Source Control Systems

SVN, Git, GitHub

Svetlin Nakov

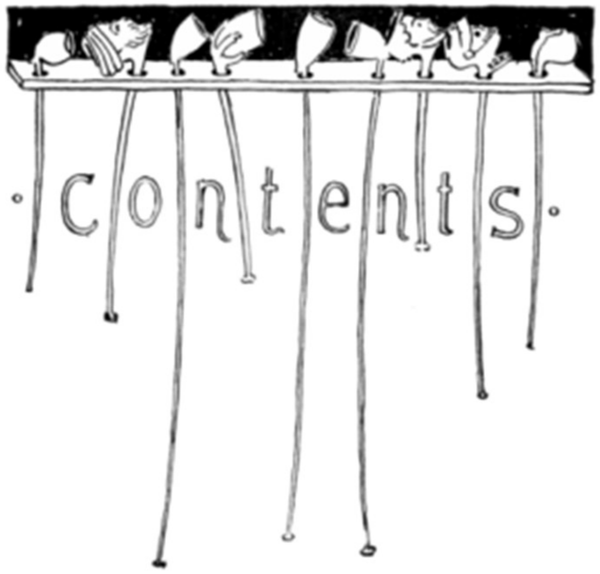
Technical Trainer

www.nakov.com

Software University

http://softuni.bg

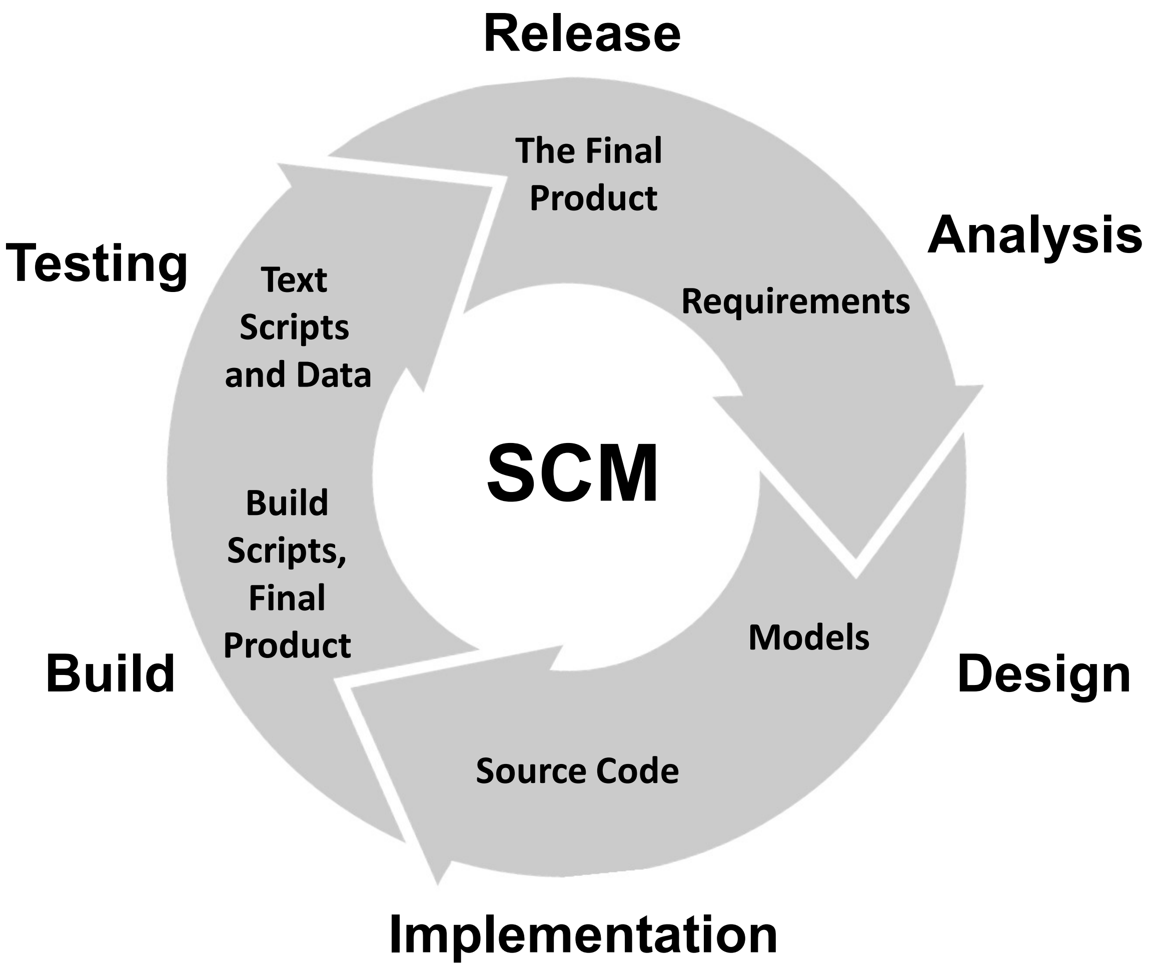
Table of Contents



Software Configuration Management (SCM)

- Version Control ≈ Software Configuration Management (SCM)
  - A software engineering discipline
  - Consists of techniques, practices and tools for working on shared source code and files
  - Mechanisms for management, control and tracking the changes
  - Defines the process of change management
  - Keeps track of what is happening in the project over the time
  - Solves conflicts in the changes

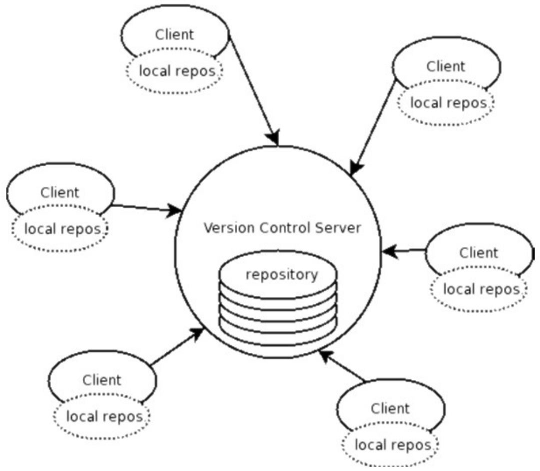
SCM and the Software Development Lifecycle



