CS520 Web Programming
Version Control with CVS and Subversion

Chengyu Sun
California State University, Los Angeles

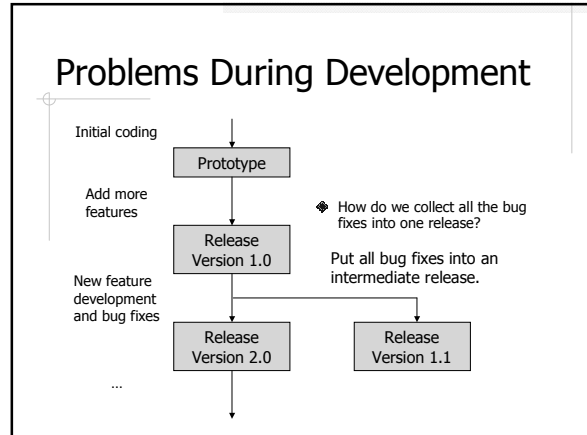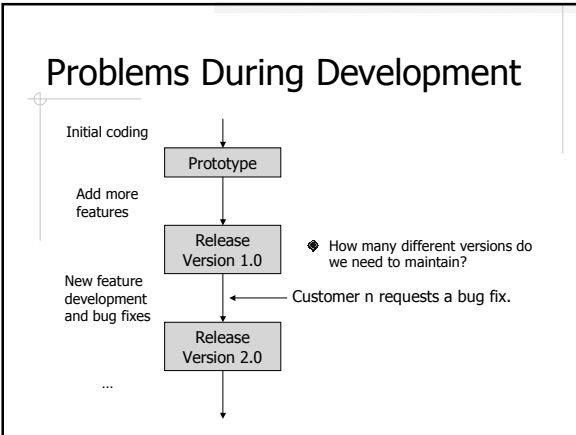# Overview

◆ Version control systems
◆ Basic concepts
  - Repository and working copies
  - Tag, branch, and merge
◆ Using Subversion

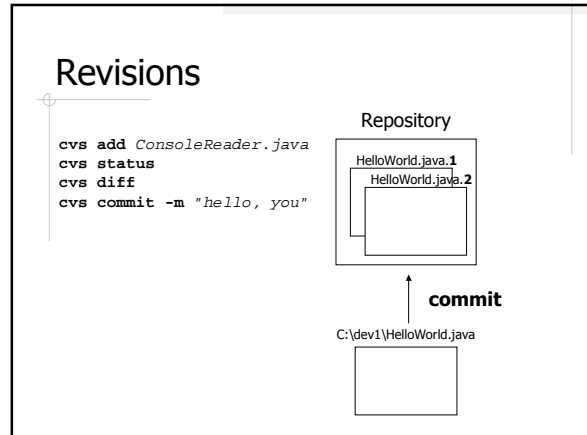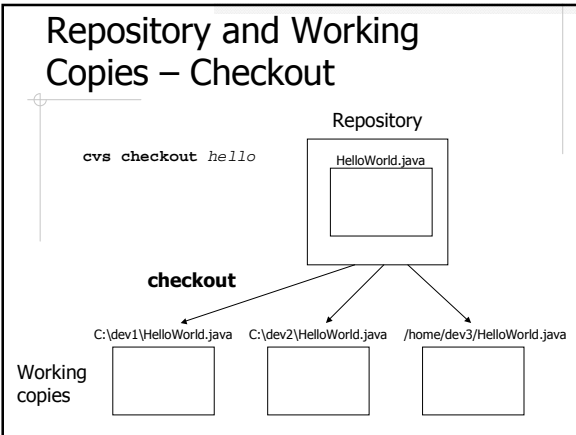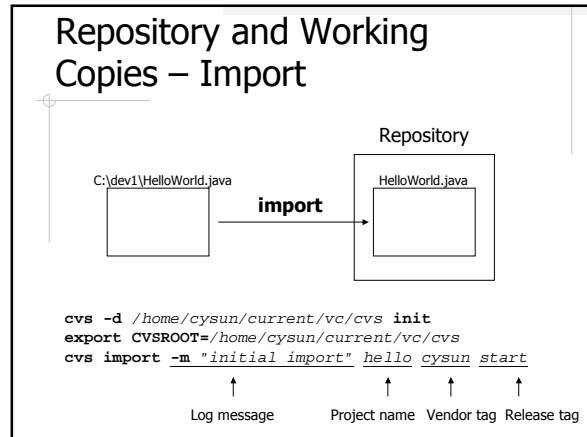# The Development of the HelloWorld Application

Initial coding

Prototype — (hello world!)

Add more features — (adding `ConsoleReader`)

Release Version 1.0

New feature development and bug fixes — (using `Scanner`)

Release Version 2.0

…

Problems During Development

Initial coding

Prototype

Add more features — New feature has broken existing code.

Release Version 1.0

◆ How do we find out which part of the code has been changed?
◆ How do we revert back to the previous version?

New feature development and bug fixes

Release Version 2.0

…

Problems During Development

Initial coding

Prototype

Add more features

Release Version 1.0

◆ Can we give the customer the development version with the bug fix?
◆ Do we still have the Version 1.0 code?

New feature development and bug fixes — Customer 1 requests a bug fix.

(HelloWorld => Hello)

Release Version 2.0

…

Problems During Development

Initial coding

Prototype

Add more features

Release Version 1.0

◆ Where do we put in the bug fix?
  - Version 1.0
  - Version 1.0 + customer 1's bug fix
  - Development version

New feature development and bug fixes — Customer 2 requests a bug fix.

