Modulo II
Software Configuration Management - SCM

Professor

Ismael H F Santos – ismael@tecgraf.puc-rio.br

Bibliografia

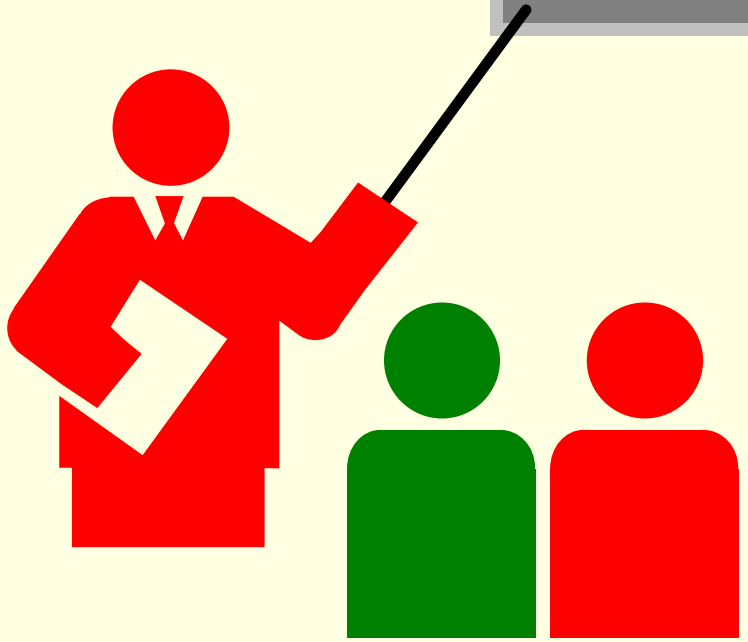
- *Introduction to Apache Maven 2*
  - *Tutorial ibm developerWorks:*
- *Introduction to Maven 2*
  - *http://www.javaworld.com/javaworld/jw-12-2005/jw-12-2005-maven.html*

Ementa

- SCM – Introducao
- Subversion
- Maven SCM plugin

MTSW-Java

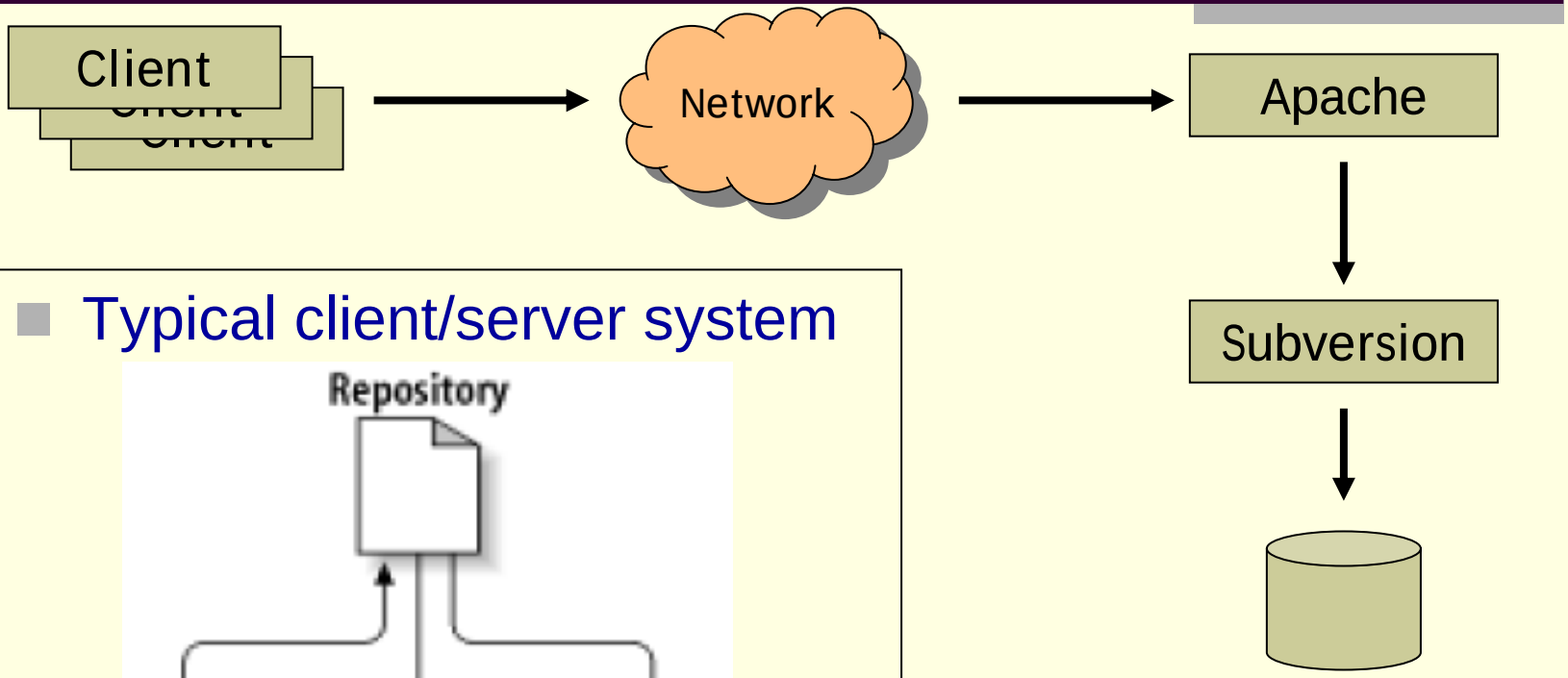
Introdução



Version Control System operations

- Initialize
- Add Files
- Commit
- View History
- Share
- Update
- Revert
- Branch