FOUNDATION

Version Control

Managing Different Versions of the Same File / Document

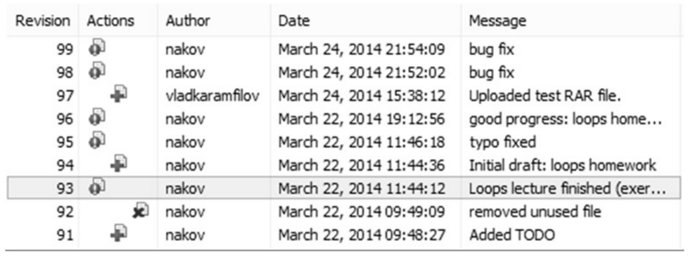


Version Control Systems (VCS)

- Functionality
  - File versions control
  - Merge and differences search
  - Branching
  - File locking
  - Console and GUI clients
- Well known products
  - CVS, Subversion (SVN) – free, open source
  - Git, Mercurial – distributed, free, open source
  - Perforce, Microsoft TFS – commercial

Version Control (Revision Control)

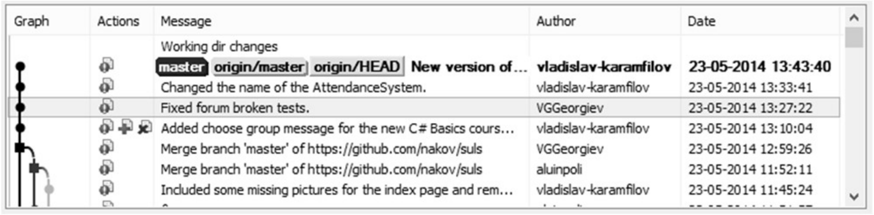
- Constantly used in software engineering
  - During the software development
  - While working with documents
- Changes are identified with an increment of the version number
  - for example 1.0, 2.0, 2.17
- Version numbers are historically linked with the person who created them
  - Full change logs are kept



Revision	Actions	Author	Date	Message
99	✎	nakov	March 24, 2014 21:54:09	bug fix
98	✎	nakov	March 24, 2014 21:52:02	bug fix
97	+	vladkaramfilov	March 24, 2014 15:38:12	Uploaded test RAR file.
96	✎	nakov	March 22, 2014 19:12:56	good progress: loops home...
95	✎	nakov	March 22, 2014 11:46:18	typo fixed
94	+	nakov	March 22, 2014 11:44:36	Initial draft: loops homework
93	✎	nakov	March 22, 2014 11:44:12	Loops lecture finished (exer...
92	✗	nakov	March 22, 2014 09:49:09	removed unused file
91	+	nakov	March 22, 2014 09:48:27	Added TODO

Change Log

- Systems for version control keep a complete change log (history)
  - The date and hour of every change
  - The user who made the change
  - The files changed + old and new version
- Old versions can be retrieved, examined and compared
- It is possible to return to an old version (revert)



Graph	Actions	Message	Author	Date
		Working dir changes		
		master origin/master origin/HEAD New version of…	**vladislav-karamfilov**	**23-05-2014 13:43:40**
		Changed the name of the AttendanceSystem.	vladislav-karamfilov	23-05-2014 13:33:41
		Fixed forum broken tests.	VGGeorgiev	23-05-2014 13:27:22
		Added choose group message for the new C# Basics cours…	vladislav-karamfilov	23-05-2014 13:10:04
		Merge branch 'master' of https://github.com/nakov/suls	VGGeorgiev	23-05-2014 12:59:26
		Merge branch 'master' of https://github.com/nakov/suls	aluinpoli	23-05-2014 11:52:11
		Included some missing pictures for the index page and rem…	vladislav-karamfilov	23-05-2014 11:45:24

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 (2)

- 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!

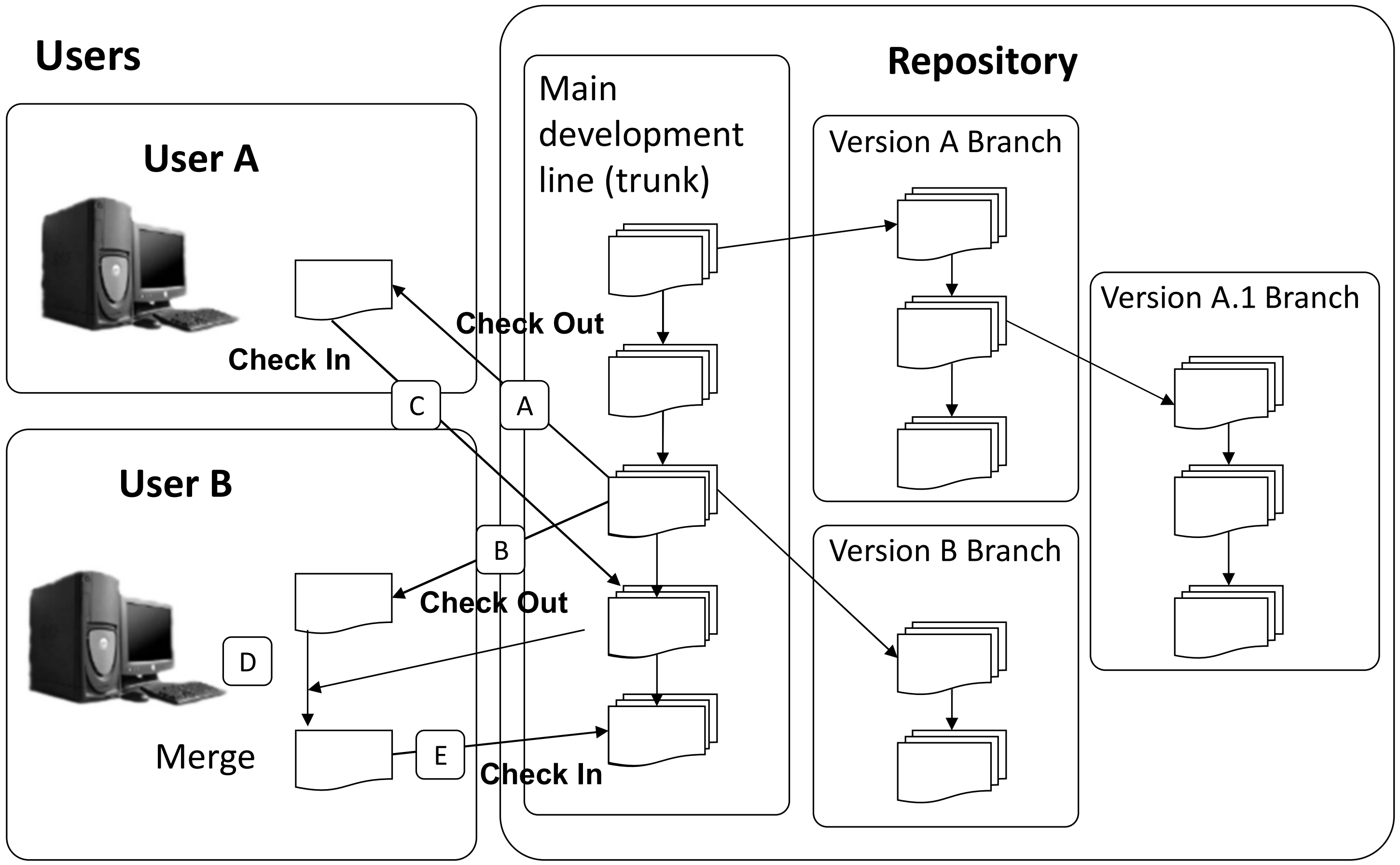
Vocabulary (3)

