

HUST

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

SOICT

School of Information and Communication Technology

ONE LOVE. ONE FUTURE.



IT3180 – Introduction to Software Engineering

17 – Configuration Management

ONE LOVE. ONE FUTURE.

Content

1. Changes and Software Configuration
2. Software Configuration Management
3. SCM Process
4. Version Control

Changes and Software Configuration

The “First Law”

No matter where you are in the system life cycle, the system will change, and the desire to change it will persist throughout the life cycle.

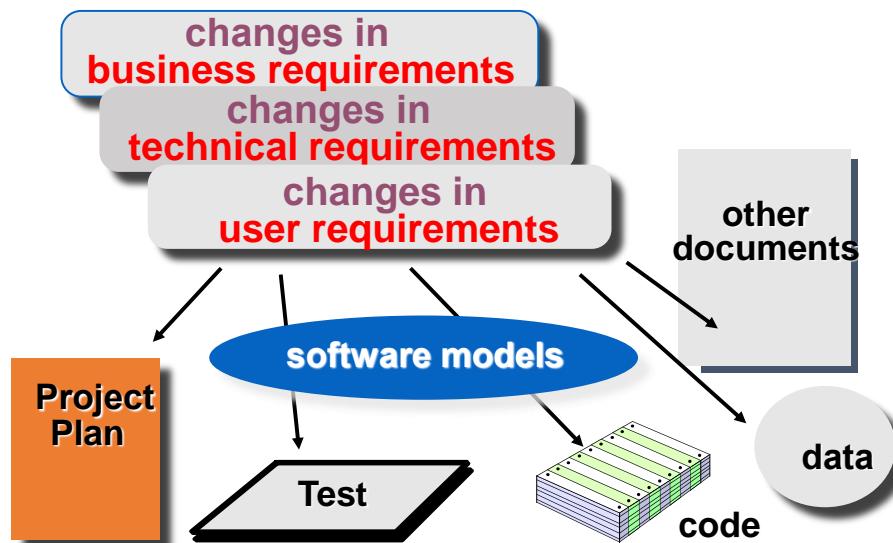
Bersoff, et al, 1980



Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

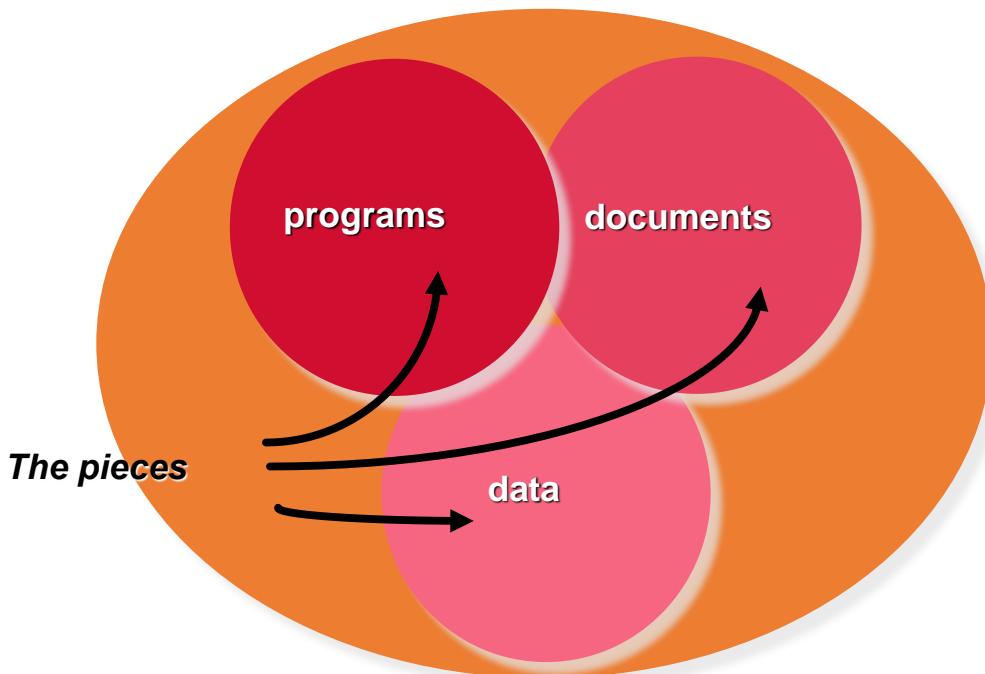
Changes and Software Configuration

What Are These Changes?



Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

The Software Configuration



Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

Changes and Software Configuration

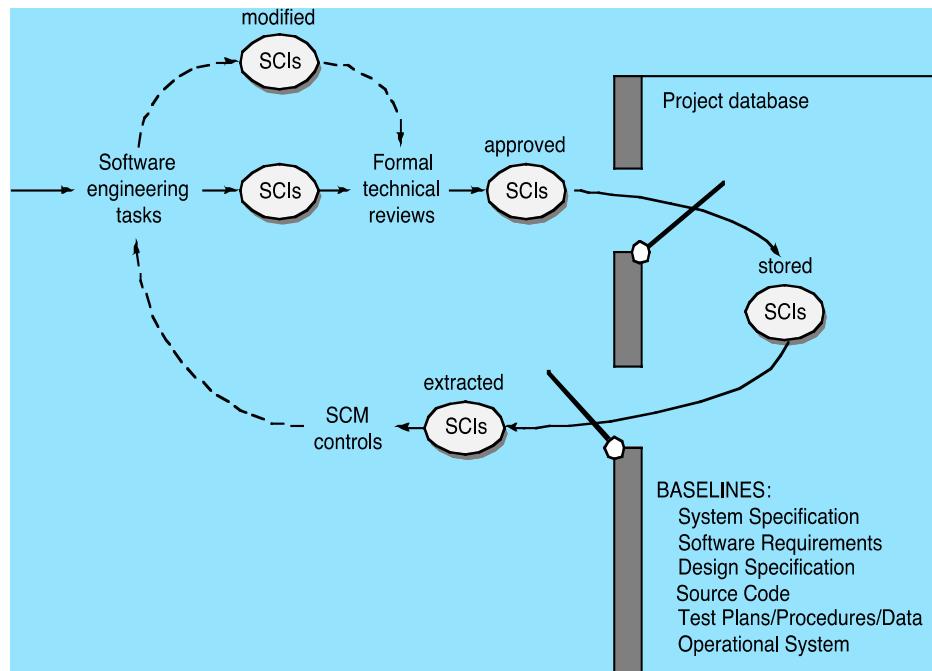
- The IEEE (IEEE Std. No. 610.12-1990) defines a baseline as:
A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures.
- a baseline is a milestone in the development of software that is marked by the delivery of one or more software configuration items and the approval of these SCIs that is obtained through a formal technical review



Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

Changes and Software Configuration

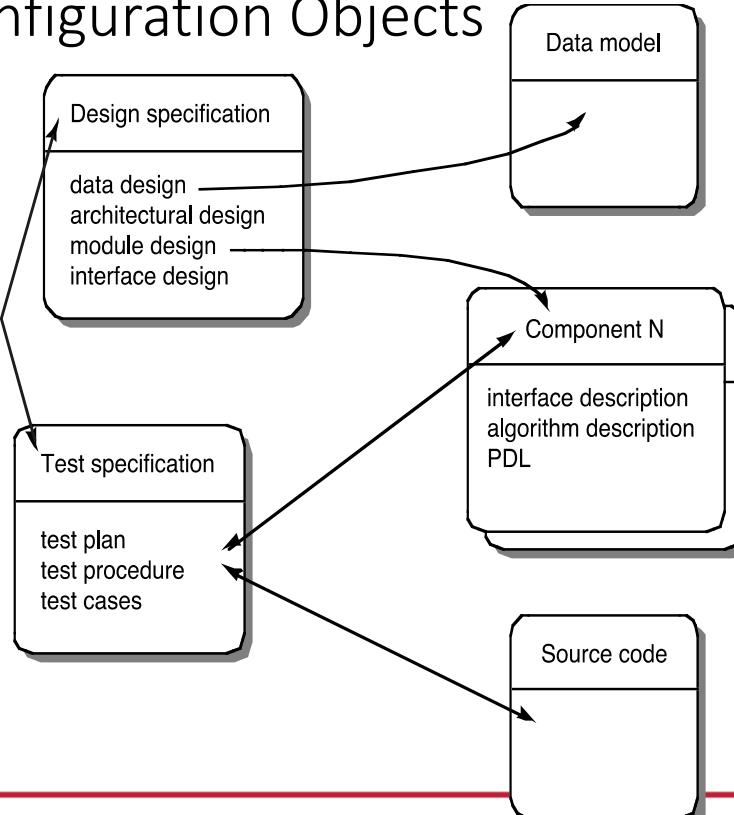
Baselines



Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

Changes and Software Configuration

Software Configuration Objects



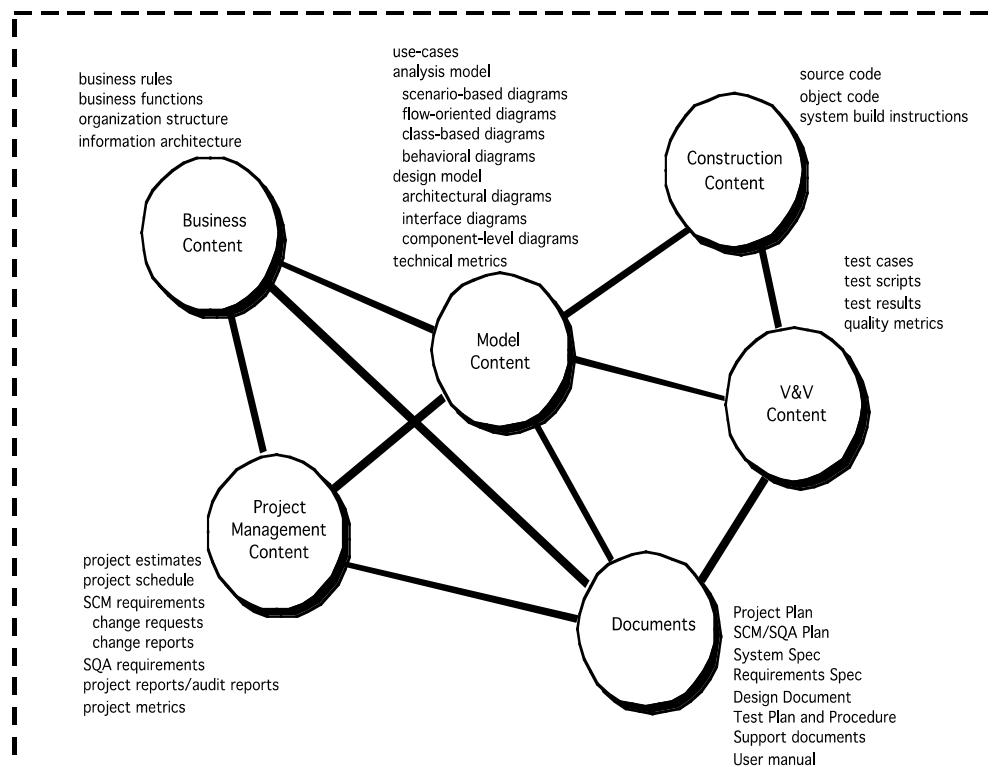
Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

SCM Repository

- The SCM repository is the set of mechanisms and data structures that allow a software team to manage change in an effective manner
- The repository performs or precipitates the following functions [For89]:
 - Data integrity
 - Information sharing
 - Tool integration
 - Data integration
 - Methodology enforcement
 - Document standardization



Repository Content



Repository Features

- **Versioning.**
 - saves all of these versions to enable effective management of product releases and to permit developers to go back to previous versions
- **Dependency tracking and change management.**
 - The repository manages a wide variety of relationships among the data elements stored in it.
- **Requirements tracing.**
 - Provides the ability to track all the design and construction components and deliverables that result from a specific requirement specification
- **Configuration management.**
 - Keeps track of a series of configurations representing specific project milestones or production releases. Version management provides the needed versions, and link management keeps track of interdependencies.
- **Audit trails.**
 - establishes additional information about when, why, and by whom changes are made.



Software Configuration Management

SCM Elements

Component elements—a set of tools coupled within a file management system (e.g., a database) that enables access to and management of each software configuration item.

Process elements—a collection of procedures and tasks that define an effective approach to change management (and related activities) for all constituencies involved in the management, engineering and use of computer software.

Construction elements—a set of tools that automate the construction of software by ensuring that the proper set of validated components (i.e., the correct version) have been assembled.