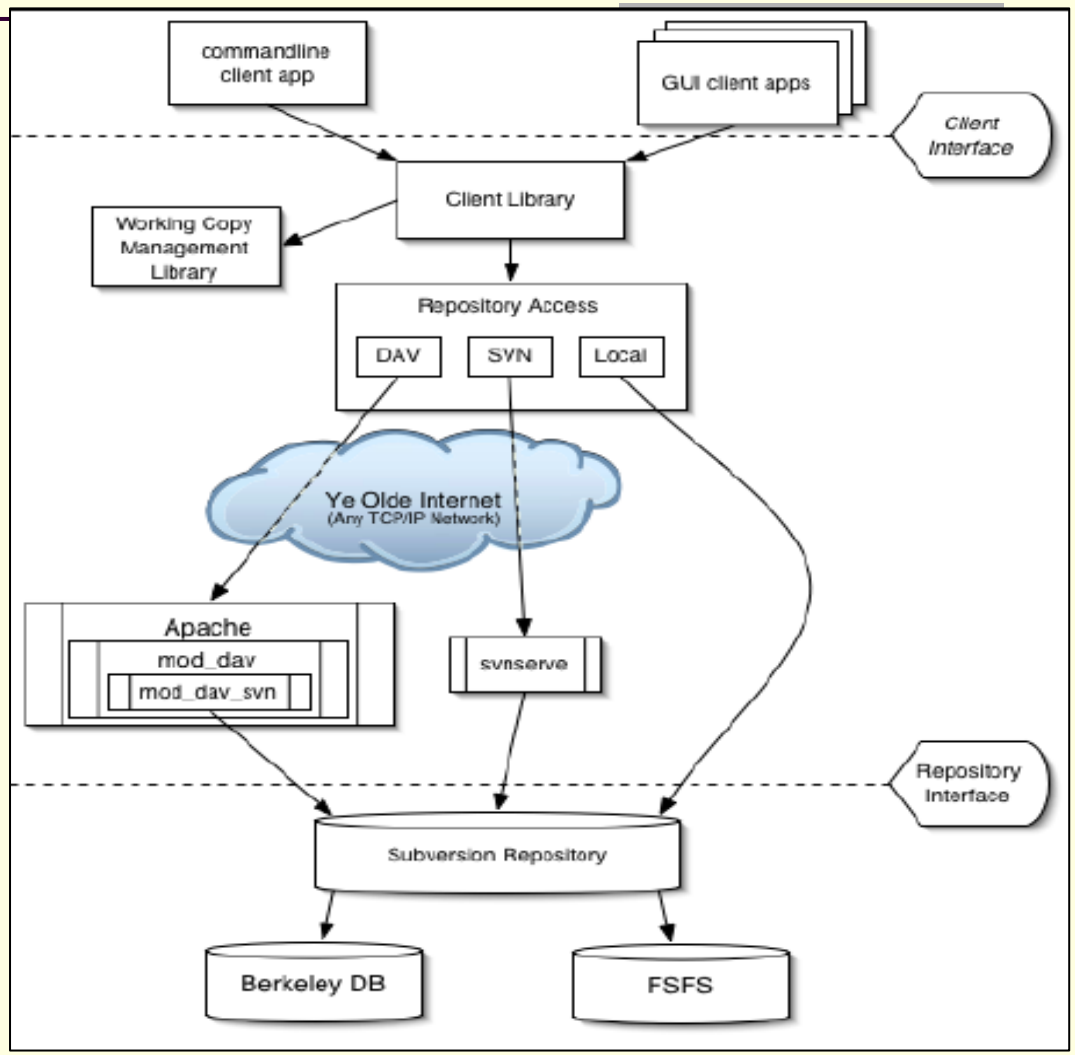
Remote Users



- Typical client/server system

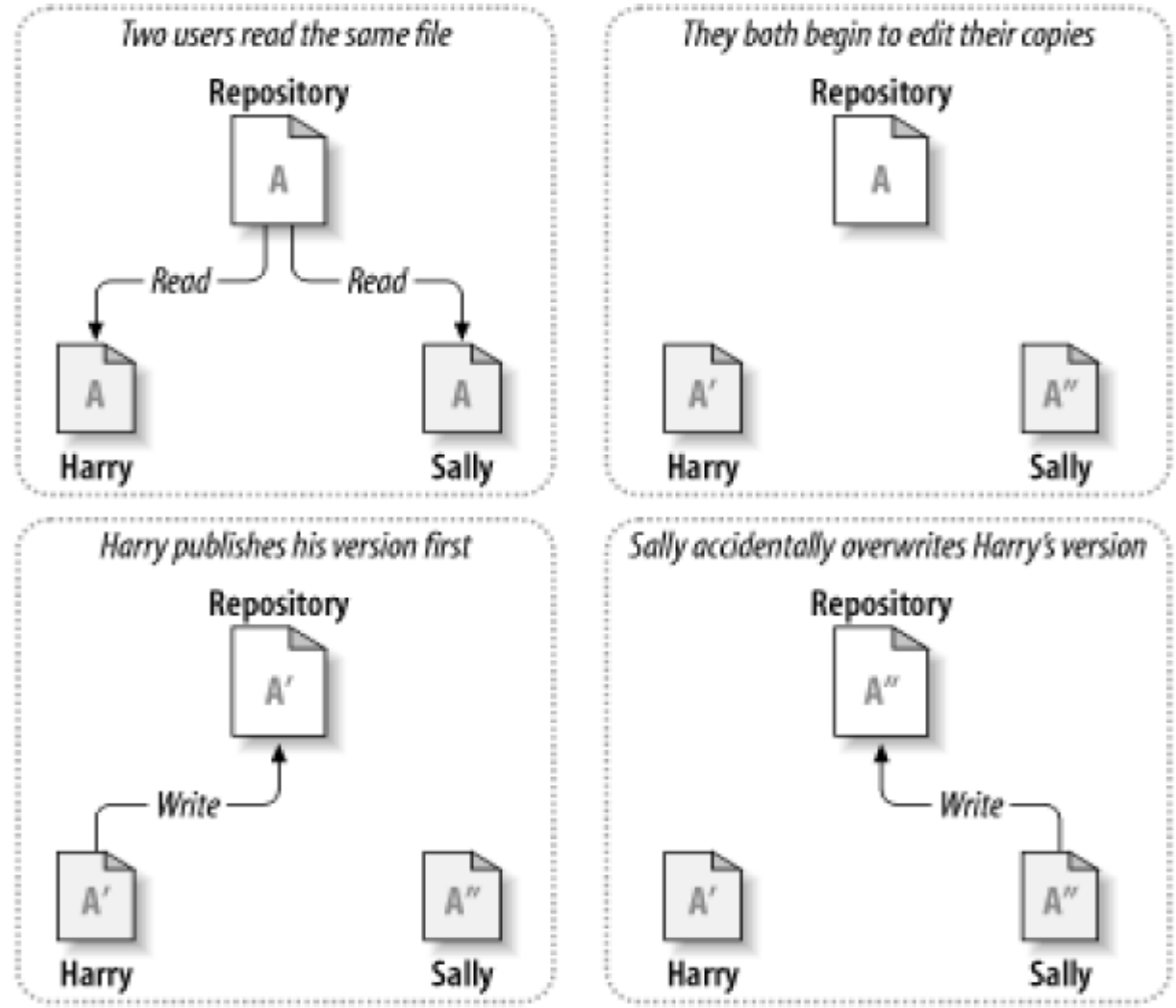
Subversion Architecture

- SVN Clients & Server
- SVN Repository



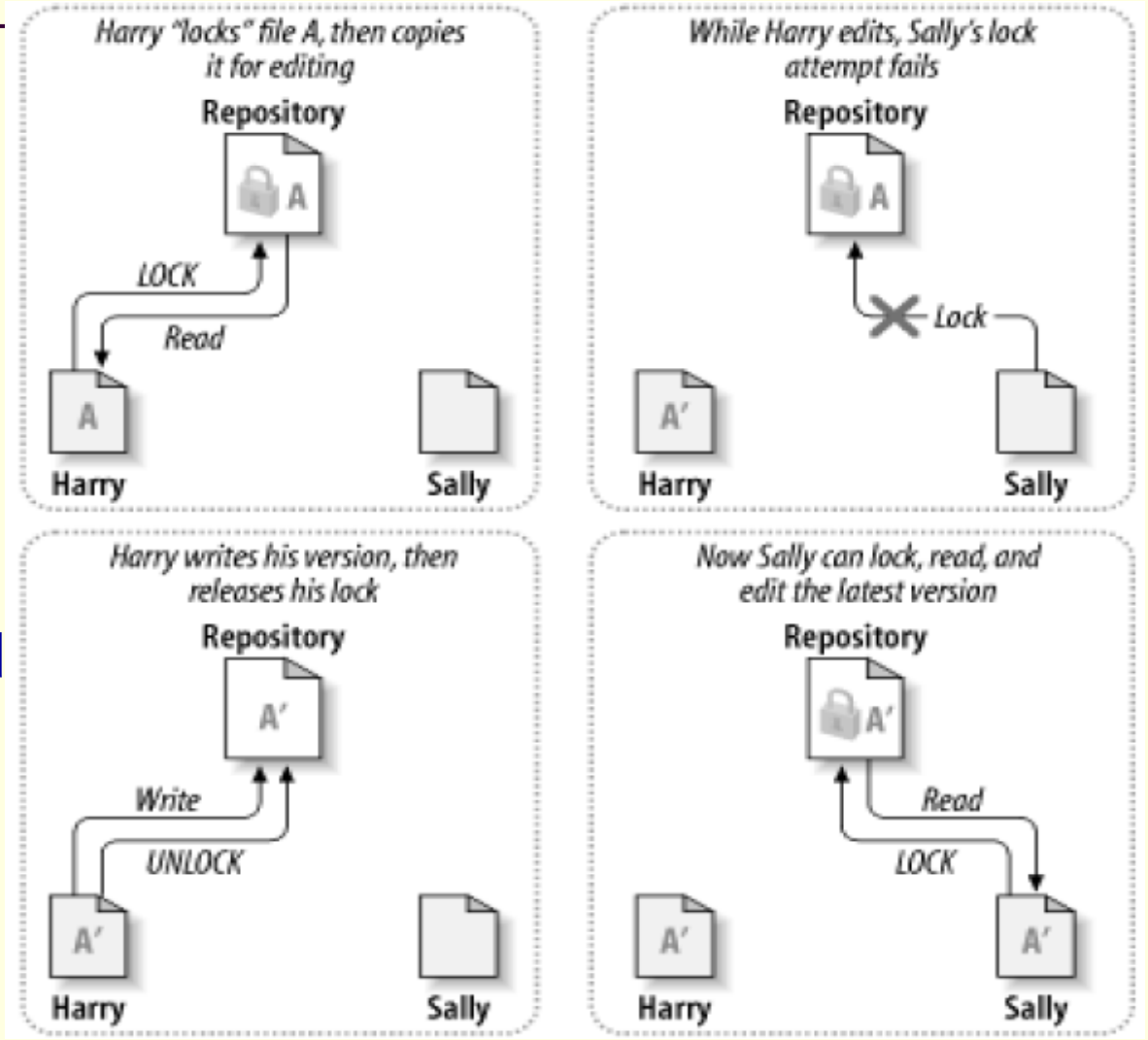
The Problem of File-Sharing

- Problem to avoid !!!