- 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
  - Restores their state from the repository

Vocabulary (4)

- 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

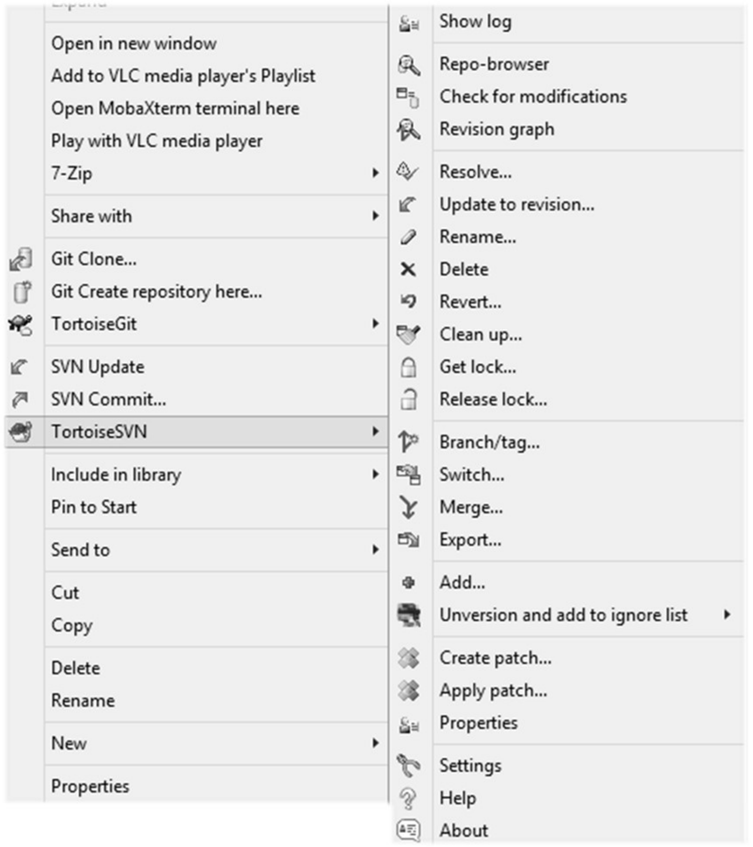


Subversion

Using Subversion and TortoiseSVN

Subversion (SVN)

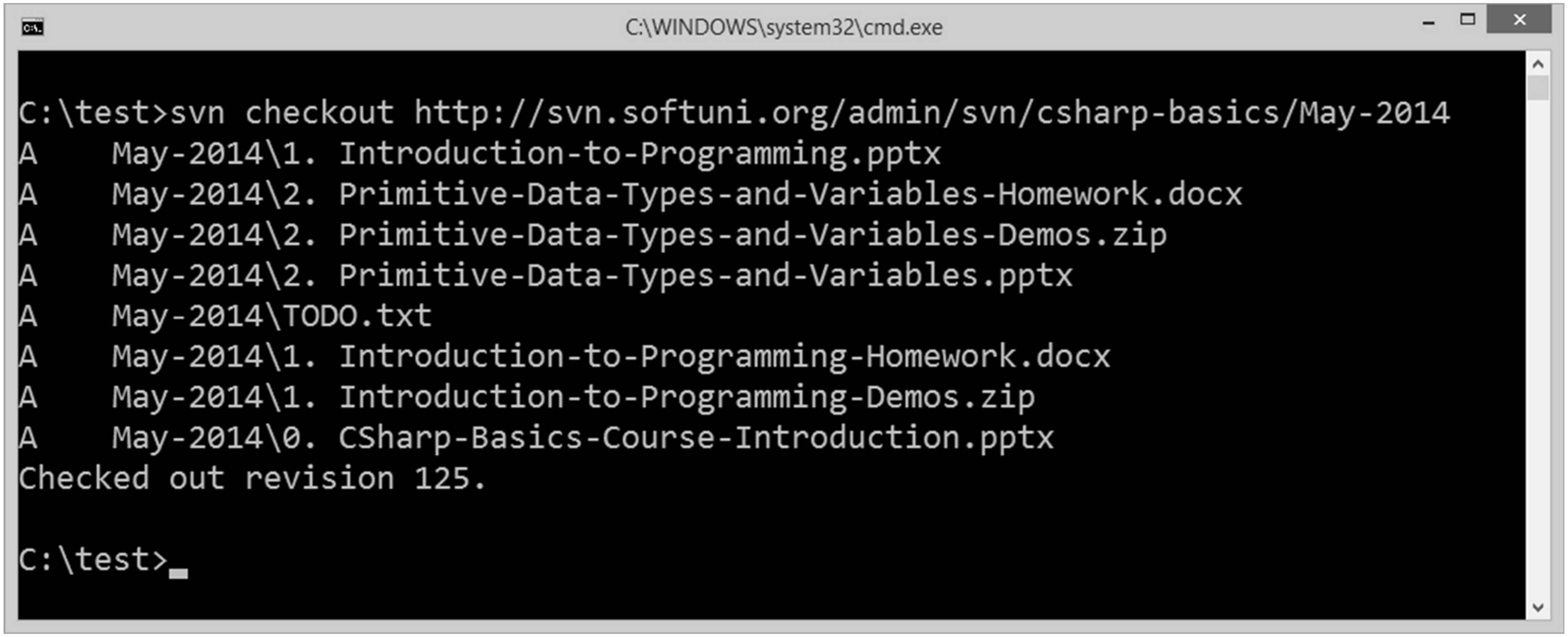
- Subversion (SVN)
  - Open source SCM repository
  - http://subversion.tigris.org
  - Runs on Linux, Windows, Mac OS
- Console client
  - **svn**
- GUI client – TortoiseSVN
  - http://tortoisesvn.tigris.org
- Visual Studio / Eclipse plug-ins



