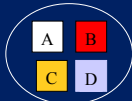


Sven

Sven finds one error. Olaf finds another error. They fix the errors and both check module D back into the baseline.



Olaf



...then Olaf checks in his module D

D

Sven checks his module D in first.

Sven's changes get lost!!

D

What is involved?

- Not just a software tool
 - naming conventions,
 - policies
 - procedures to make sure parties are involved
- Document the above:
Software Configuration Management Plan

What does the plan give us?

- We know how to report a *problem*,
- We know how to request a *requirement*,
- Everyone is *informed* of suggested changes and opinions are *solicited*,
- Change requests are *prioritized* and *scheduled*,
- Intermediate and final products are *under control*.

How does change happen?

- By requesting a new requirement,
- By fixing a known problem,
- After discussion with all parties,
- According to the priority schedule drawn up by the *Change Control Board*.
- If other kind of change is made, then the project is *out of control*.

What items are under control?

- System Specifications and Requirements,
- Documentation,
- Source Code,
- Test cases.

Baselines

- Software items that are under control form a *baseline*,
- Developers draw items from the baseline that represents a particular version,
- *Versions* are not the same as *releases*.
- A baseline might be associated with a milestone or build number.

CM and Project Management

- Configuration Management is *not* Project Management.
- Project Management defines:
 - a plan and a release schedule,
 - milestones and deliverables,
- Configuration Management provides:
 - tracking of changes,
 - controlled development.

What kinds of Control?

- Version Control
 - track *revisions* to *items*
 - *who* made the change, *when* and *why*?
- Release Control
 - which *items* are built into which *release* of the system and *when* is the release finished?
- Build Control
 - *how* do we build this release?

What Is In The Plan?

- Naming convention for items,
- Who has responsibility...
 - ...for approving a change?
 - ...for making a change?
- What policies exist for change control?
- Where and how CM information is held.
- How software tools to manage the CM process are configured and used.

When Does Change Happen?

- In response to new requirements,
- To fix errors during development,
- During Maintenance:
 - *corrective* maintenance,
 - *adaptive* maintenance,
 - *perfective* maintenance.

Version Control

- Control changes at development level,
- Control multiple changes to multiple files
- Provide *concurrency*,
- Provide *traceability* and *visibility*:
 - who changed what, when and why?
 - Allow change history to be viewed.
- Examples: PVCS, SCCS, RCS, CVS.

Build Control

- How is the system built?
 - Project files, Makefiles, build scripts,
 - GNU **autoconf** (platform variances).
- System integration is part of CM.
- How is the system integrated?
 - Daily?
 - According to the PM schedule?
 - Towards the end of the project?

Build Control...

- Integration strategies:
 - phased,
 - incremental.
- The daily build
 - role of the Build Master,
 - breaking the build,
 - Always Having a Working System™

Release Control

- When is the system released to the client?
 - ...when it works (and the client believes it)?
 - ...when 80% of it works?
 - ...when we reach the deadline?
- What do we call the release?
 - Version 1.0, 1.1, 2.0, 2000,
 - major and minor version numbers,
- Release policy

Summary

- CM is an essential part of a project,
- It's purpose is to provide control and minimize confusion,
- It can be made to fit in many ways,
- Many tools exist to support the process.