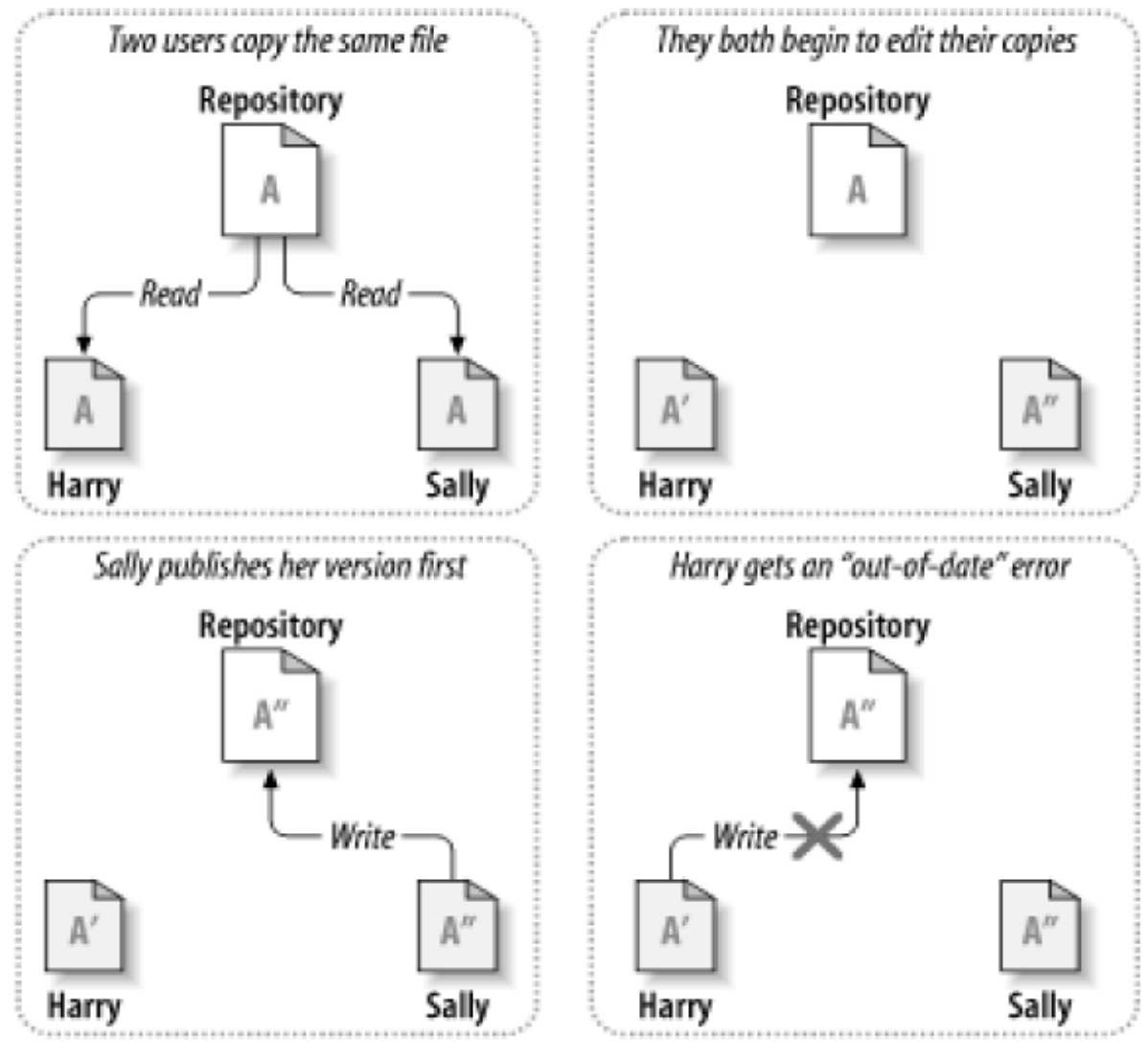
The Lock-Modify-Unlock Solution

- Used by Many version control systems
- The repository allows only one person to change a file at a time.
- This exclusivity policy is managed using locks.
- Very restritive solution, lock related problems