Subversion – Features

- Versioning of the directory structure
- Complete change log
  - Deletion of files and directories
  - Renaming of files and directories
  - Saving of files or directories
- Can work on it’s own or integrated with Apache as a module
- Simple to use, based on central SVN repository
- Works effectively with tags and branches

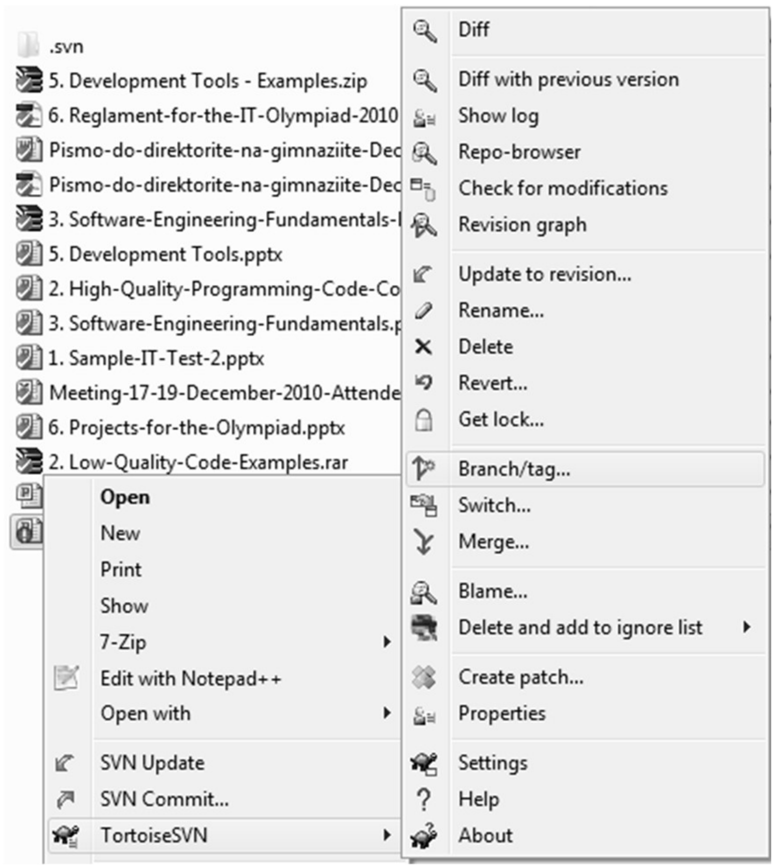
SVN – Console Client



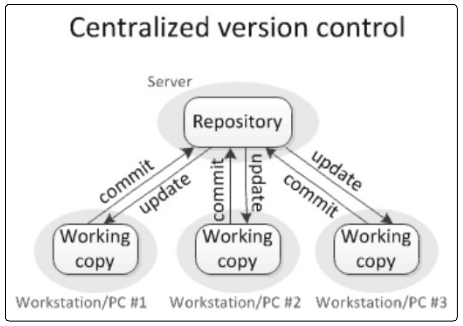
```
C:\test>svn checkout http://svn.softuni.org/admin/svn/csharp-basics/May-2014
A    May-2014\1. Introduction-to-Programming.pptx
A    May-2014\2. Primitive-Data-Types-and-Variables-Homework.docx
A    May-2014\2. Primitive-Data-Types-and-Variables-Demos.zip
A    May-2014\2. Primitive-Data-Types-and-Variables.pptx
A    May-2014\TODO.txt
A    May-2014\1. Introduction-to-Programming-Homework.docx
A    May-2014\1. Introduction-to-Programming-Demos.zip
A    May-2014\0. CSharp-Basics-Course-Introduction.pptx
Checked out revision 125.

C:\test>_
```

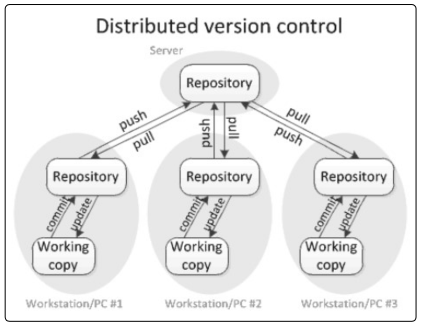
TortoiseSVN

- TortoiseSVN
  - Open source GUI client for Subversion for Windows
  - Integrated in Windows Explorer
  - http://tortoisesvn.tigris.org


.svn
5. Development Tools - Examples.zip
6. Reglament-for-the-IT-Olympiad-2010
Pismo-do-direktorite-na-gimnaziite-Dec
Pismo-do-direktorite-na-gimnaziite-Dec
3. Software-Engineering-Fundamentals-
5. Development Tools.pptx
2. High-Quality-Programming-Code-Co
3. Software-Engineering-Fundamentals.p
1. Sample-IT-Test-2.pptx
Meeting-17-19-December-2010-Attende
6. Projects-for-the-Olympiad.pptx
2. Low-Quality-Code-Examples.rar
Open
New
Print
Show
7-Zip
Edit with Notepad++
Open with
SVN Update
SVN Commit...
TortoiseSVN
Diff
Diff with previous version
Show log
Repo-browser
Check for modifications
Revision graph
Update to revision...
Rename...
Delete
Revert...
Get lock...
Branch/tag...
Switch...
Merge...
Blame...
Delete and add to ignore list
Create patch...
Properties
Settings
Help
About