(hello =>hola)

Release Version 2.0

…

Problems During Development

Initial coding

Add more features

New feature development and bug fixes

...

Prototype → Release Version 1.0 → Release Version 2.0

Customer n requests a bug fix.

◈ How many different versions do we need to maintain?

Problems During Development

Initial coding

Add more features

New feature development and bug fixes

...

Prototype → Release Version 1.0 → Release Version 2.0 / Release Version 1.1

◈ How do we collect all the bug fixes into one release?

Put all bug fixes into an intermediate release.
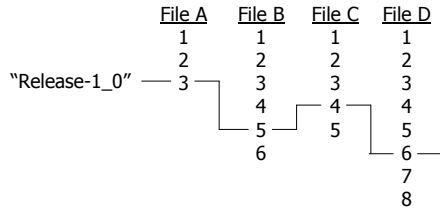
Version Control Systems

◈ **CVS**
  - Most popular / well known / widely used open source version control system
  - Somewhat obsolete due to some inherent system limitations

◈ **Subversion**, Arch, Monotone

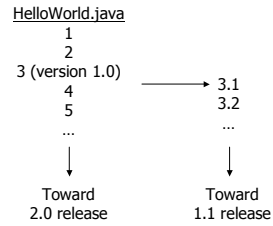◈ Commercial
  - Visual SourceSafe
  - ClearCase
  - BitKeeper

Repository and Working Copies – Import

Repository

C:\dev1\HelloWorld.java → **import** → HelloWorld.java

```
cvs -d /home/cysun/current/vc/cvs init
export CVSROOT=/home/cysun/current/vc/cvs
cvs import -m "initial import" hello cysun start
```

Log message   Project name   Vendor tag   Release tag

Repository and Working Copies – Checkout

```
cvs checkout hello
```

Repository

HelloWorld.java

checkout

C:\dev1\HelloWorld.java   C:\dev2\HelloWorld.java   /home/dev3/HelloWorld.java

Working copies

Revisions

```
cvs add ConsoleReader.java
cvs status
cvs diff
cvs commit -m "hello, you"
```

Repository

HelloWorld.java.**1**
HelloWorld.java.**2**

commit

C:\dev1\HelloWorld.java

Tag – Mark A Moment in Time

File A	File B	File C	File D
1	1	1	1
2	2	2	2
"Release-1_0" — 3	3	3	3
	4	4	4
	5	5	5
	6		6
			7
			8

```
cvs tag "Release-1_0"
cvs checkout -r "Release-1_0" -d hello-1.0 hello
```

Branch – Work in Parallel

HelloWorld.java
```
1
2
3 (version 1.0)  ──────►  3.1
4                         3.2
5                         ...
...
        │                     │
        ▼                     ▼
     Toward                Toward
   2.0 release           1.1 release
```

```
cvs tag -b "1_0-bugfix1"
cvs checkout -r "1_0-bugfix1" -d hello-1.0-bugfix1 hello
```

Merge

◆ Copy changes between different branches

HelloWorld.java
```
1
2
3 (version 1.0)  ───►  3.1
4                      3.2
5                      3.3
6                      3.4
...                    ...
     │  ◄─── 3.10 (version 1.1)
     ▼
   Toward
 2.0 release
```
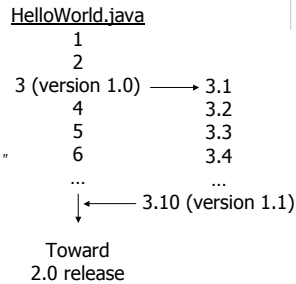
```
cvs update -j "1_0-bugfix1"
```

Subversion

◆ Pros: A better CVS
  - Fixed many annoying aspects of CVS
    - Recursive add, binary file handling, keyword substitution, local diff, status output etc.
  - Significant improvements
    - Atomic commit, constant time branching and tagging, better structure design etc.
  - *Feels like CVS*
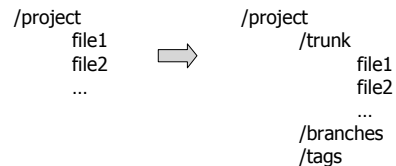◆ Cons: Just a better CVS
  - Does not scale to large, distributed development environments

---

Project Directory Structure

◆ Branching and tagging in Subversion are through directory copying.
◆ Consequently, a project usually has the following directory <u>in the repository</u>:

```
/project
    /trunk
    /branches
    /tags
```

Working with the Directory Structure

```
/project                    /project
    file1                       /trunk
    file2        ⇨                  file1
    ...                             file2
                                    ...
                                /branches
                                /tags
```

Common Command Syntax

```
svn <command> [src_dir] [dest_dir]
```

Could be local directories or URLs.

Tagging

◆ Copy `/trunk` to a directory under `/tags`

Branching

◆ Copy `/trunk` to a directory under `/branches`

Merging

◆ Apply the difference between two versions to the *local working copy*.

Other Useful Subversion Commands

◆ Add
◆ Remove
◆ Update
◆ Revert
◆ Commit

◆ Log
◆ Status

Online Resources

◆ The CVS book - http://cvsbook.red-bean.com/
◆ The Subversion book - http://svnbook.red-bean.com/
◆ Subversion official site - http://subversion.tigris.org/
◆ Subversion Eclipse plugin – http://subclipse.tigris.org/