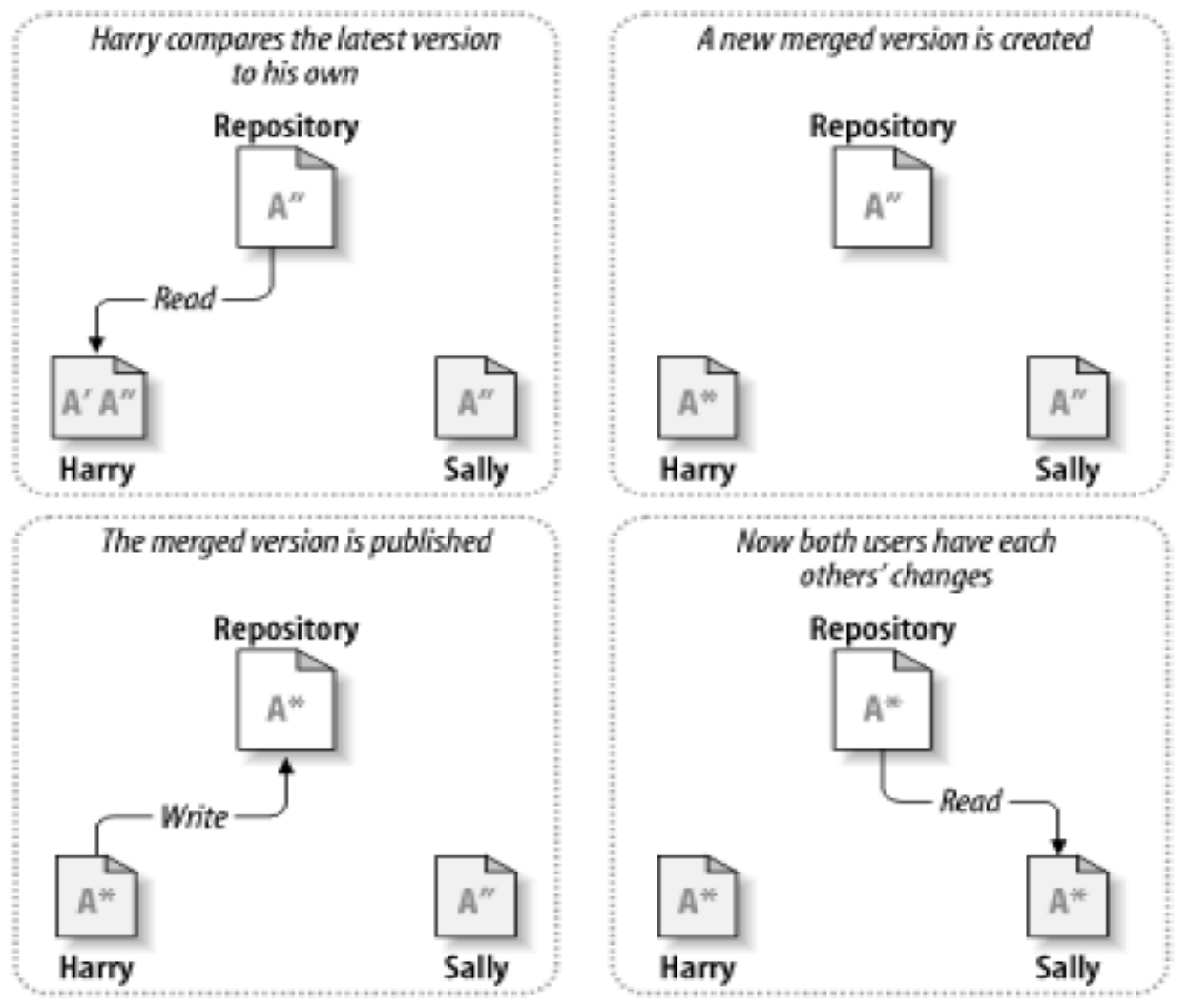
The Copy-Modify-Merge Solution

- Subversion, CVS, and other version control systems use it
- Users can work in parallel, never waiting for one another.
- When people work on the same files, it turns out that most of their concurrent changes don't overlap at all;

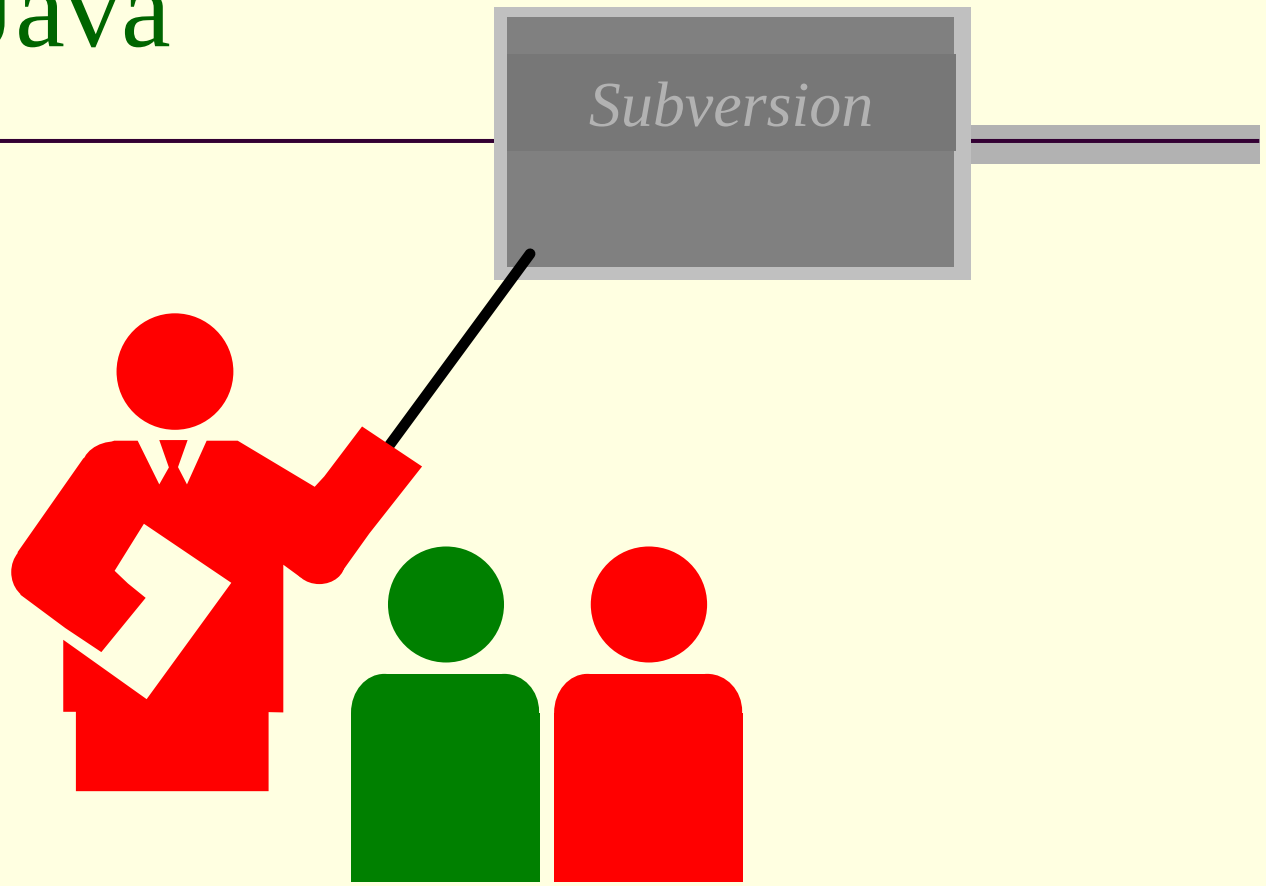


The Copy-Modify-Merge Solution

- conflicts are infrequent.
- the amount of time it takes to resolve conflicts is far less than the time lost by a locking system.



MTSW-Java



Subversion Basics

- Open Source
  - http://subversion.tigris.org/
- Designed for remote users
- Unreserved checkouts
  - Similar to CVS
  - Checkout, edit, merge, commit

Subversion Basics

- Many clients possible
  - Command-line client (included)
  - Windows GUI client (in process)
  - Other clients (more GUIs, integration with IDEs, etc)
- Apache-based network server
  - WebDAV-based network protocol
  - High performance, scalable, secure, and more

Why Open Source?

- If it is great, then it could replace CVS
  - Widespread usage would establish it as a legitimate tool
- More and better clients are possible
- Peer review and testing
- Community feedback and contributions

Basic Method of Operation

- Similar to CVS
- Unreserved instead of reserved
- Client-side working copies
- Four basic steps
  - Check out a “working copy”
  - Make any edits
  - Merge changes from server
  - Commit your changes

Clients

- Command-line client comes standard
- Windows client: TortoiseSVN
- Linux: GTK (X Windows) client
- Integration with IDEs: SubEclipse plugin
- WebDAV clients, too!
  - Microsoft Windows, Office, and others
  - Open source clients such as cadaver, Nautilus, and Goliat

Apache as a Server

- WebDAV (DeltaV) for the network protocol
- Apache 2.0 and mod_dav
- mod_dav_svn as glue to Subversion

- High performance, scalability, and robust
- SSL provides encryption and authentication
- Proxy/firewall friendly

Subversion vs CVS

- Most CVS features
- Directory versioning
- Metadata
- Atomic commits
- Branching and tagging are cheap
- Designed for the network
- Better binary file handling
- Layered library design

Revision Numbering

- Global revision number rather than per-file
- Allows you to talk about “revision 2524”
- Unique identifier for a state of your project
  - Simple way to tag (next slide)
- Each revision corresponds to a single commit
  - Contains author, log message, and date

Tagging and Branching

- Subversion implements “cheap copies”
- Branches are just copies of the main trunk
  - Make changes in the copy
  - Merge the changes back to the main trunk
- Tags are copies which are never changed
  - Might not even be necessary if you simply record the global revision number that built your product

Example Repository Layout

```
http://svn.example.com/repos/project/
   trunk/
      source/
      docs/
      buildtools/
   branches/
      issue-1003/
      gstein/
   tags/
      alpha-1/
      1.0.0/
      1.0.1/
```

Just an example – you are free to structure the repository in whatever way fits your project's needs and goals

Authentication

- CVS uses a custom authentication mechanism
  - Part of CVS's custom (non-standard) protocol
  - "I LOVE YOU" or "I HATE YOU"
  - pserver sends passwords in the clear
- Alternate authentication schemes
  - kserver, gserver
  - SSH tunneling
- Subversion uses HTTP as its protocol
  - Integrates with existing authentication systems
  - Standardized!