18

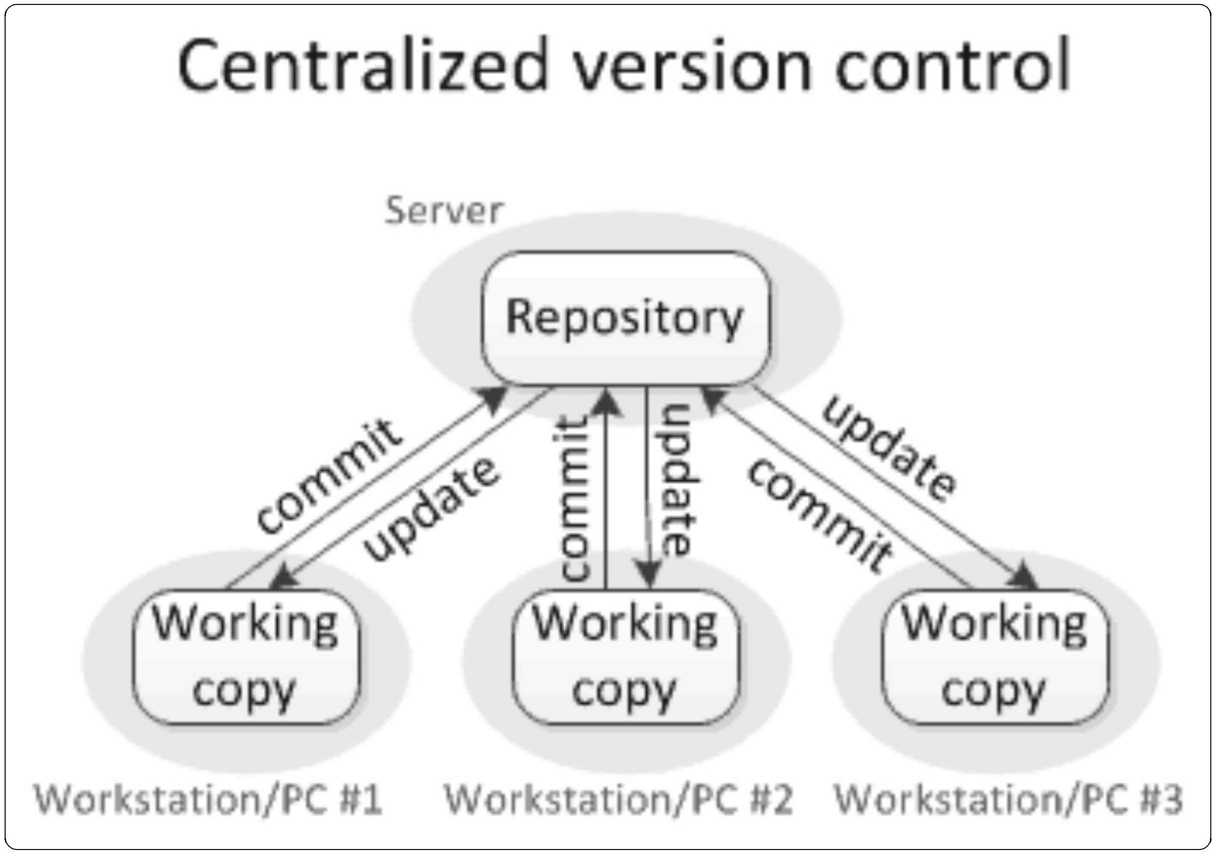




Versioning Models

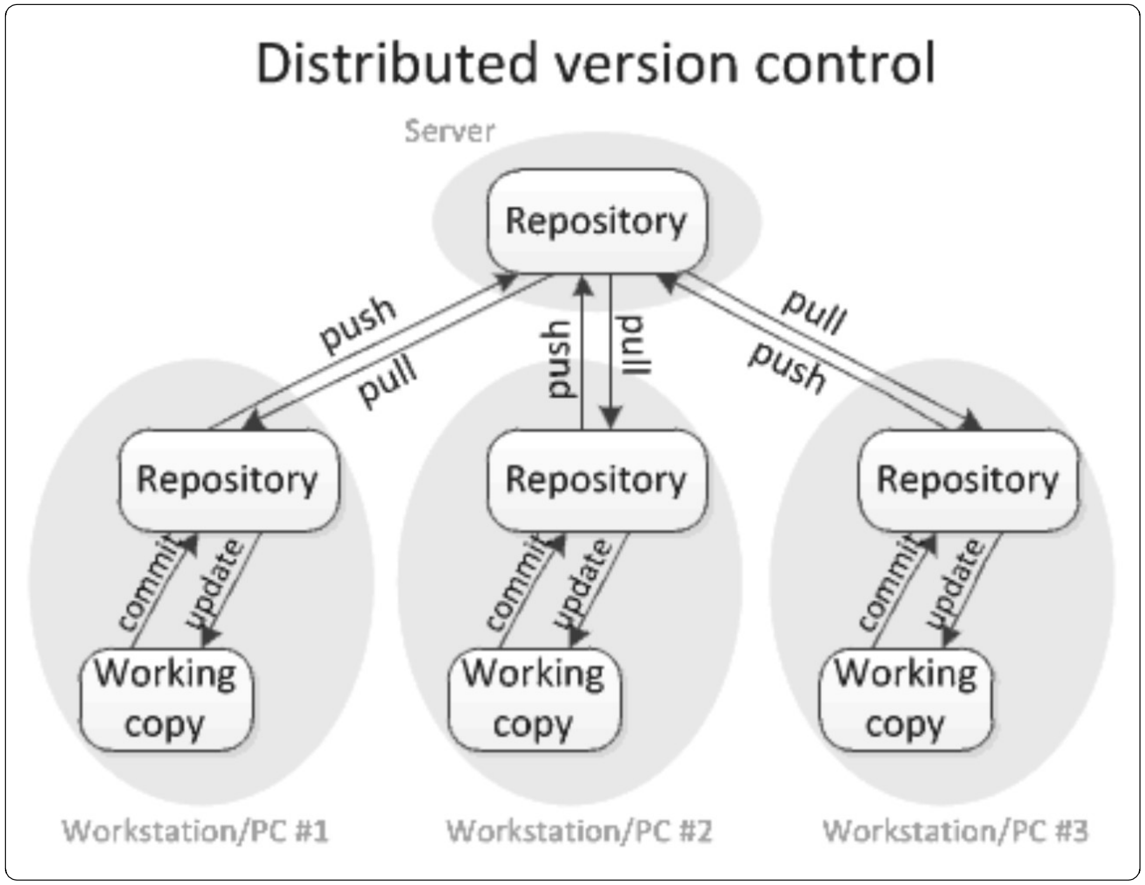
Lock-Modify-Unlock,
Copy-Modify-Merge,
Distributed Version Control