Human elements—to implement effective SCM, the software team uses a set of tools and process features (encompassing other CM elements)



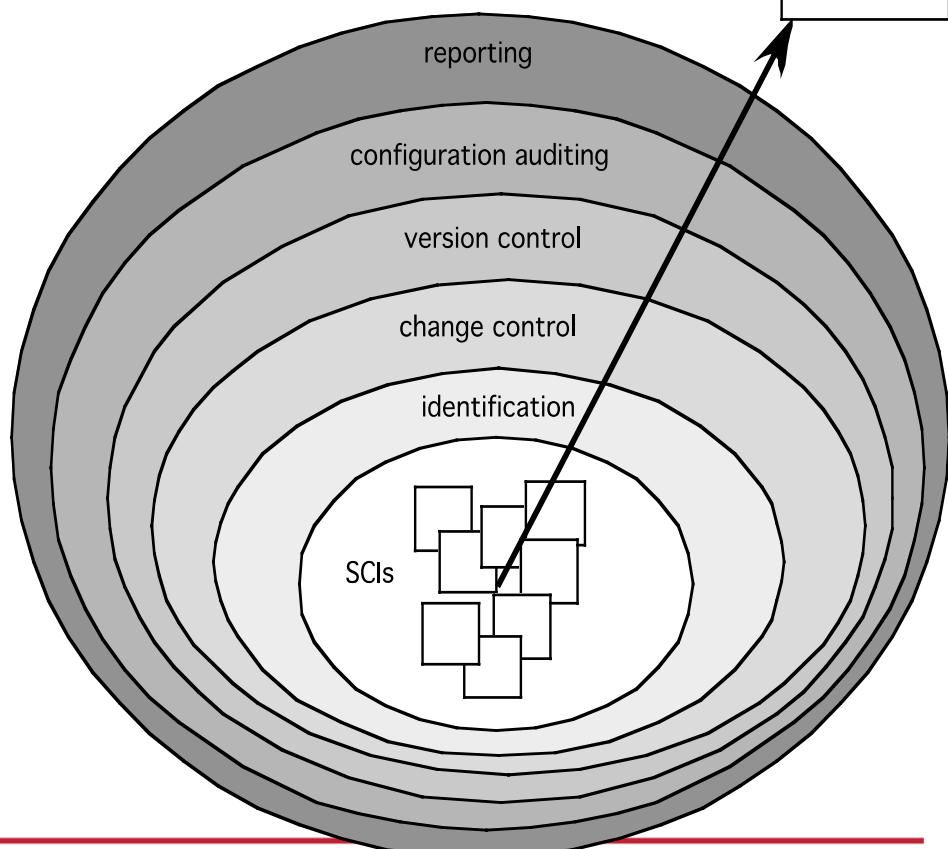
Addresses the following questions ...

- How does a software team identify the discrete elements of a software configuration?
- How does an organization manage the many existing versions of a program (and its documentation) in a manner that will enable change to be accommodated efficiently?
- How does an organization control changes before and after software is released to a customer?
- Who has responsibility for approving and ranking changes?
- How can we ensure that changes have been made properly?
- What mechanism is used to appraise others of changes that are made?



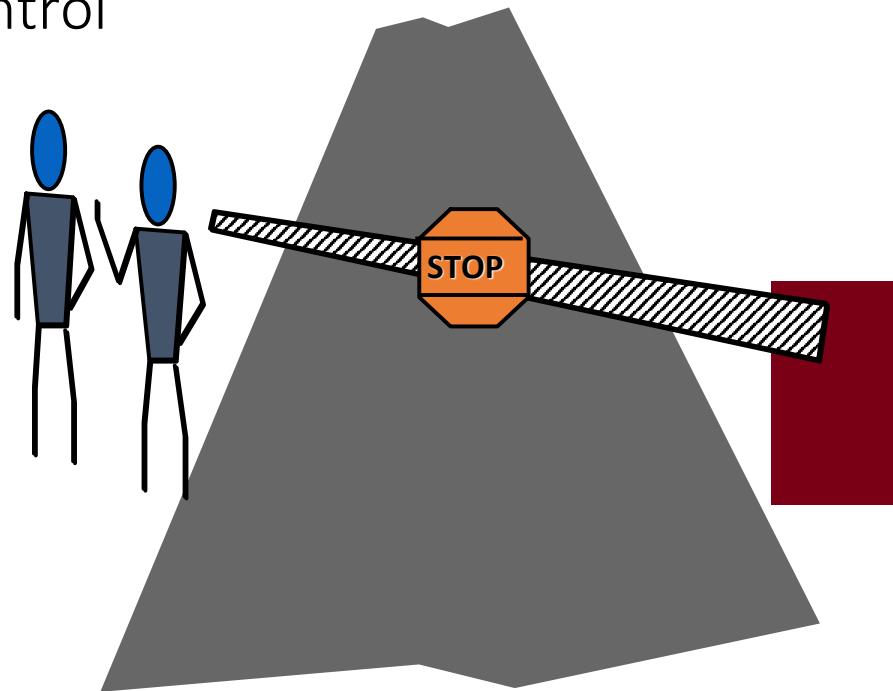
Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

16

Software
Vm.n

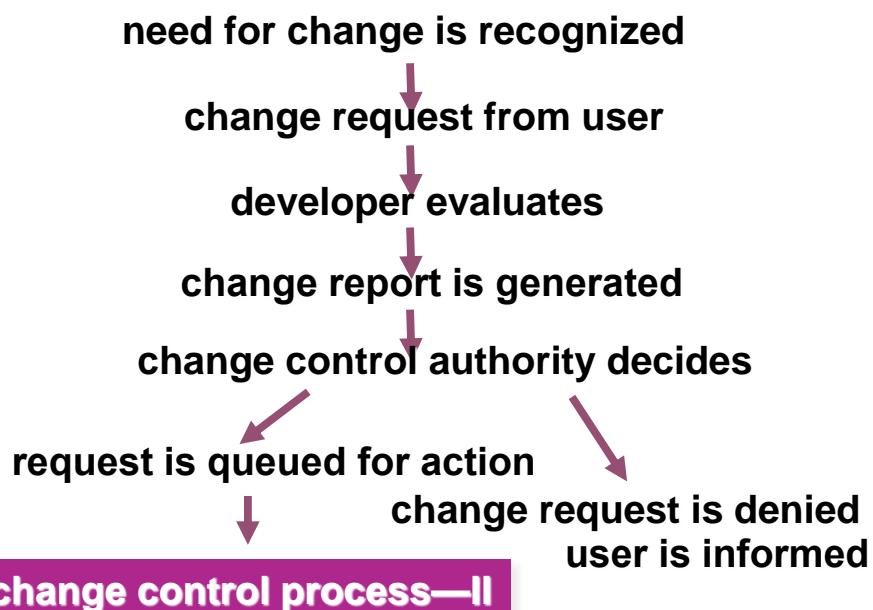
Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

Change Control



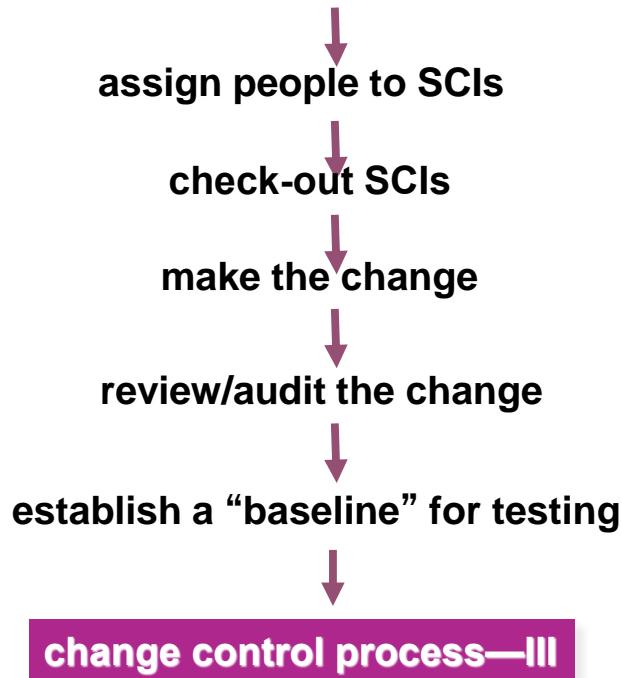
Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

Change Control Process—I



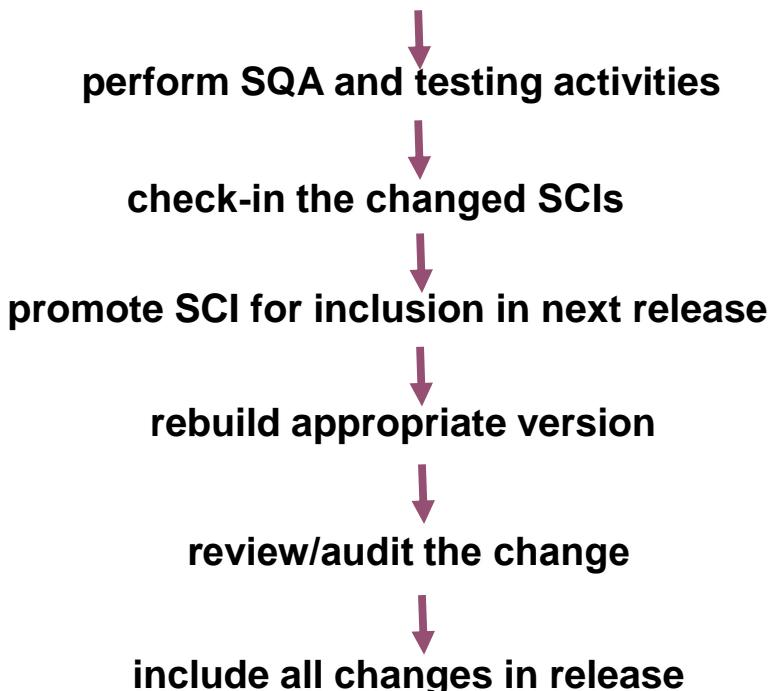
Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

Change Control Process-II



Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

Change Control Process-III



Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

Version Control

- Version control combines procedures and tools to manage different versions of configuration objects that are created during the software process
- A version control system implements or is directly integrated with four major capabilities:
 - a *project database (repository)* that stores all relevant configuration objects
 - a *version management* capability that stores all versions of a configuration object (or enables any version to be constructed using differences from past versions);
 - a *make facility* that enables the software engineer to collect all relevant configuration objects and construct a specific version of the software.
 - an *issues tracking* (also called *bug tracking*) capability that enables the team to record and track the status of all outstanding issues associated with each configuration object.