Modules

- CVS modules
  - Live in CVSROOT
    - The "**`modules`**" file
    - Only the administrator can alter module definitions
  - Only apply to checkout
    - Changes are not detected during "**`cvs update`**"
- Subversion modules
  - Directory property ("**`svn:externals`**")
    - Users can define them, edit them, inspect them
    - Attach to any directory
  - Versioned, as with any property
    - Changes are detected during "**`svn update`**"

Keywords

- CVS keywords are automatically expanded
  - User must explicitly disable this behavior
  - Risk of destroying a binary file
- Subversion keywords are optionally expanded
  - User must proactively enable keyword expansion
  - The user states the set of keywords to expand (some or all)
  - The behavior is controlled by a property: **`svn:keywords`**

Directory Versioning

- Directories are versioned items
- Deletes and moves are recorded
- Copy sources are remembered

Metadata

- Any file or directory can store properties
- Properties are name/value pairs
- Some standard properties exist
  - **`svn:ignore`**
  - **`svn:mime-type`**
  - **`svn:eol-style`**
  - etc.
- User-defined properties are allowed
- Property values can be text or binary

Improved Features (1 of 2)

- Atomic commits
  - CVS can commit one file, fail on the second
  - Subversion commits **all** changes, or nothing
- Binary file handling
  - Subversion uses MIME types
  - Binary diffs
- Newline and keyword handling is improved
  - Subversion does not munge your files until you tell it to

Improved Features (2 of 2)

- Designed for the network
  - CVS had network support “bolted on”
    - Two code paths to maintain
    - Authentication poorly integrated
  - Subversion has network support (based on WebDAV/DeltaV) that was planned from the very early stages
  - Binary diffs in both directions

Layered Library Design

- Many levels for interaction
  - High-level client
  - Repository access (local, remote, pipe, etc)
  - Direct access to the storage
- Enables scripting
- Clients can share a lot of code
  - The command-line client is a small application built on top of the high-level client library
  - GUI clients can also use the high-level library
- Library-based enables third-parties

Details: repositories

- Subversion uses URLs for repository locations
  - **`http://svn.collab.net/repos/svn/`** is the actual URL for Subversion itself
- Web browsers can view the “head”
  - Full ViewCVS-like functionality coming soon
- “file” URLs are also allowed for local access
  - Example: **`file:///home/gstein/repos/testing/`**

Details: checkout

- Creates a local working copy

```
$ svn checkout http://svn.example.com/repos/project/trunk
A  trunk/file1
A  trunk/file2
A  trunk/subdir/file3
A  trunk/subdir/file4
Checked out revision 5.
$ cd trunk
$ ls -aF
./  ../  .svn/  file1  file2  subdir/
$
```

Details: commit

- Commit changes to the repository

```
$ jed file1
$ svn commit –m "changed file1"
Sending        file1
Transmitting file data .
Committed revision 6.
$
```

Details: add

- Add new files and directories

```
$ touch file5
$ mkdir subdir2
$ svn add file5 subdir2
A         file5
A         subdir2
$ svn commit –m "added items"
Adding          file5
Adding          subdir2
Transmitting file data .
Committed revision 7.
$
```

Details: mkdir

- Simplify directory creation

```
$ svn mkdir subdir3
A         subdir3
$ svn commit –m "added subdir3"
Adding          subdir3

Committed revision 8.
$
```

Details: mkdir <URL>

- Quickly sets up a new repository directory

```
$ svn mkdir http://svn.example.com/repos/project/branches \
    -m "create branches area"

Committed revision 9.
$
```

Details: delete

- Delete files and directories

```
$ svn delete file5 subdir3
D file5
D subdir3
$ svn commit –m "deleted items"
Deleting       file5
Deleting       subdir3

Committed revision 10.
$
```

Details: delete <URL>

- Delete items directly from the repository
  - Great for removing obsolete tags or branches

```
$ svn delete \
http://svn.example.com/repos/project/branches/issue-10 \
    -m "delete unused branch"

Committed revision 11.
$
```

Details: update

- Retrieve changes made by other users

```
$ svn update
U  ./file2
A  ./newfile
Updated to revision 12.
$
```

The above example assumes that another user has created revisions 11 and 12. We update the working copy from revision 10 to 12.

Details: status

- Shows changes to the working copy

```
$ svn status
M      ./file2
M      ./moved-dir/file3
$ svn status –u
M       *       12   ./file2
M               12   ./moved-dir/file3
Head revision:     13
$
```

Details: copy

- Copy files and directories
  - Source and destination can be working copies and/or direct repository references

```
$ svn copy file1 file6
$ svn commit –m "made a copy"
Adding       file6

Committed revision 14.
$
```

Subversion remembers that **`file6`** came from **`file1`**.

Details: move

- Move files and directories
  - The source and destination must both be working copy references, or they must both be URLs

```
$ svn move subdir moved-dir
A         moved-dir
D  subdir/file3
D  subdir/file4
D  subdir
$ svn commit –m "moved a dir"
Adding       moved-dir
Deleting     subdir

Committed revision 15.
$
```

Subversion remembers that **`moved-dir`** came from **`subdir`**.

Details: diff

- Shows changes to the working copy

```
$ svn diff
Index: ./file2
===================================================================
--- ./file2
+++ ./file2     Tue Jul 11 17:41:15 2002
@@ -1,2 +1,3 @@
 foo
 bar
+baz
$
```

FAST! Subversion keeps an original copy, so it does not need to talk to the server to show the differences.

Details: log

- Shows changes that have been committed

```
$ svn log file1
------------------------------------------------------------------------
rev 2:  gstein | Tue, 12 Jul 2002 15:53:56 -0700 | 1 line

Changed file1
------------------------------------------------------------------------
rev 1:  gstein | Tue, 12 Jul 2002 13:30:03 -0700 | 1 line

Initial checkin
------------------------------------------------------------------------
$
```

Details: revert

- Reverts changes made to a working copy
  - Replaces CVS's idiom of "**`rm file; cvs update file`**"

```
$ svn status
M      ./file2
M      ./moved-dir/file3
$ svn revert --recursive .
Reverted ./file2
Reverted ./moved-dir/file3
$
```

"svn revert" requires an explicit target, and statement of recursive operation. This is for safety reasons, since changes are discarded.