Centralized Version Control



Source: http://homes.cs.washington.edu/~mernst/advice/version-control.html

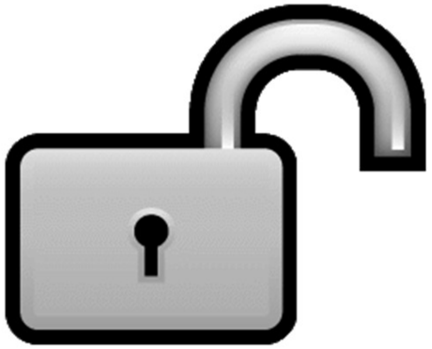
Distributed Version Control

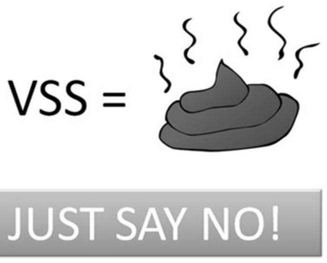


Source: http://homes.cs.washington.edu/~mernst/advice/version-control.html

Versioning Models

- Lock-Modify-Unlock
  - Only one user works on a given file at a time
    - No conflicts occur
    - Users wait each other for the locked files → works for small development teams only
    - Pessimistic concurrency control
  - Examples:
    - Visual SourceSafe (VSS) – old fashioned
    - SVN, Git, TFS (with exclusive locking)
  - Lock-modify-unlock is rarely used





Versioning Models (2)

- Copy-Modify-Merge
  - Users make parallel changes to their own working copies
  - Conflicts are possible when multiple user edit the same file
    - Conflicting changes are merged and the final version emerges (automatic and manual merge)
  - Optimistic concurrency control
  - Examples:
    - SVN, Git, TFS

Versioning Models (3)

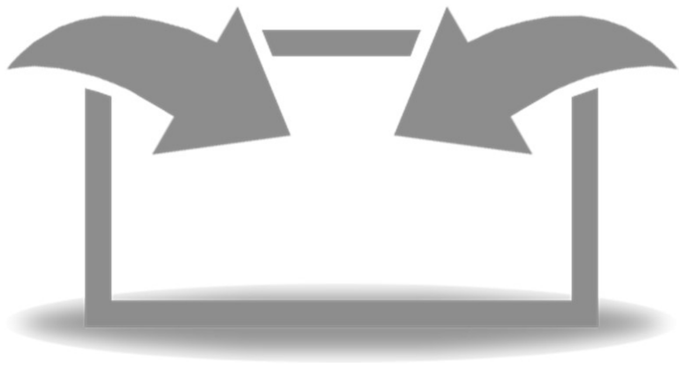
- Distributed Version Control
  - Users work in their own repository
    - Using the Lock-Modify-Unlock model
    - Local changes are locally committed
    - No concurrency, no local conflicts
  - From time to time, the local repository is pushed to the central repository
    - Conflicts are possible and merges often occur
  - Example of distributed version control systems:
    - Git, Mercurial

Problems with Locking

- Administrative problems:
  - Someone locks a given file and forgets about it
  - Time is lost while waiting for someone to release a file → works in small teams only
- Unneeded locking of the whole file
  - Different changes are not necessary in conflict
  - Example of non-conflicting changes:
    - Andy works at the begging of the file
    - Bobby works at the end of the file

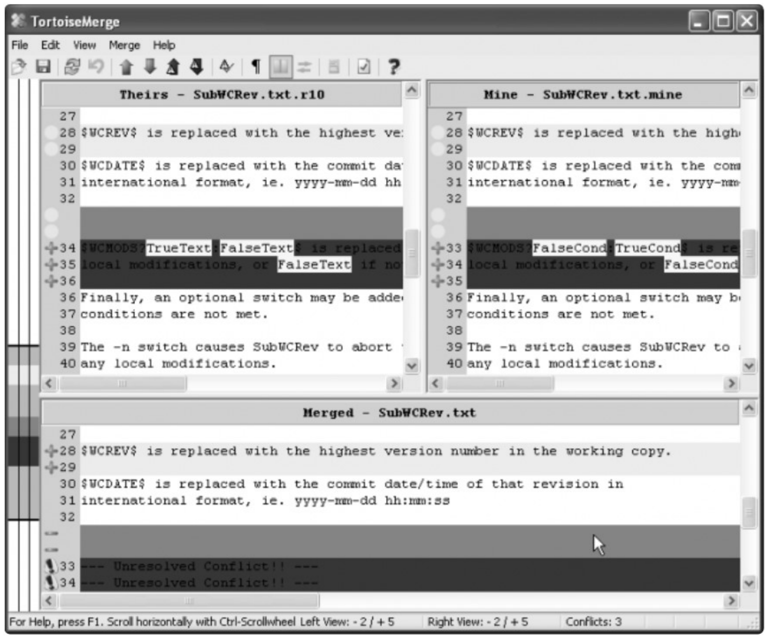
Merging Problems

- When a file is concurrently modified, changes should be merged
  - Merging is hard!
  - It is not always automatic process
- Coordination and responsibility between the developers is required
  - Commit changes as early as finished
  - Do not commit code that does not compile or blocks the work of the others
  - Leave meaningful comments at each commit



File Comparison / Merge Tools

- During manual merge use file comparison
- There are visual comparison / merge tools:
  - TortoiseMerge
  - WinDiff
  - AraxisMerge
  - WinMerge
  - BeyondCompare
  - CompareIt
  - …



Theirs - SubWCRev.txt.r10

```

$WCREV$ is replaced with the highest ve

$WCDATE$ is replaced with the commit da
international format, ie. yyyy-mm-dd hh


$WCMODS?TrueText:FalseText$ is replaced
local modifications, or FalseText if no

Finally, an optional switch may be adde
conditions are not met.

The -n switch causes SubWCRev to abort
any local modifications.
```