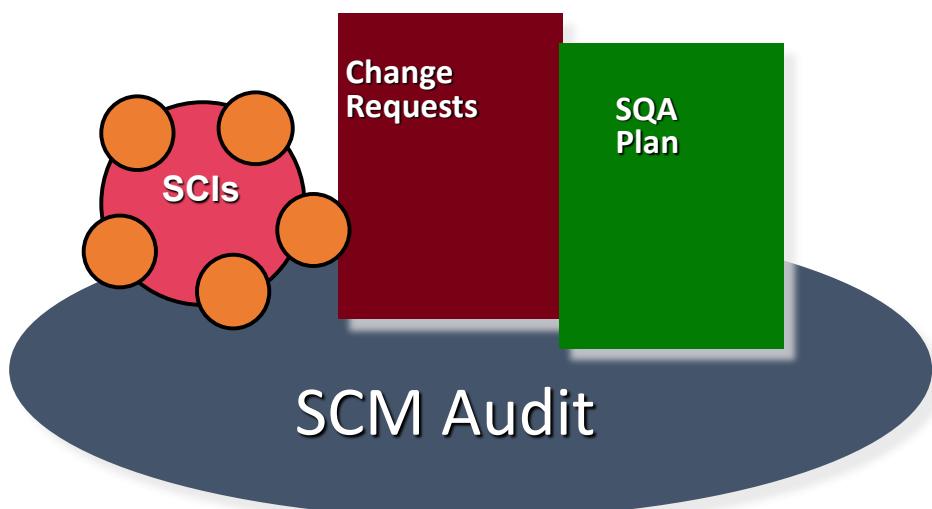


Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

22

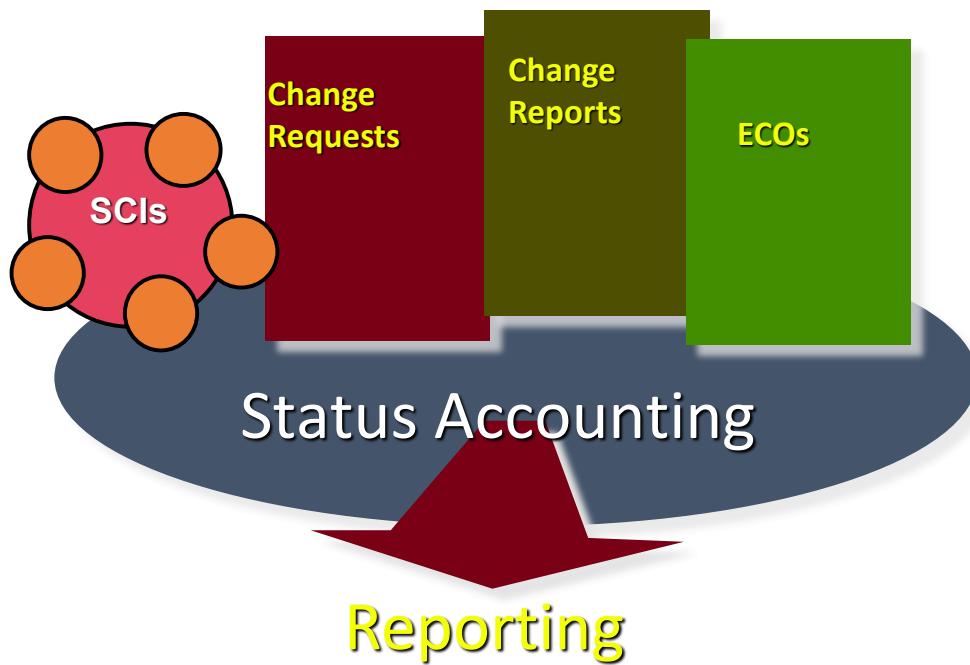
SCM Process

Configuration Auditing



Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

Status Accounting



Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

SCM for Web Engineering-I

- **Content**

- A typical WebApp contains a vast array of content—text, graphics, applets, scripts, audio/video files, forms, active page elements, tables, streaming data, and many others.
- The challenge is to organize this sea of content into a rational set of configuration objects (Section 27.1.4) and then establish appropriate configuration control mechanisms for these objects.

- **People**

- Because a significant percentage of WebApp development continues to be conducted in an ad hoc manner, any person involved in the WebApp can (and often does) create content.



Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

SCM for Web Engineering-II

- **Content**

- As size and complexity grow, small changes can have far-reaching and unintended affects that can be problematic. Therefore, the rigor of configuration control mechanisms should be directly proportional to application scale.

- **Politics**

- Who ‘owns’ a WebApp?
- Who assumes responsibility for the accuracy of the information on the Web site?
- Who assures that quality control processes have been followed before information is published to the site?
- Who is responsible for making changes?
- Who assumes the cost of change?



Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

Content Management-I

- **The collection subsystem** encompasses all actions required to create and/or acquire content, and the technical functions that are necessary to
 - convert content into a form that can be represented by a mark-up language (e.g., HTML, XML)
 - organize content into packets that can be displayed effectively on the client-side.
- **The management subsystem** implements a repository that encompasses the following elements:
 - **Content database**—the information structure that has been established to store all content objects
 - **Database capabilities**—functions that enable the CMS to search for specific content objects (or categories of objects), store and retrieve objects, and manage the file structure that has been established for the content
 - **Configuration management functions**—the functional elements and associated workflow that support content object identification, version control, change management, change auditing, and reporting.



Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

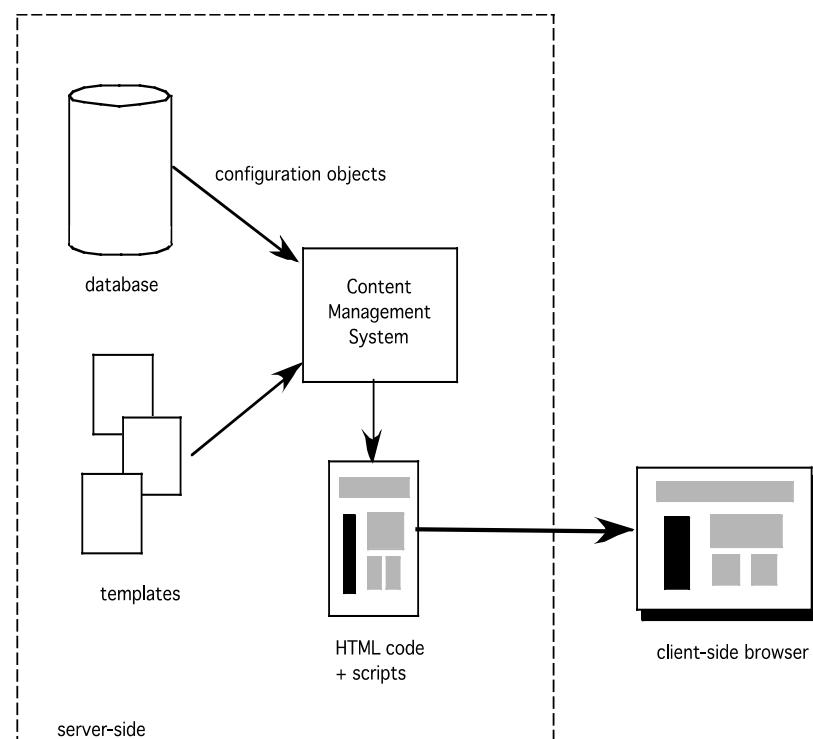
Content Management-II

- The **publishing subsystem** extracts from the repository, converts it to a form that is amenable to publication, and formats it so that it can be transmitted to client-side browsers. The publishing subsystem accomplishes these tasks using a series of templates.
- Each *template* is a function that builds a publication using one of three different components [BOI02]:
 - *Static elements*—text, graphics, media, and scripts that require no further processing are transmitted directly to the client-side
 - *Publication services*—function calls to specific retrieval and formatting services that personalize content (using predefined rules), perform data conversion, and build appropriate navigation links.
 - *External services*—provide access to external corporate information infrastructure such as enterprise data or “back-room” applications.



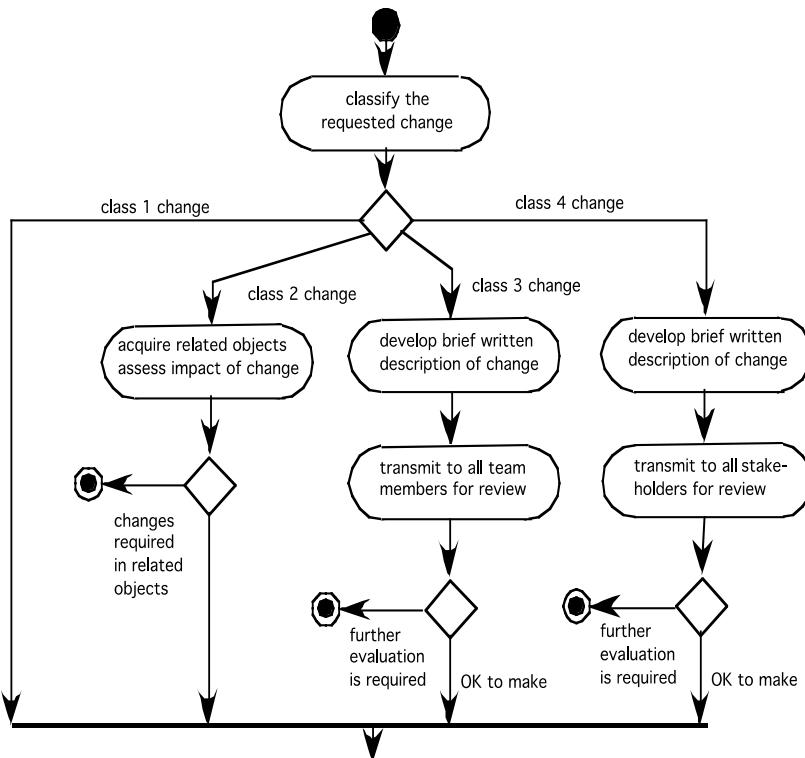
Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

Content Management



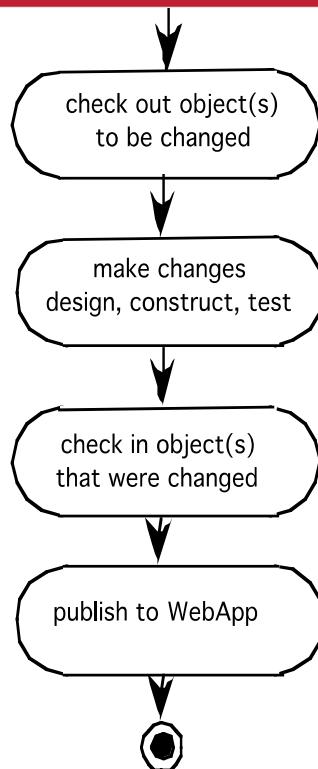
Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

Change Management for WebApps-I



Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

30 Change Management for WebApps-II



Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

4. Version Control



4.1. Why version control?

- Scenario 1:
 - Your program is working
 - You change “just one thing”
 - Your program breaks
 - You change it back
 - Your program is still broken--*why?*
- Has this ever happened to you?



Why version control?

- Your program worked well enough yesterday
- You made a lot of improvements last night...
 - ...but you haven't gotten them to work yet
- You need to turn in your program *now*
- Has this ever happened to you?



Version control for teams

- Scenario:
 - You change one part of a program--it works
 - Your co-worker changes another part--it works
 - You put them together--it doesn't work
 - Some change in one part must have broken something in the other part
 - What were all the changes?



- Scenario:
 - You make a number of improvements to a class
 - Your co-worker makes a number of different improvements to the same class
- How can you merge these changes?



Tools: diff

- There are a number of tools that help you spot changes (differences) between two files
- Tools include **diff**, **rcsdiff**, **jDiff**, etc.
- Of course, they won't help unless you kept a copy of the older version
- Differencing tools are useful for finding a *small* number of differences in a *few* files

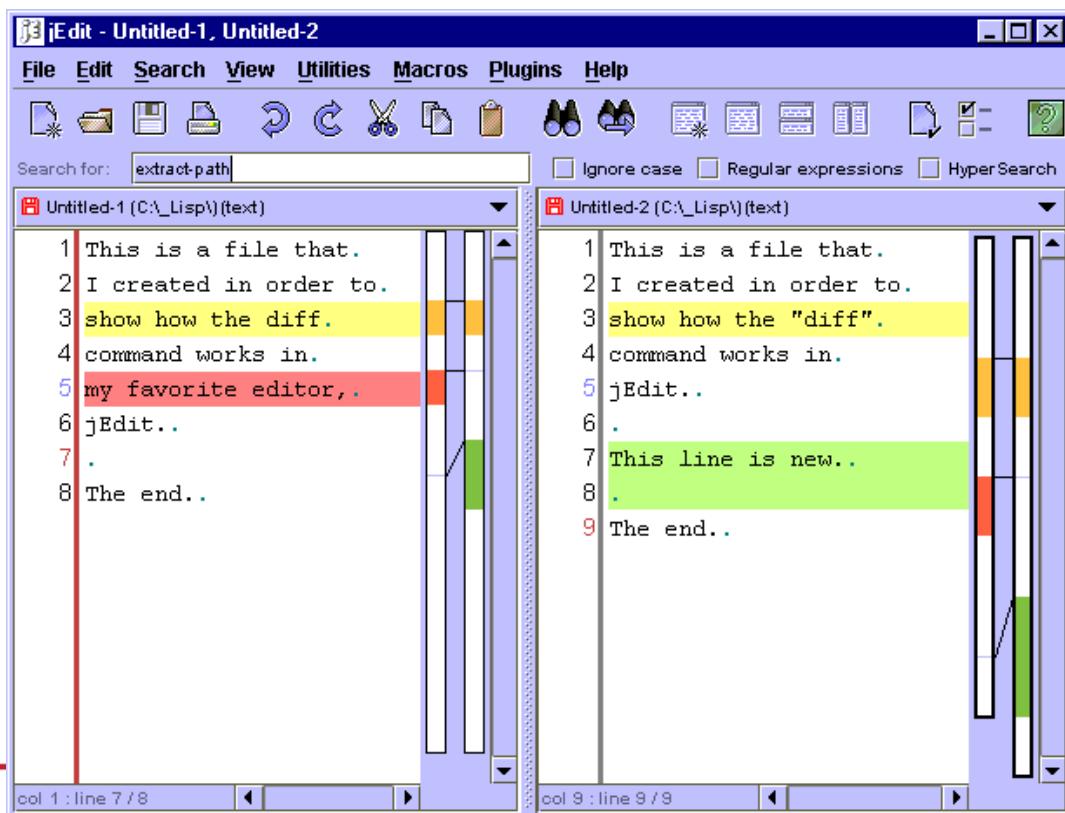


Tools: jDiff

- jDiff is a plugin for the jEdit editor
- Advantages:
 - Everything is color coded
 - Uses synchronized scrolling
 - It's inside an editor--you can make changes directly
- Disadvantages:
 - Not stand-alone, but must be used within jDiff
 - Just a diff tool, not a complete solution