Details: info

- Provide information about files / directories

```
$ svn info file2
Path: file2
Name: file2
Url: http://
http://svn.example.com/repos/project/trunk/file2
Revision: 16
Node Kind: file
Schedule: normal
Last Changed Author: gstein
Last Changed Rev: 13
Last Changed Date: Tue, 14 Jul 2002 08:37:02 -0700
Text Last Updated: Tue, 14 Jul 2002 08:49:02 -0700
Properties Last Updated: Tue, 16 Jul 2002 08:40:52 -0700
Checksum: 9Hx1YUCHqN2Ti6Ss/yUklA==
$
```

Details: properties

- Five different commands for manipulating properties on files and directories

```
$ svn propset test-property "hi there" file2
property `test-property' set on 'file2'
$ svn proplist file2
Properties on 'file2':
  test-property
$ svn propget test-property file2
hi there
$ svn propedit test-property file2
```

editor pops up here

```
Set new value for property `test-property' on `file2'
$ svn propget test-property file2
changed the property value
$ svn propdel test-property file2
property `test-property' deleted from 'file2'.
$
```

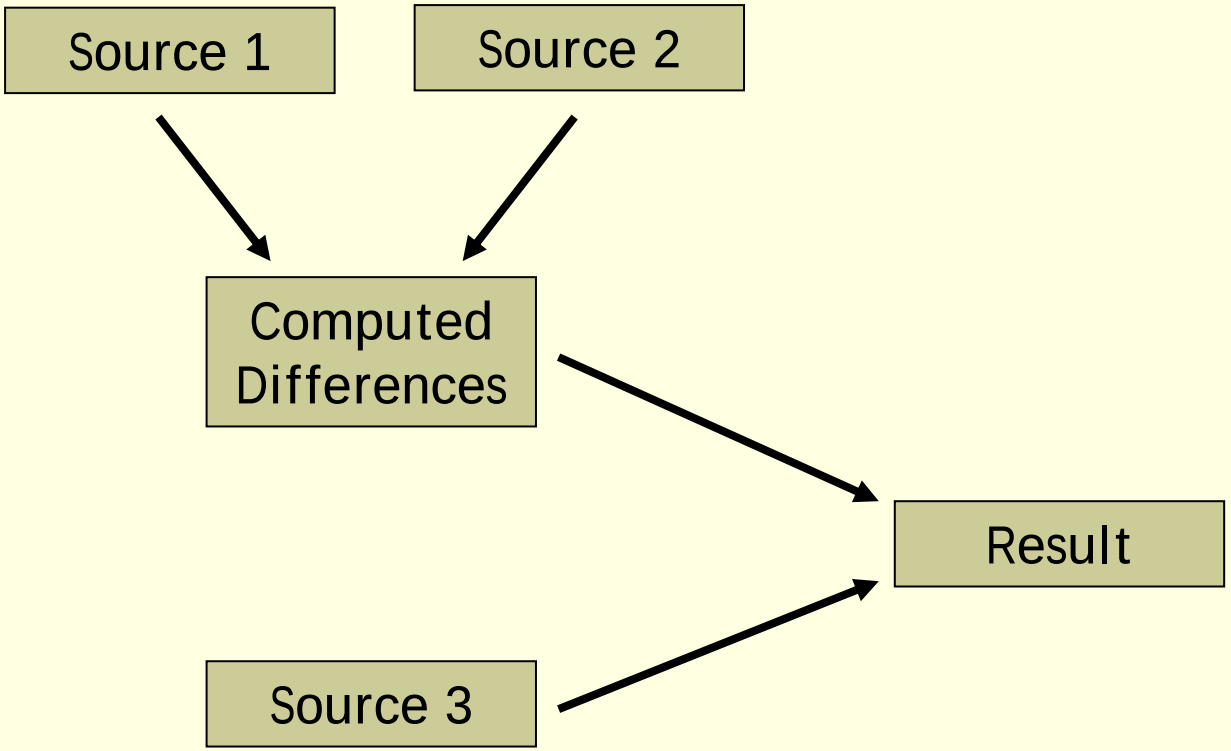
Details: merge

- Merges changes from two sources/revisions into a target

```
$ svn merge -r 15:16 file2 file6
U    file6
$
```

Merging is a huge topic for discussion. However, we can definitely say this is nicer than CVS's merging via "cvs update"

Three-way Merge



Details: resolve

- Cleans up conflict files left from a conflict during “svn update” or “svn merge”

```
$ ls file6*
file6
file6.63212.00001.working
file6.63216.00001.r16
file6.63220.00001.r15
$ svn resolve file6
Resolved conflicted state of file6
$ ls file6*
file6
$
```

Similar to CVS, Subversion inserts conflict markers into the conflicted source file (“file6” in this example).

Details: import

- Loads new content into a repository

```
$ svn import http://svn.example.com/repos/project/ \
    localdir trunk –m "initial import"
Adding         localdir/file10
Adding         localdir/file11
Transmitting file data ..
Committed revision 1.
$
```

Details: export

- Just like a checkout, but the **`.svn`** administrative subdirectories are omitted

```
$ svn export http://svn.example.com/repos/project/trunk
A  trunk/file11
A  trunk/file10
Checked out revision 1.
$ ls –aF trunk
./  ../  file10  file11
$
```

Keywords are expanded and newline translation is performed.

Details: switch

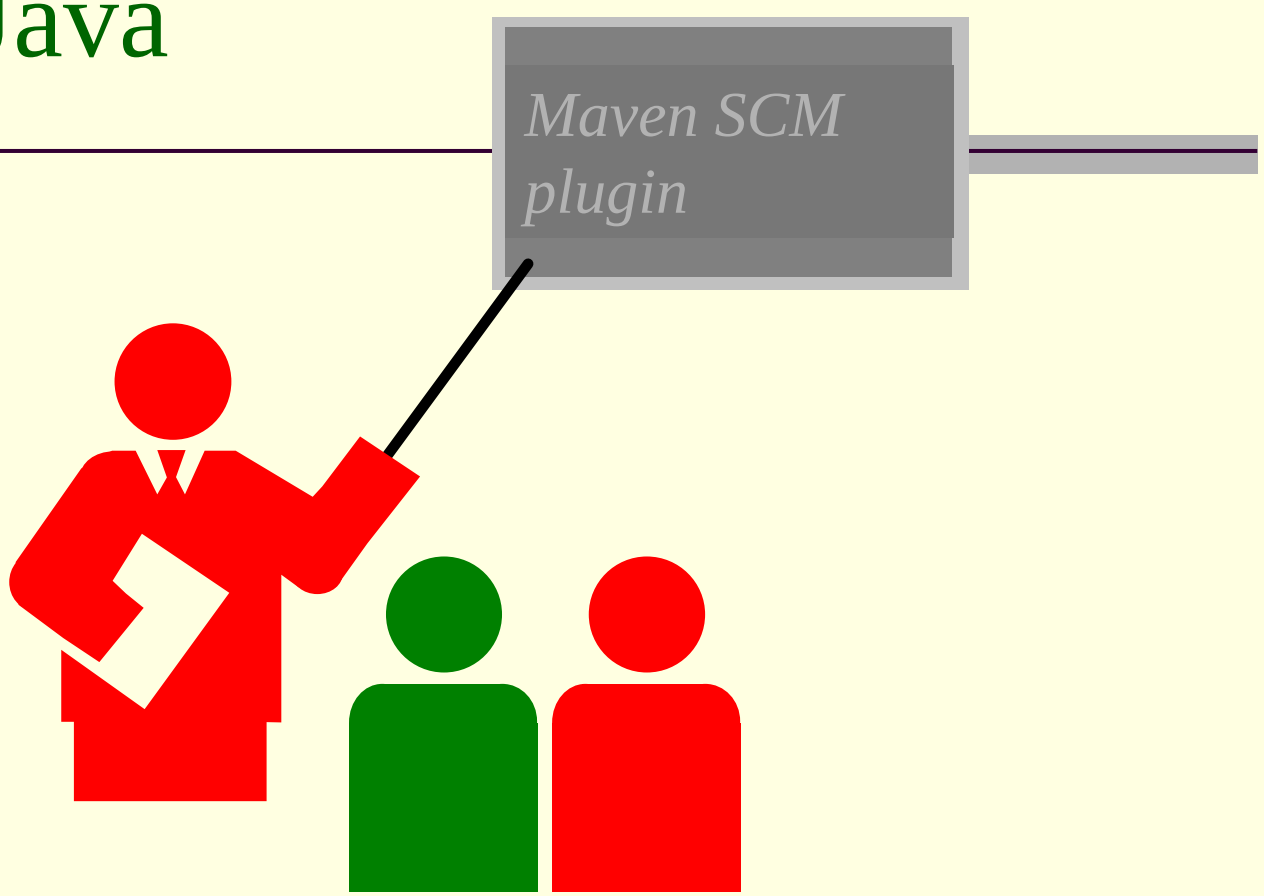
- Switch a working copy to a branch

```
$ svn info | grep Url:
Url: http://svn.example.com/repos/test/trunk
$ svn switch http://svn.example.com/repos/project/branches/issue-10
U  ./file2
Updated to revision 18.
$ svn info | grep Url:
Url: http://svn.example.com/repos/test/branches/issue-10
$
```