Mine - SubWCRev.txt.mine

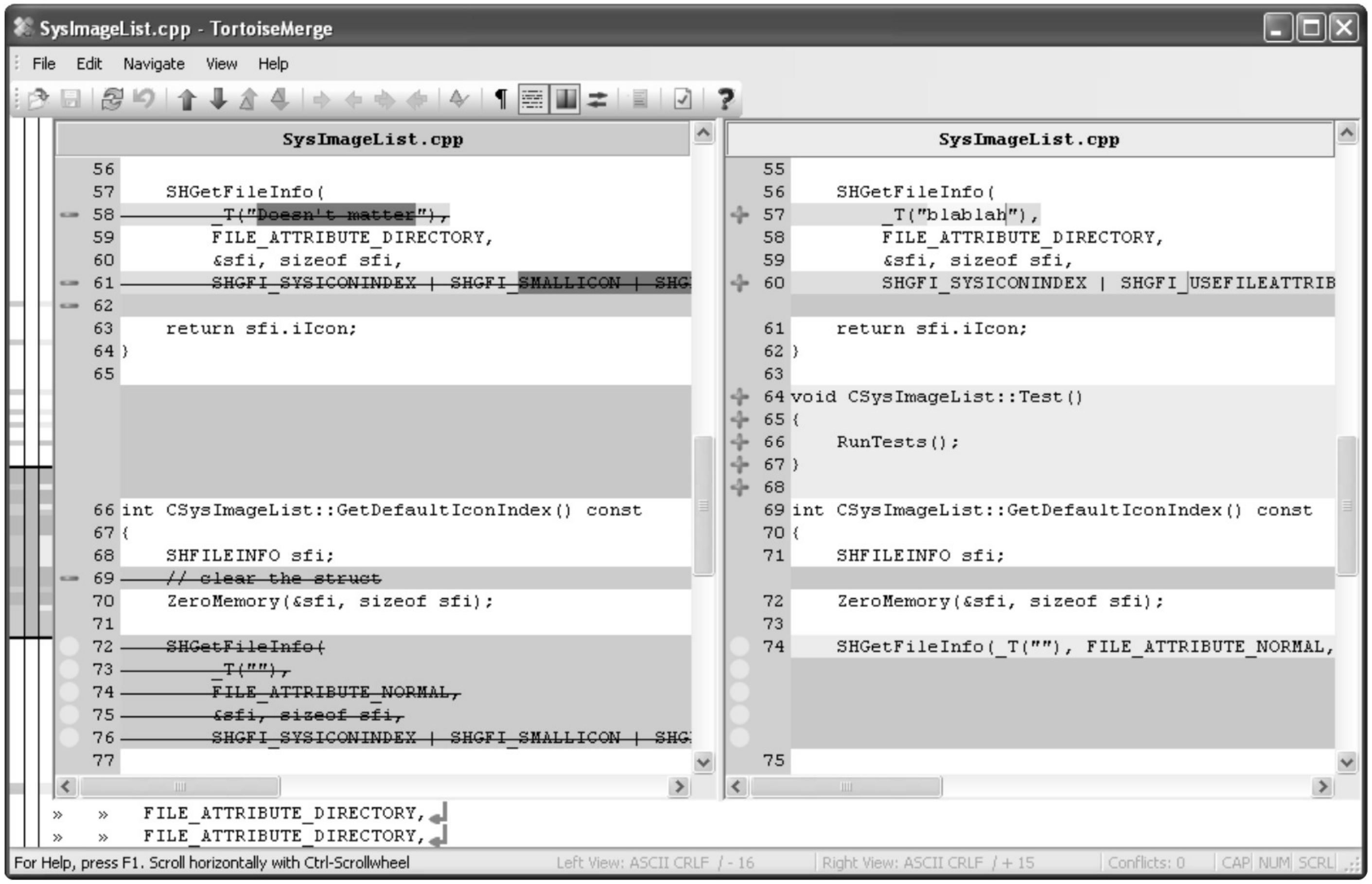
```

$WCREV$ is replaced with the high

$WCDATE$ is replaced with the comm
international format, ie. yyyy-mm

$WCMODS?FalseCond:TrueCond$ is re
local modifications, or FalseCond

Finally, an optional switch may b
conditions are not met.

The -n switch causes SubWCRev to
any local modifications.
```

Merged - SubWCRev.txt

```

$WCREV$ is replaced with the highest version number in the working copy.

$WCDATE$ is replaced with the commit date/time of that revision in
international format, ie. yyyy-mm-dd hh:mm:ss

--- Unresolved Conflict!! ---
--- Unresolved Conflict!! ---
```

For Help, press F1. Scroll horizontally with Ctrl-Scrollwheel Left View: - 2 / + 5 Right View: - 2 / + 5 Conflicts: 3

File Comparison – Example



SysImageList.cpp

```

    SHGetFileInfo(
        ~~_T("Doesn't matter"),~~
        FILE_ATTRIBUTE_DIRECTORY,
        &sfi, sizeof sfi,
        ~~SHGFI_SYSICONINDEX | SHGFI_SMALLICON | SHG~~

    return sfi.iIcon;
}

int CSysImageList::GetDefaultIconIndex() const
{
    SHFILEINFO sfi;
    ~~// clear the struct~~
    ZeroMemory(&sfi, sizeof sfi);

    ~~SHGetFileInfo(~~
        ~~_T(""),~~
        ~~FILE_ATTRIBUTE_NORMAL,~~
        ~~&sfi, sizeof sfi,~~
        ~~SHGFI_SYSICONINDEX | SHGFI_SMALLICON | SHG~~

```

SysImageList.cpp

```

    SHGetFileInfo(
        _T("blablah"),
        FILE_ATTRIBUTE_DIRECTORY,
        &sfi, sizeof sfi,
        SHGFI_SYSICONINDEX | SHGFI_USEFILEATTRIB

    return sfi.iIcon;
}

void CSysImageList::Test()
{
    RunTests();
}

int CSysImageList::GetDefaultIconIndex() const
{
    SHFILEINFO sfi;

    ZeroMemory(&sfi, sizeof sfi);

    SHGetFileInfo(_T(""), FILE_ATTRIBUTE_NORMAL,

```

```
»    »    FILE_ATTRIBUTE_DIRECTORY,
»    »    FILE_ATTRIBUTE_DIRECTORY,
```

For Help, press F1. Scroll horizontally with Ctrl-Scrollwheel | Left View: ASCII CRLF / - 16 | Right View: ASCII CRLF / + 15 | Conflicts: 0 | CAP | NUM | SCRL