Tools: jDiff



Version control systems

- A version control system (often called a source code control system) does these things:
 - Keeps multiple (older and newer) versions of everything (not just source code)
 - Requests comments regarding every change
 - Allows “check in” and “check out” of files so you know which files someone else is working on
 - Displays differences between versions



4.2. Versioning Models

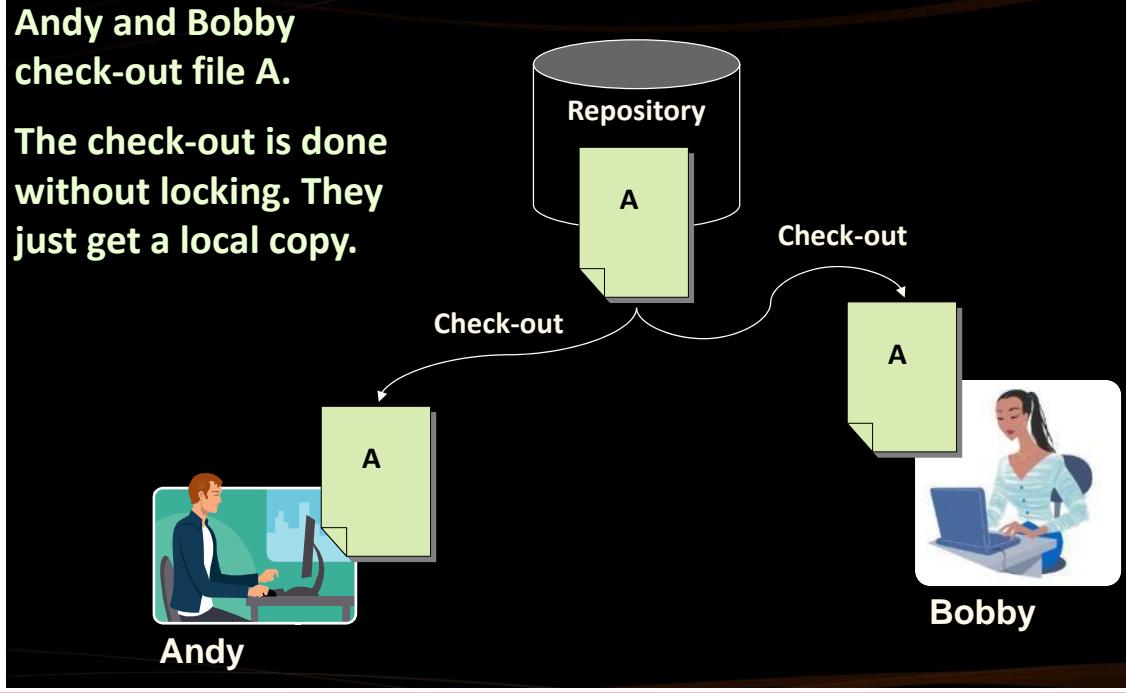
- Lock-Modify-Unlock
- Copy-Modify-Merge



a. The Lock-Modify-Unlock Model

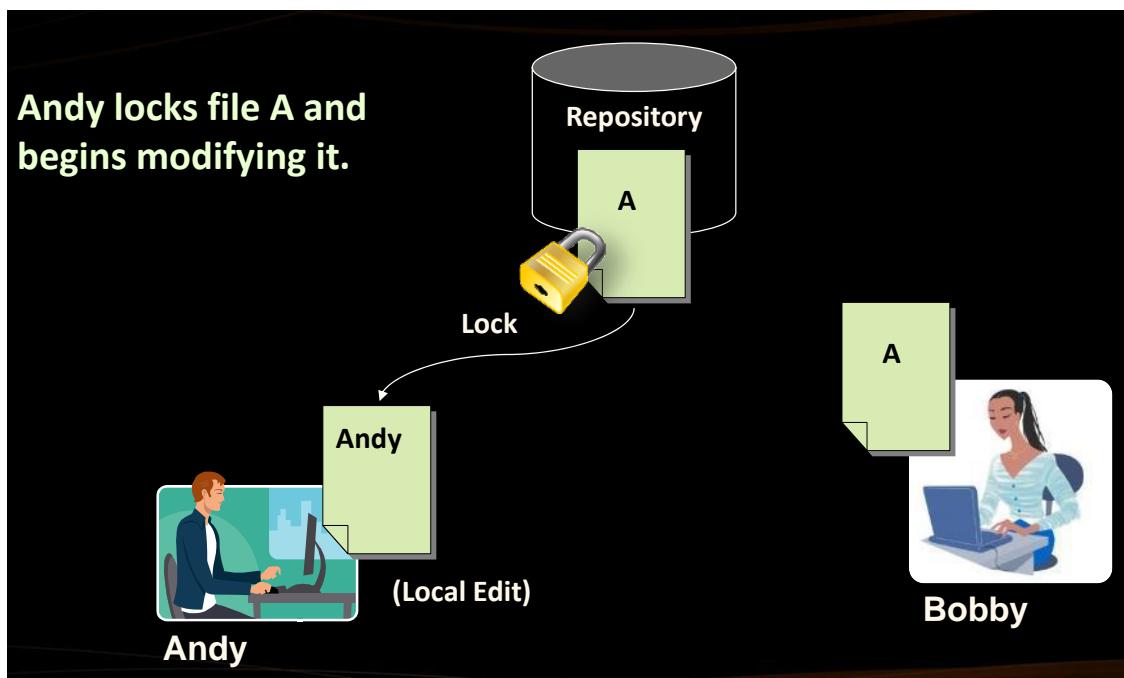
Andy and Bobby
check-out file A.

The check-out is done
without locking. They
just get a local copy.



a. The Lock-Modify-Unlock Model (2)

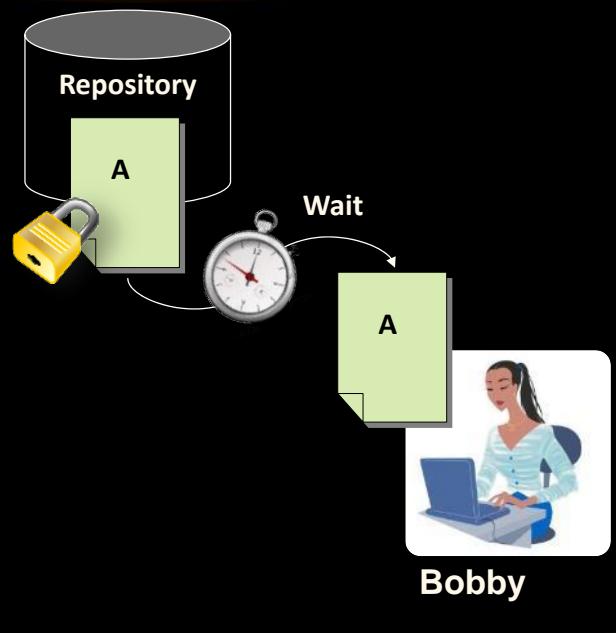
Andy locks file A and
begins modifying it.



a. The Lock-Modify-Unlock Model (3)

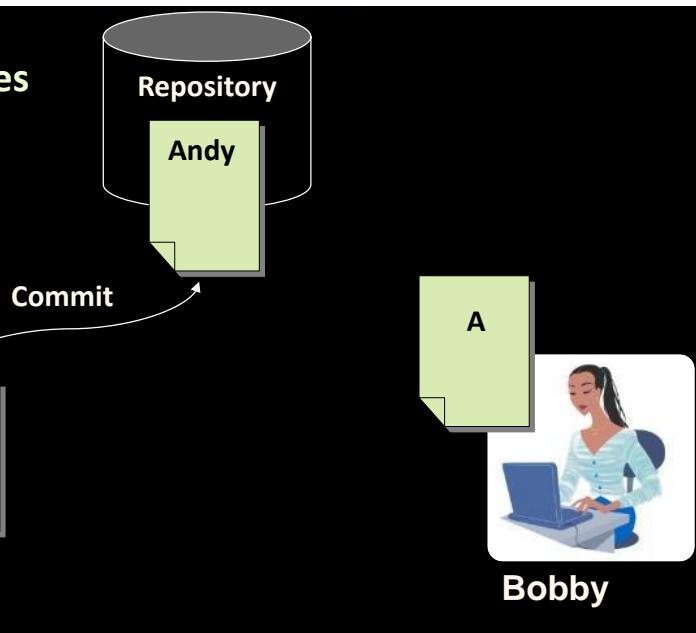
Bobby tries to lock the file too, but she can't.

Bobby waits for Andy to finish and unlock the file.



a. The Lock-Modify-Unlock Model (4)

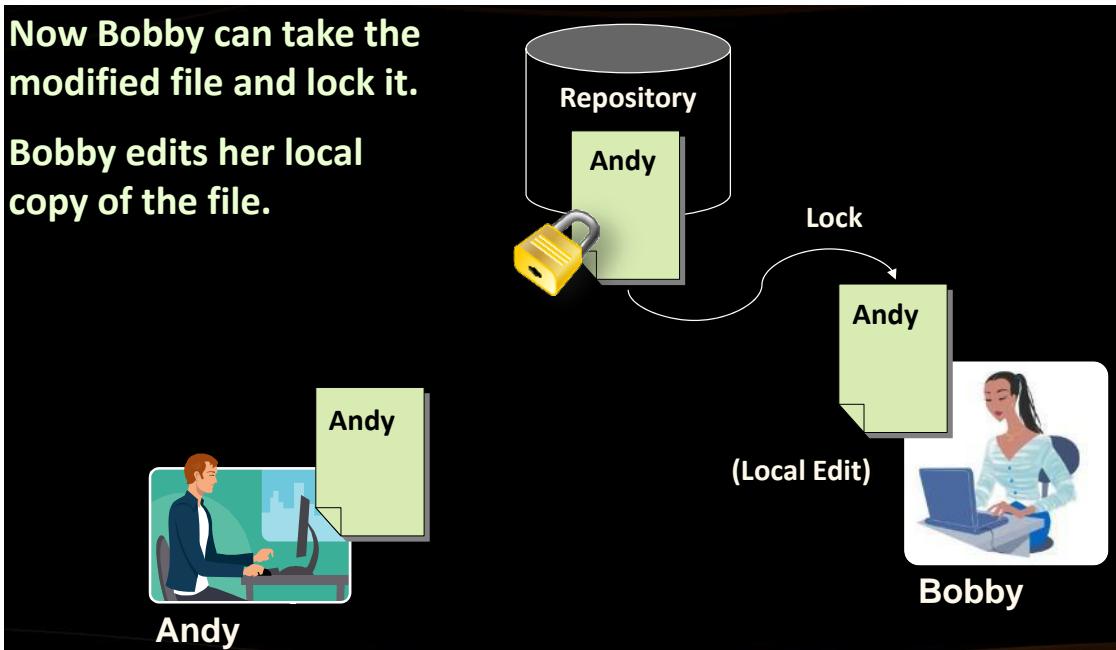
Andy commits his changes and unlocks the file.



a. The Lock-Modify-Unlock Model (5)