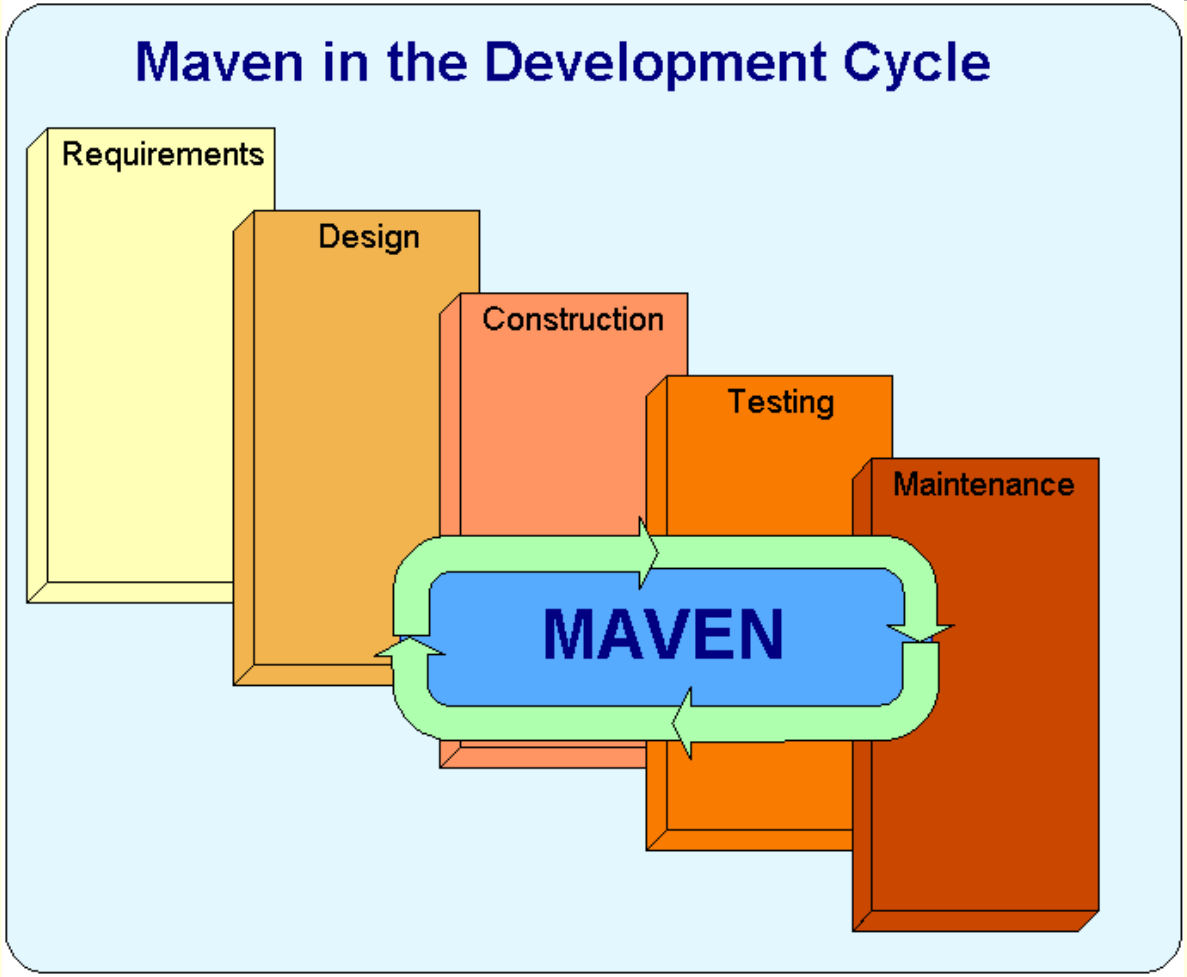
Additional Tools

- cvs2svn
- ViewSVN
- Hook scripts
  - Send commit emails
  - Simple ACL support
  - Simple reserved checkouts
  - Repository backup
- Libraries, scripting, svnlook