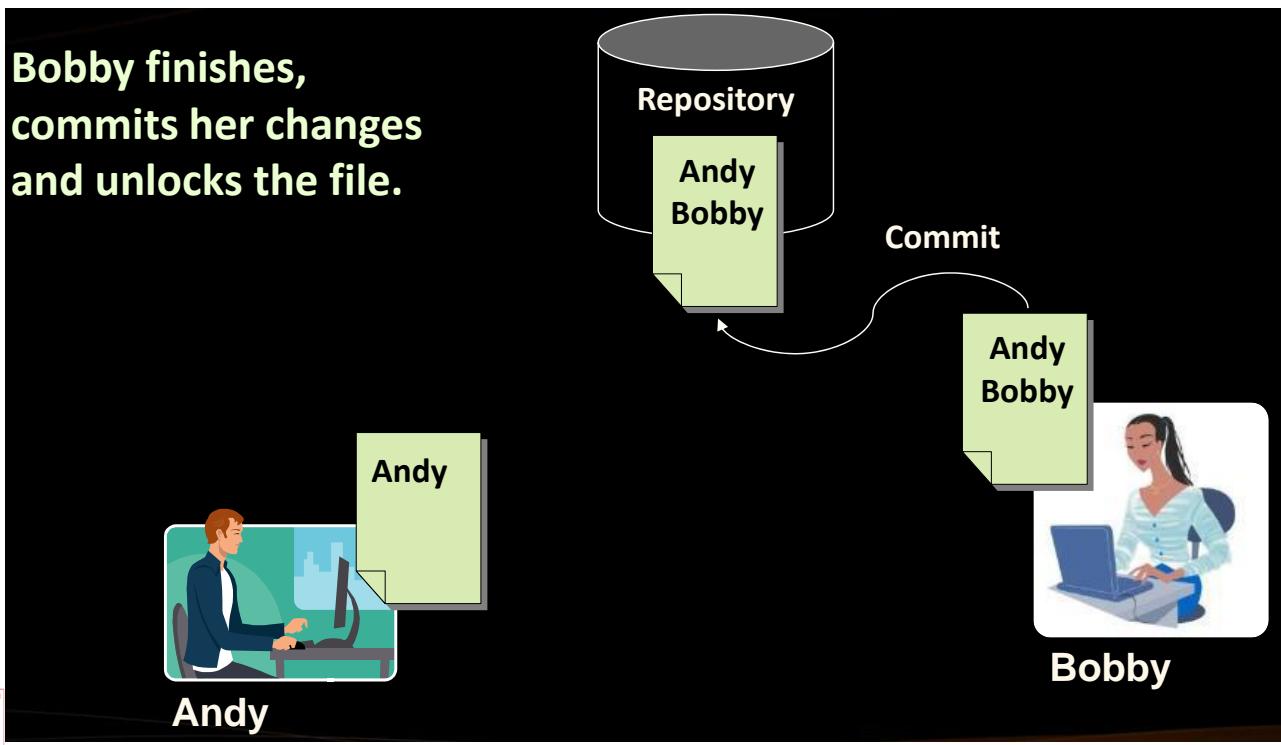
Now Bobby can take the modified file and lock it.

Bobby edits her local copy of the file.

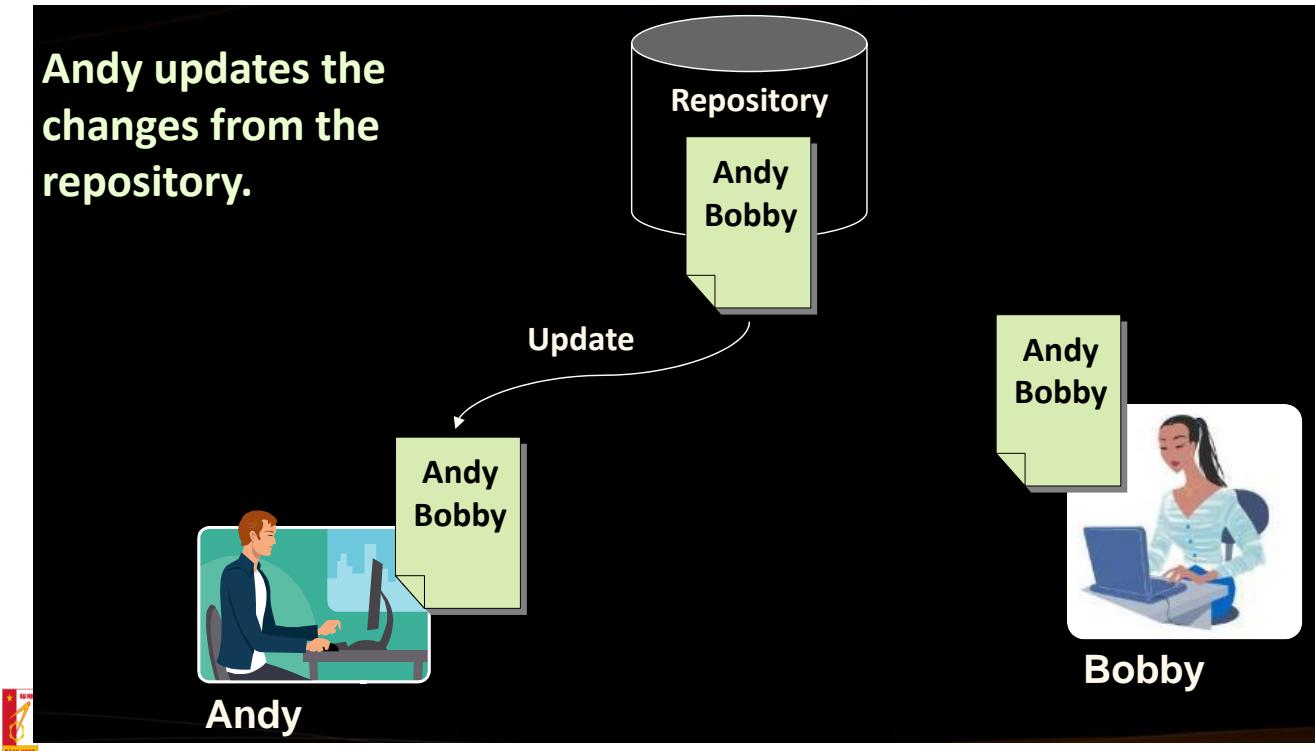


a. The Lock-Modify-Unlock Model (6)

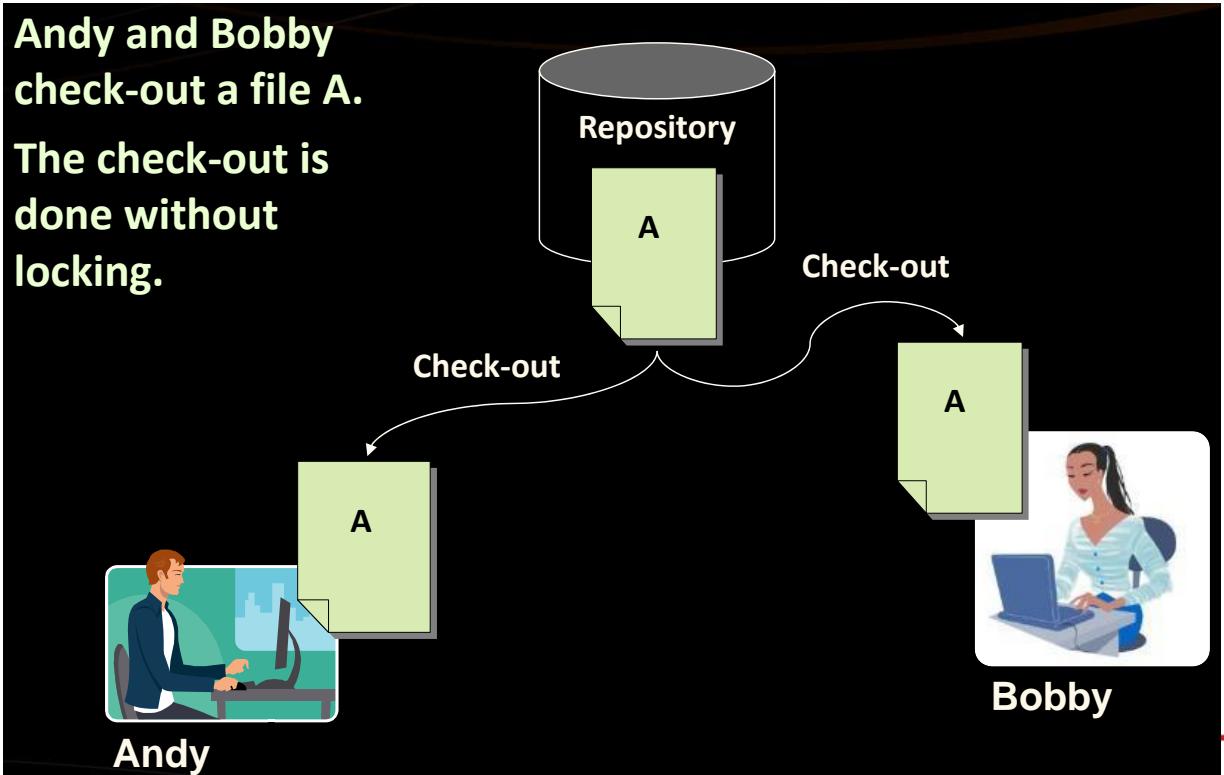
Bobby finishes, commits her changes and unlocks the file.



a. The Lock-Modify-Unlock Model (7)

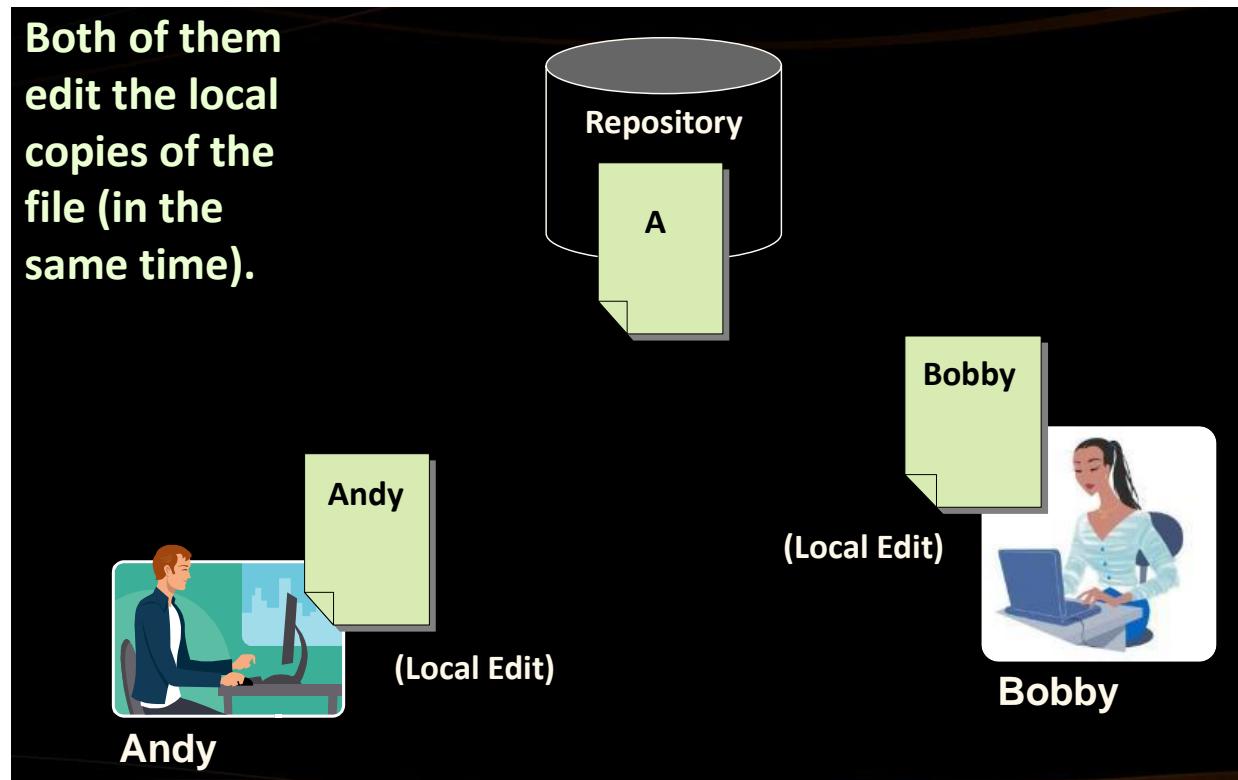


b. The Copy-Modify-Merge Model



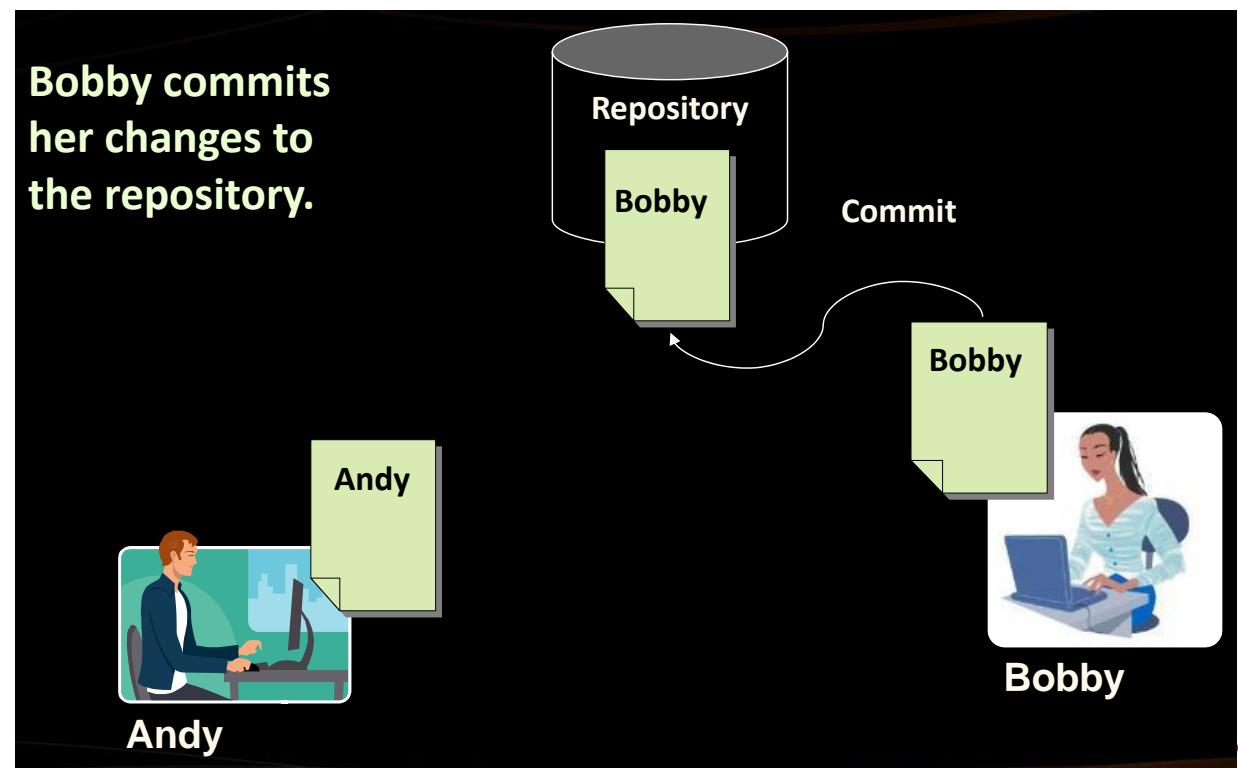
b. The Copy-Modify-Merge Model (2)

Both of them edit the local copies of the file (in the same time).

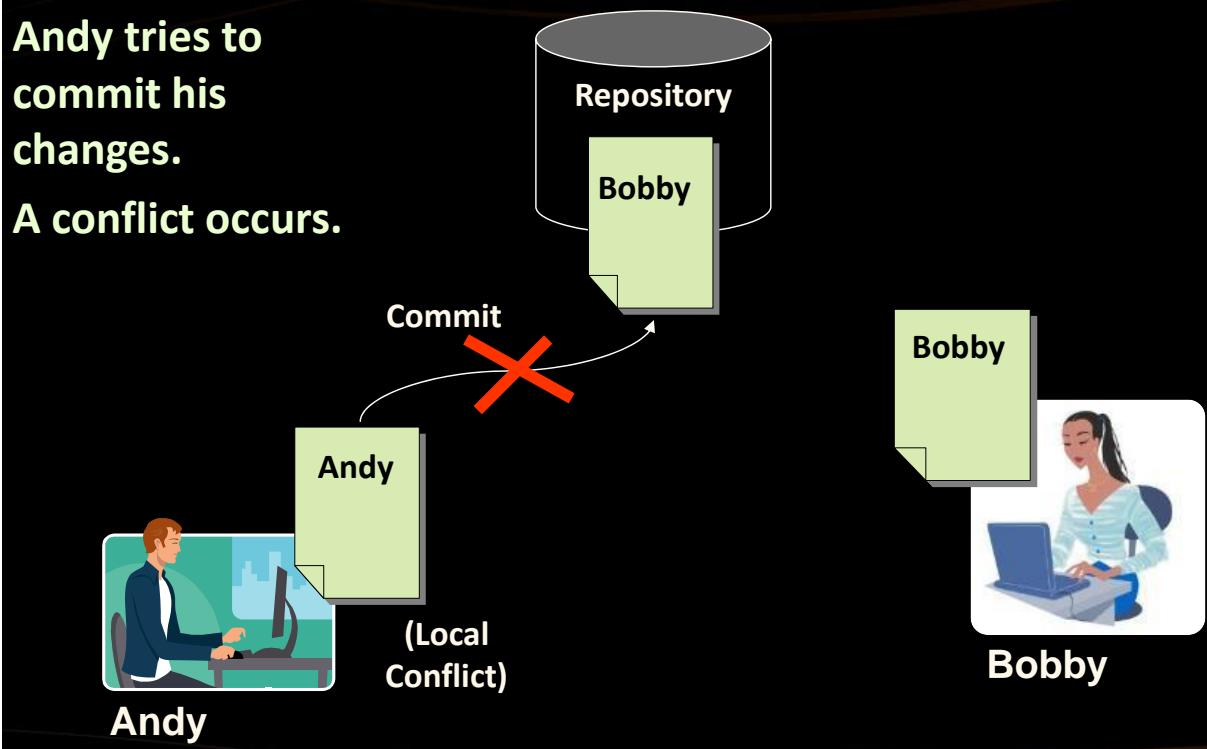


b. The Copy-Modify-Merge Model (3)

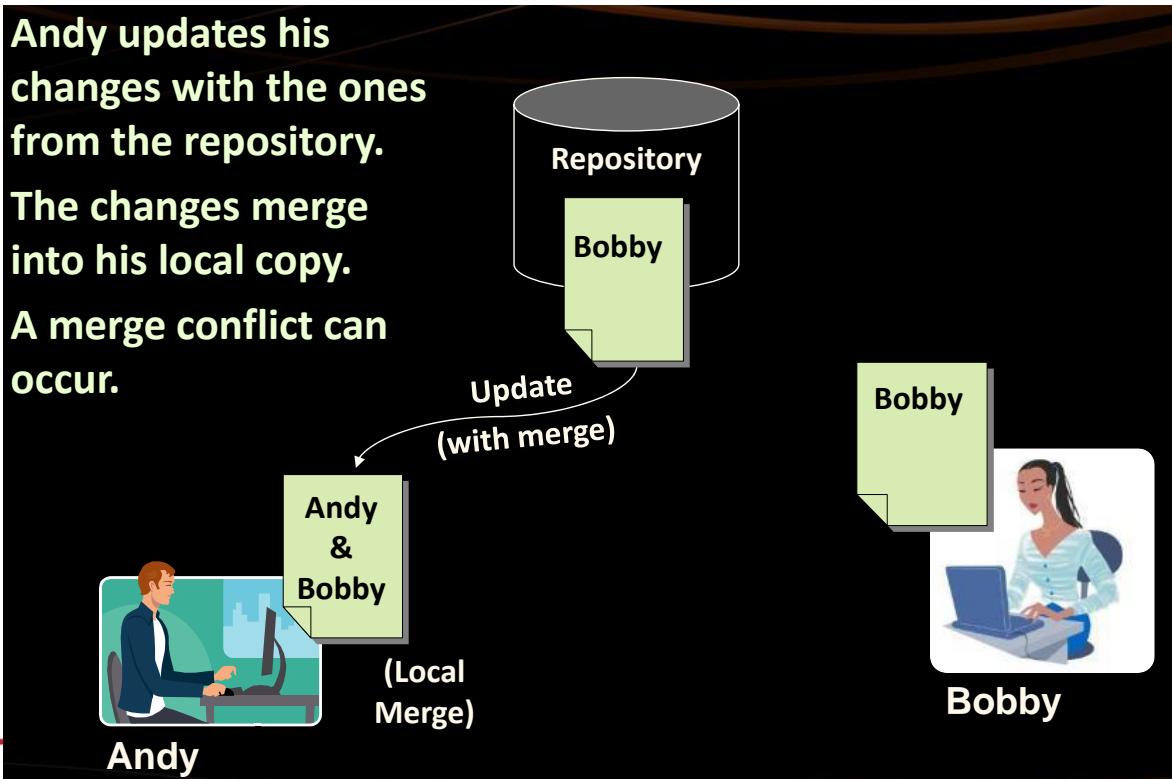
Bobby commits her changes to the repository.



b. The Copy-Modify-Merge Model (4)



b. The Copy-Modify-Merge Model (5)

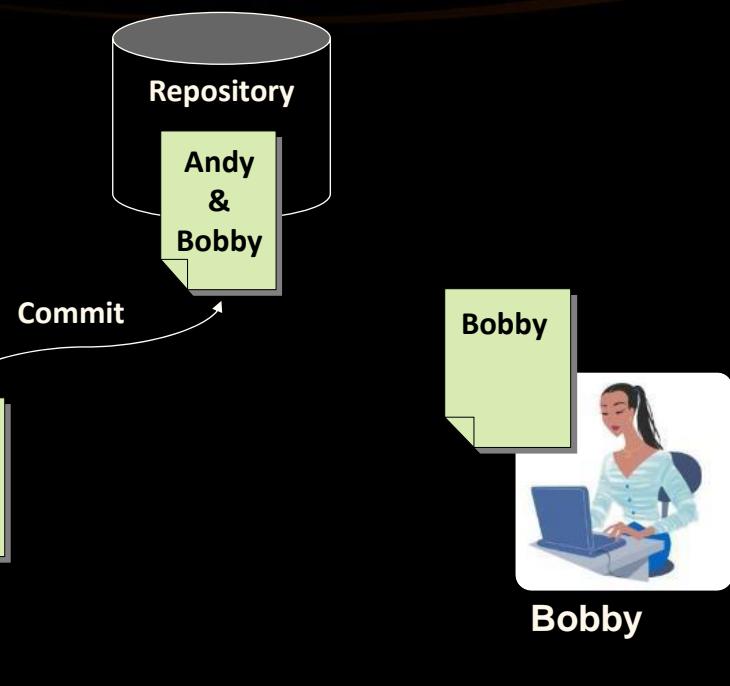


b. The Copy-Modify-Merge Model (6)

Andy commits the merged changes to the repository.

A common version with the changes of Andy and Bobby is inserted.

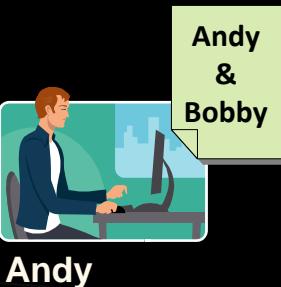
Andy



b. The Copy-Modify-Merge Model (7)

Bobby updates the changes from the repository.

She gets the common version with both changes from Andy and Bobby.



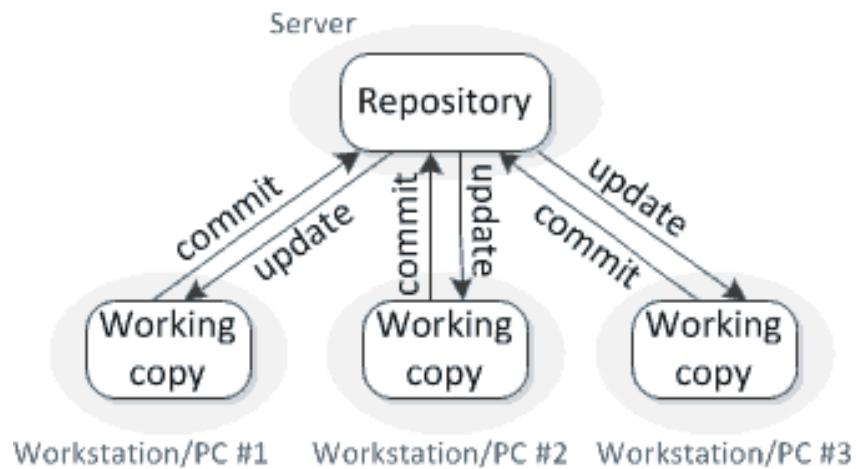
Andy



4.3. Central vs. Distributed version control

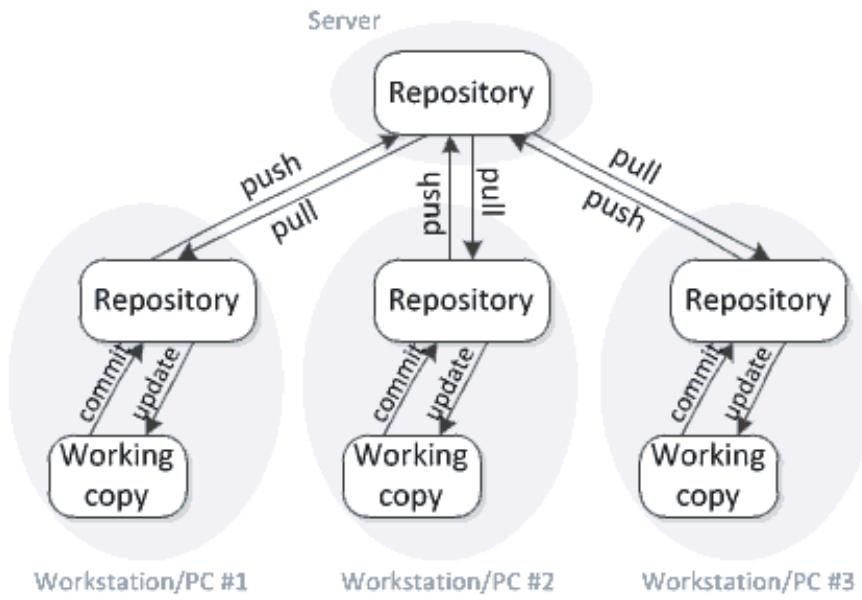
- version control
 - Only one repository