MTSW-Java



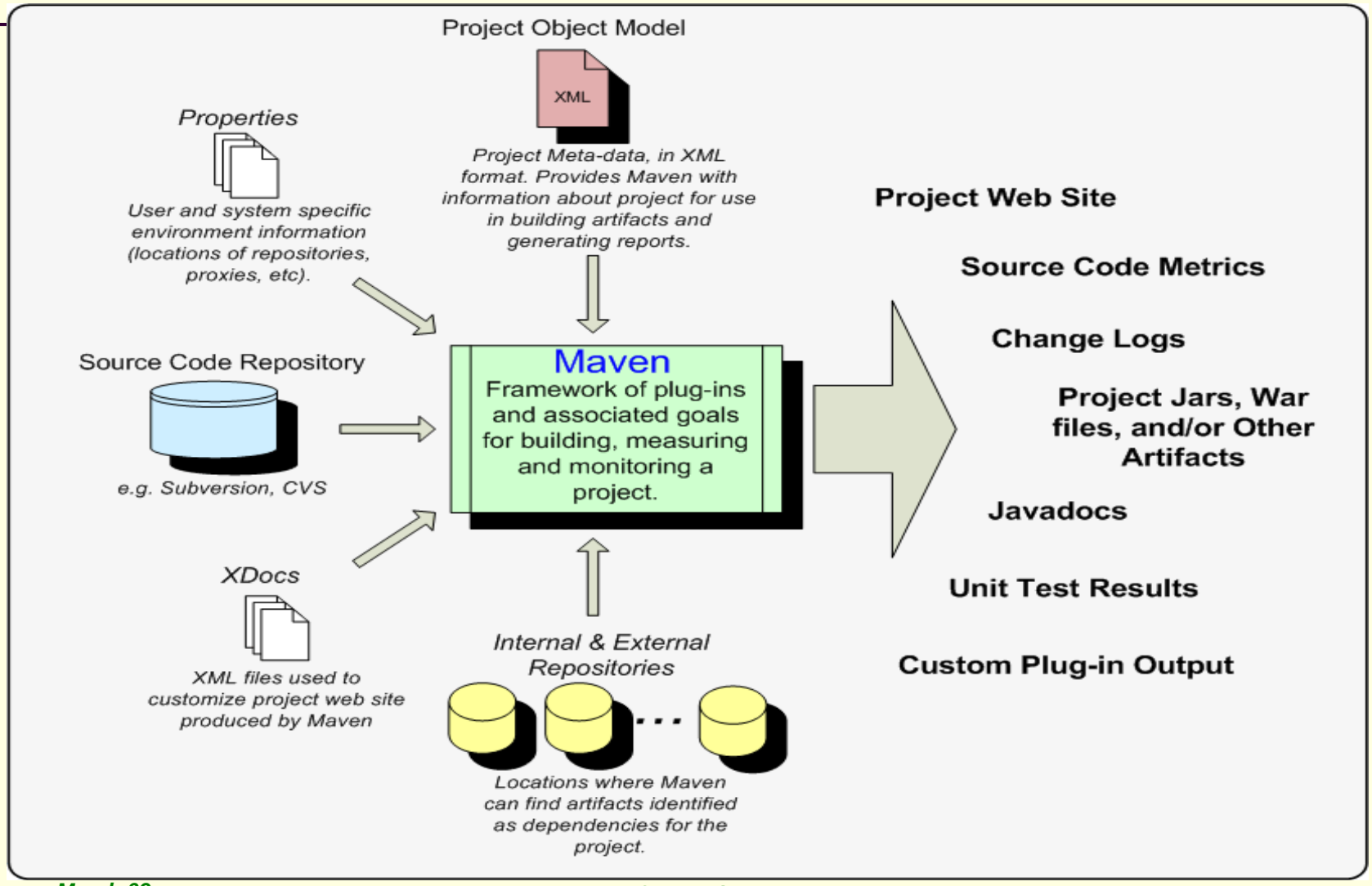
Maven in Development



Maven Features/Functions

- Manage External Dependencies
- Retrieve Source from SCM
- Build Artifacts
- Run Tests
- Package Artifacts (jar,war,zip,tar…)
- Generate Project Reports
- Create Website of Project Info
- Deploy Website and Distributions
- Extensible via Plugins

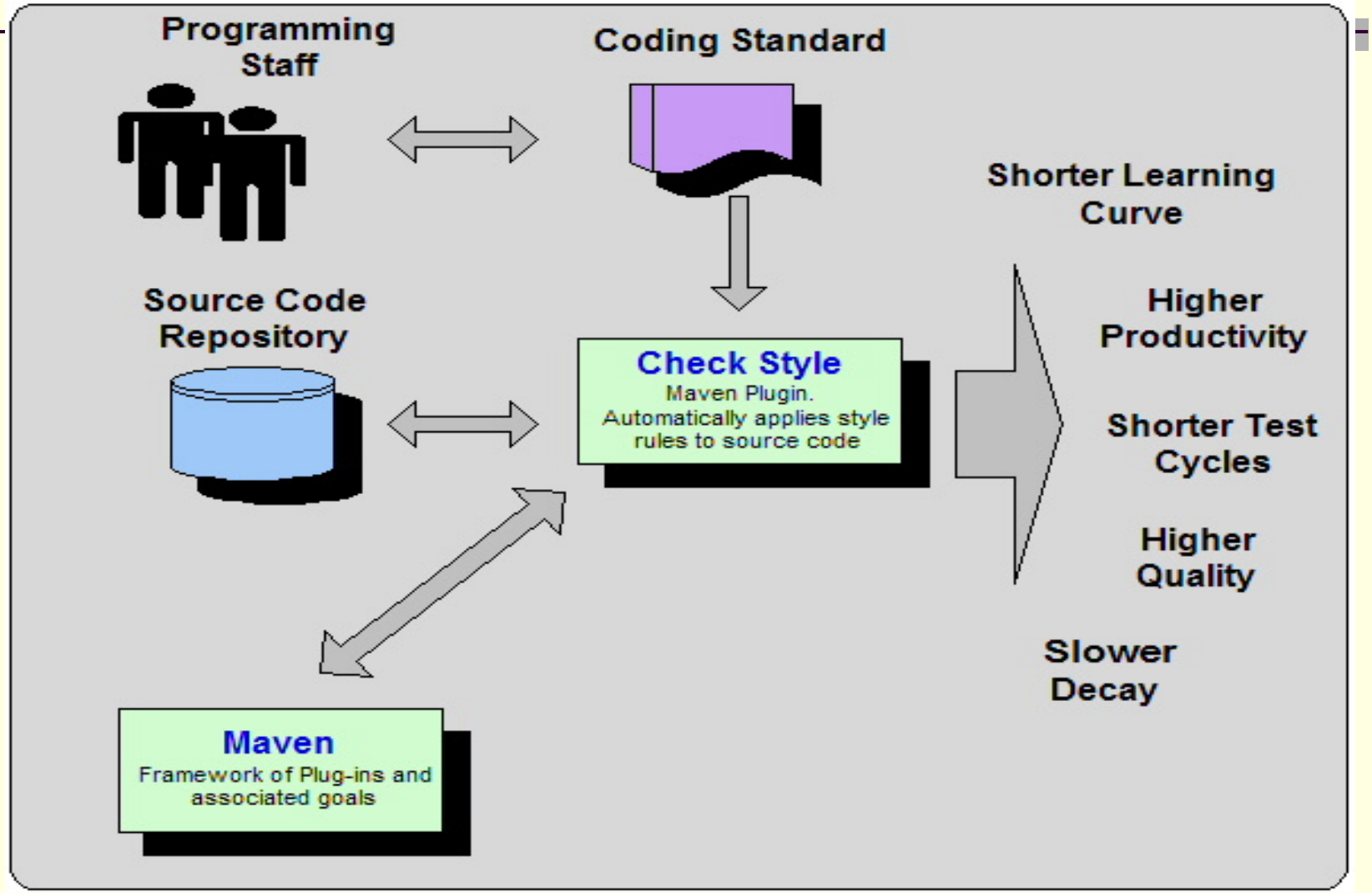
Maven Overview



Coding Standards & Maven

- Maven Provides a Plug-In Architecture for 3rd Party Applications
- Check Style Description
  - Development Tool for Java Programmers
  - Flexible Enough to Support Any Standard
  - Checks Coding Style and Coding Structure
  - Optional Checks Include J2EE Requirements

Coding Standards & Maven



SCM Plugin has 16 goals

- Seamlessly integrates Maven processes with your SCM repository.
  - mvn scm:diff
    - Creates UNIX diff file
  - mvn scm:tag
  - mvn scm:status
- Dozens of SCM systems supported.
  - CVS, Subversion, Git

Maven SCM Support

- Seamlessly integrates Maven processes with your SCM repository.
  - scm:branch - branch the project
  - scm:validate - validate the scm information in the pom
  - scm:add - command to add file
  - scm:unedit - command to stop editing the working copy
  - scm:export - command to get a fresh exported copy
  - scm:bootstrap - command to checkout and build a project
  - scm:changelog - command to show the source code revisions
  - scm:list - command for get the list of project files

SCM Plugin

- The scm plugin maps a lot of commands to a variety of scm implementations
  - checkin - commit the changes to the remote repository (scm server).
  - update - updates the local working copy with the one from the remote repository (scm server).

SCM Plugin

- Configuration

```
...
<scm>
<connection>
scm:svn:http://somerepository.com/svn_repo/trunk
</connection>
<developerConnection>
  scm:svn:https://somerepository.com/svn_repo/trunk
</developerConnection>
<url>http://somerepository.com/view.cvs</url>
</scm>
...
</project>
```

- http://maven.apache.org/scm/maven-scm-plugin/usage.html

SCM Plugin

- Configuration

```
...
<scm>
<connection>
scm:svn:http://somerepository.com/svn_repo/trunk
</connection>
<developerConnection>
  scm:svn:https://somerepository.com/svn_repo/trunk
</developerConnection>
<url>http://somerepository.com/view.cvs</url>
</scm>
...
</project>
```

- http://maven.apache.org/scm/maven-scm-plugin/usage.html

SCM Plugin

- Bootstrapping a Project Using a POM
  - http://maven.apache.org/scm/maven-scm-plugin/examples/bootstrapping-with-pom.html
- Other SCM Commands
  - http://maven.apache.org/scm/maven-scm-plugin/examples/scm-advance-features.html