Centralized version control



Distributed Version Control

Distributed version control



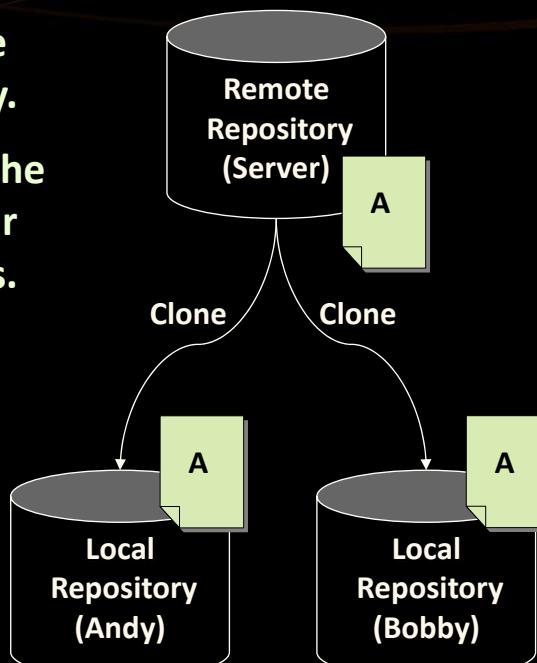
Distributed Version Control (1)

Andy and Bobby clone the remote repository locally.

They both have the same files in their local repositories.



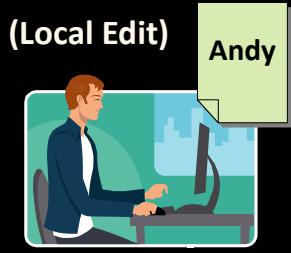
Andy



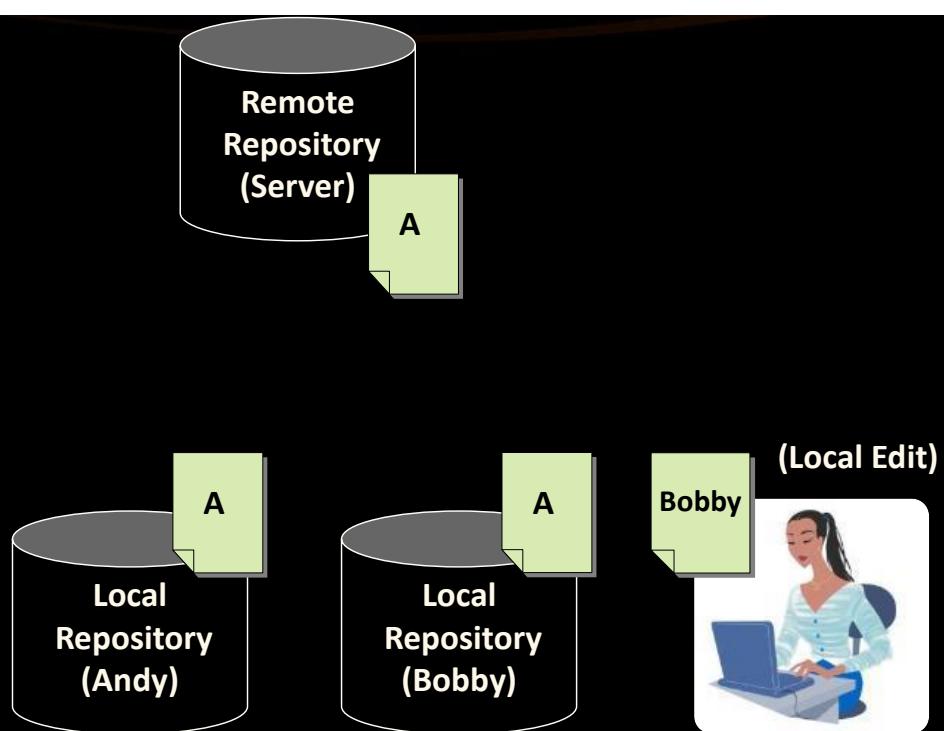
Bobby

Distributed Version Control (2)

Andy and Bobby work locally on a certain file A.



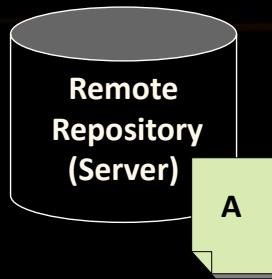
Andy



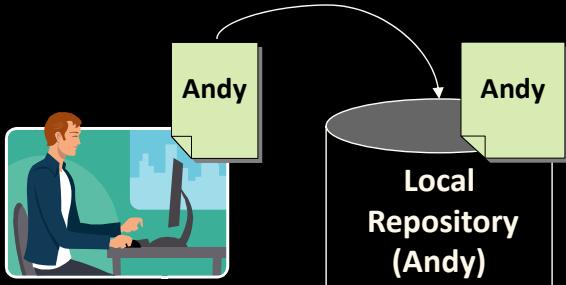
Bobby

Distributed Version Control (3)

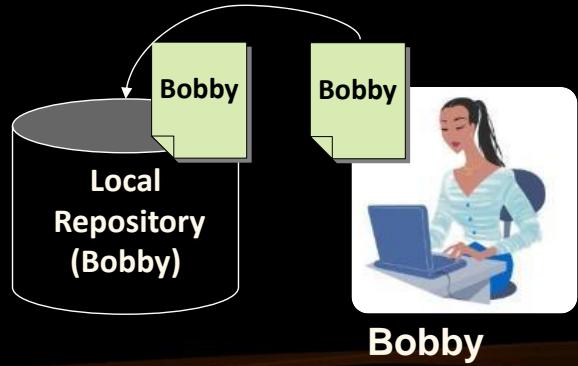
Andy and Bobby commit locally the modified file A into their local repositories.



Commit (locally)



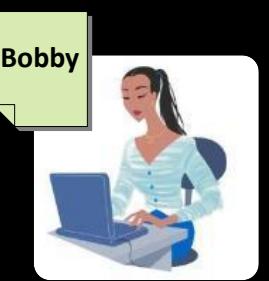
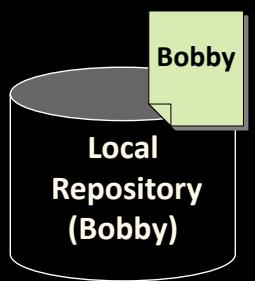
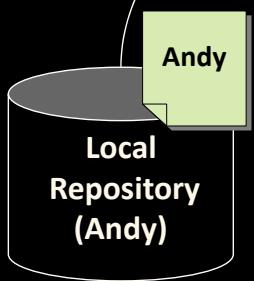
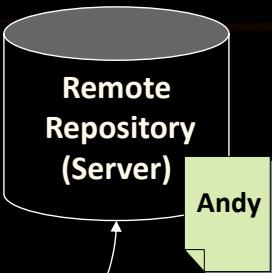
Commit (locally)



Distributed Version Control (4)

Andy pushes the file A to the remote repository.

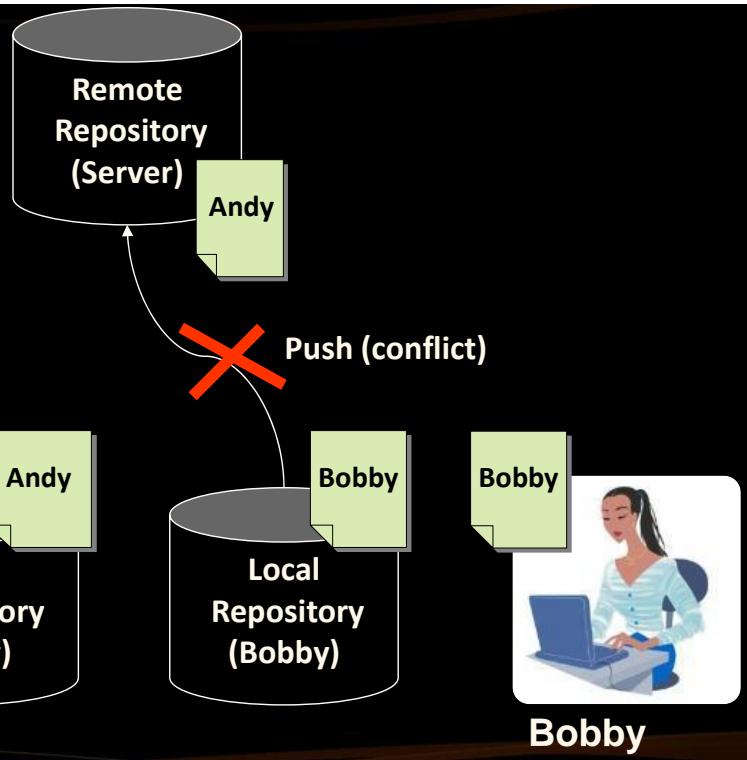
Still no conflicts occur.



Distributed Version Control (5)

Bobby tries to push her changes.

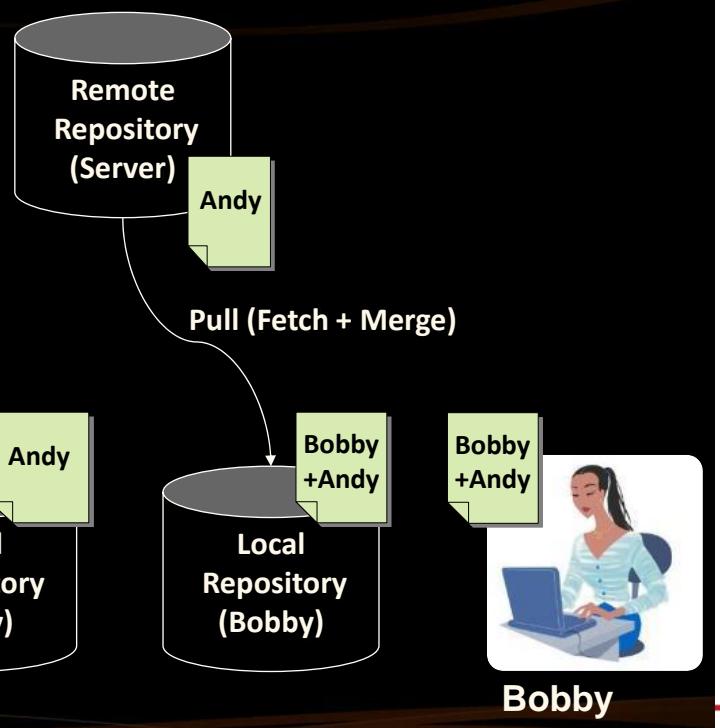
A versioning conflict occurs.



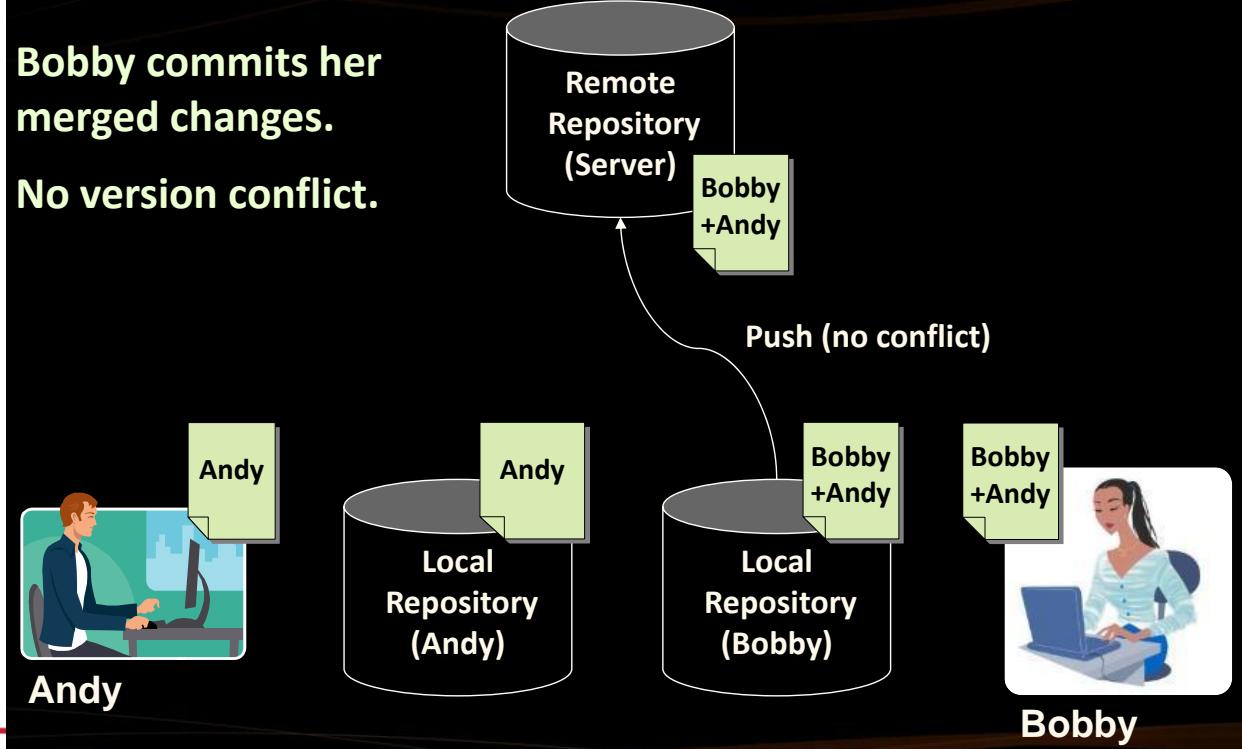
Distributed Version Control (6)

Bobby merges the her local files with the files from the remote repository.

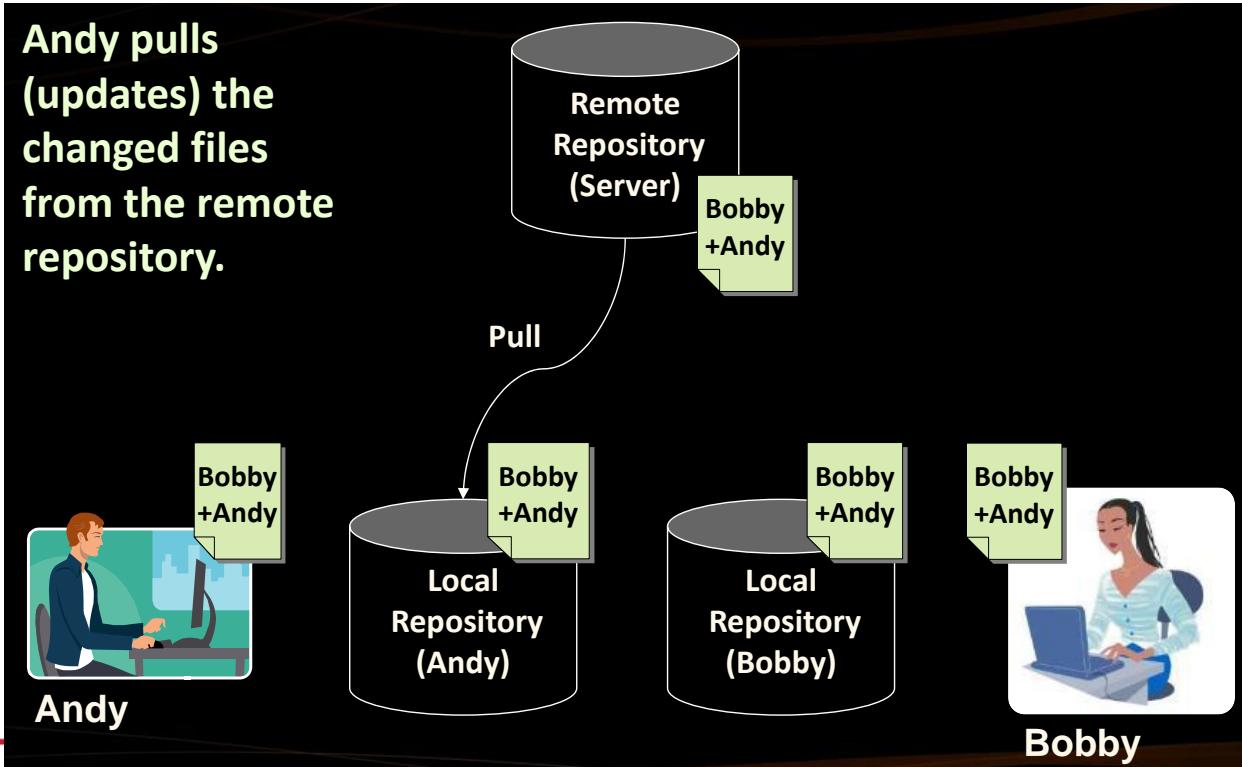
Conflicts are locally resolved.



Distributed Version Control (7)



Distributed Version Control (8)



4.3. Vocabulary

- Repository (source control repository)
 - A server that stores the files (documents)
 - Keeps a change log
- Revision, Version
 - Individual version (state) of a document that is a result of multiple changes
- Check-Out, Clone
 - Retrieves a working copy of the files from a remote repository into a local directory
 - It is possible to lock the files



Vocabulary

- Change
 - A modification to a local file (document) that is under version control
- Change Set / Change List
 - A set of changes to multiple files that are going to be committed at the same time
- Commit, Check-In
 - Submits the changes made from the local working copy to the repository
 - Automatically creates a new version
 - Conflicts may occur!



Vocabulary

- Conflict
 - The simultaneous change to a certain file by multiple users
 - Can be solved automatically and manually
- Update, Get Latest Version, Fetch / Pull
 - Download the latest version of the files from the repository to a local working directory + merge conflicting files
- Undo Check-Out, Revert / Undo Changes
 - Cancels the local changes

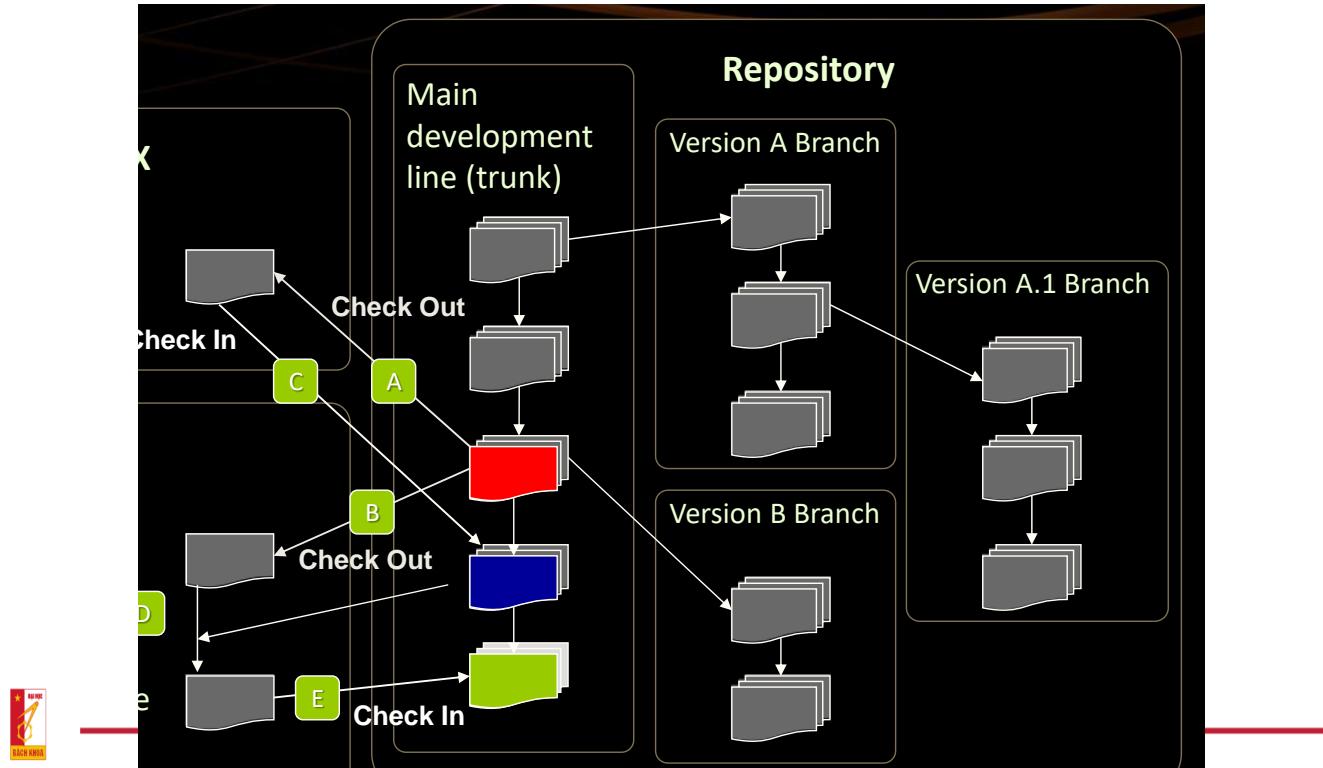


Vocabulary

- Merge
 - Combines the changes to a file changed locally and simultaneously in the repository
 - Can be automated in most cases
- Label / Tag
 - Labels mark with a name a group of files in a given version
 - For example a release
- Branch / Branching
 - Division of the repositories in a number of separate workflows



Version Control: Typical Scenario



70

4.4. Tools

- Central version control
 - SVN (Subversion)
 - TFS
 - Source safe (commercial)
- Distributed version control
 - Git
 - Mercurial

What is Git?

- Git

- Distributed source-control system
- Work with local and remote repositories
- Git bash – command line interface for Git
- Free, open-source
- Has Windows version (msysGit)
 - <http://msysgit.github.io>
 - <https://www.atlassian.com/git/tutorials/setting-up-a-repository>



72

Installing Git

- msysGit Installation on Windows
 - Download Git for Windows from: <http://msysgit.github.io>
 - “Next, Next, Next” does the trick
 - Options to select (they should be selected by default)
 - “Use Git Bash only”
 - “Checkout Windows-style, commit Unix-style endings”
- Git installation on Linux:
`sudo apt-get install git`



Basic Git Commands

- Cloning an existing Git repository
`git clone [remote url]`
- Fetch and merge the latest changes from the remote repository
`git pull`
- Preparing (adding / selecting) files for a commit
`git add [filename] ("git add ." adds everything)`
- Committing to the local repository
`git commit -m "[your message here]"`



Basic Git Commands

- Check the status of your local repository (see the local changes)
`git status`
- Creating a new local repository (in the current directory)
`git init`
- Creating a remote (assign a short name for remote Git URL)
`git remote add [remote name] [remote url]`
- Pushing to a remote (send changes to the remote repository)
`git push [remote name] [local name]`



Using Git: Example

```
mkdir work  
cd work  
git clone https://github.com/SoftUni/test.git dir  
cd test  
dir  
git status  
(edit some file)  
git status  
git add .  
git commit -m "changes"  
git push
```



Project Hosting Sites

- GitHub – <https://github.com>
 - The #1 project hosting site in the world
 - Free for open-source projects
 - Paid plans for private projects
- GitHub provides own Windows client
 - GitHub for Windows
 - <http://windows.github.com>
 - Dramatically simplifies Git
 - For beginners only



Project Hosting Sites

- Google Code –
<http://code.google.com/projecthosting/>
 - Source control (SVN), file release, wiki, tracker
 - Very simple, basic functions only, not feature-rich
 - Free, all projects are public and open source
 - 1-minute signup, without heavy approval process
- SourceForge – <http://www.sourceforge.net>
 - Source control (SVN, Git, ...), web hosting, tracker, wiki, blog, mailing lists, file release, statistics, etc



7º Project Hosting Sites

- CodePlex – <http://www.codeplex.com>
 - Microsoft's open source projects site
 - Team Foundation Server (TFS) infrastructure
 - Source control (TFS), issue tracker, downloads, discussions, wiki, etc.
 - Free, all projects are public and open source
- Bitbucket – <http://bitbucket.org>
 - Source control (Mercurial), issue tracker, wiki, management tools
 - Private projects, free and paid editions



- Introduction to version control
 - <https://www.youtube.com/watch?v=gY2JwRfin1M>
 - See Episode 1-> 5
- Bitbucket
 - https://www.youtube.com/watch?v=BtEvnE79jxY&list=PL57RkP_325rLuT3EmFTf3IkoLw93Iw1_8



17 – Verification and Testing

(end of lecture)