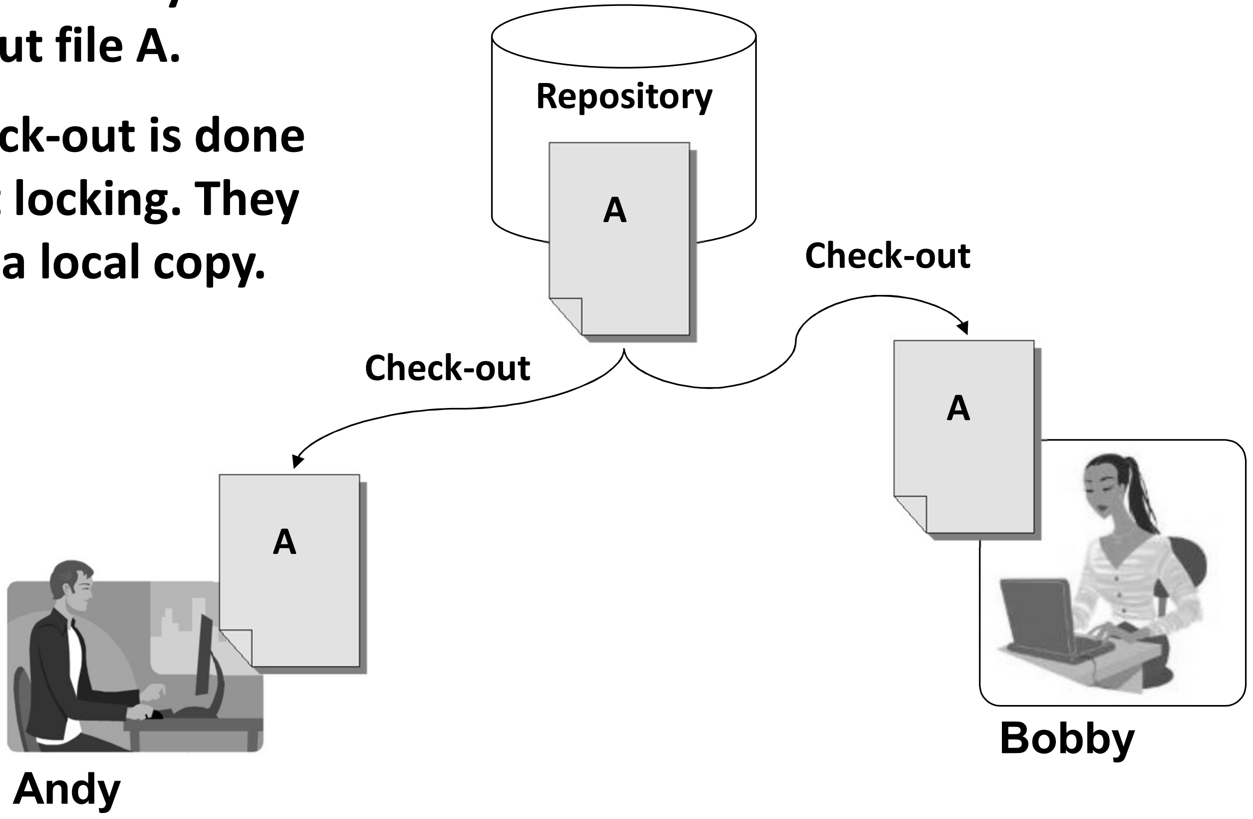
FOUNDATION

The "Lock-Modify-Unlock" Model

The Lock-Modify-Unlock Model (1)

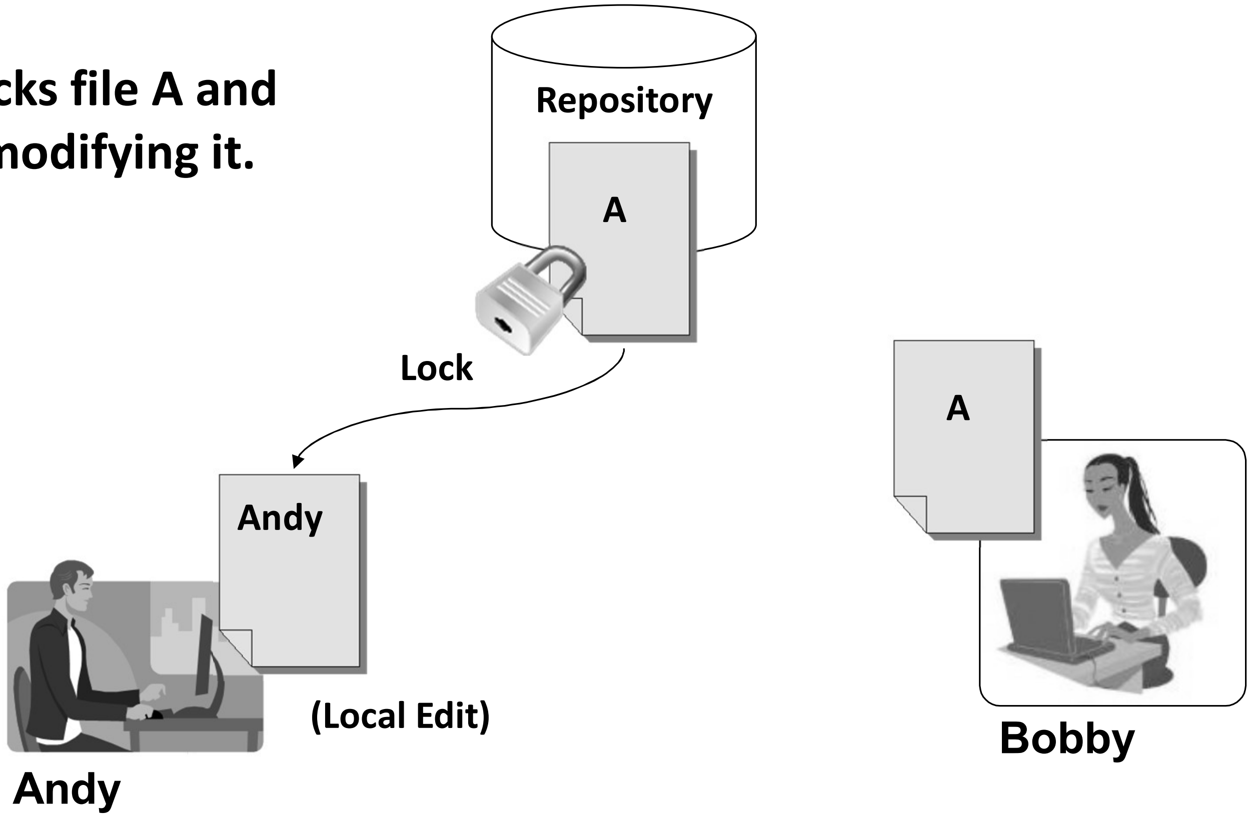
Andy and Bobby check-out file A.

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



The Lock-Modify-Unlock Model (2)

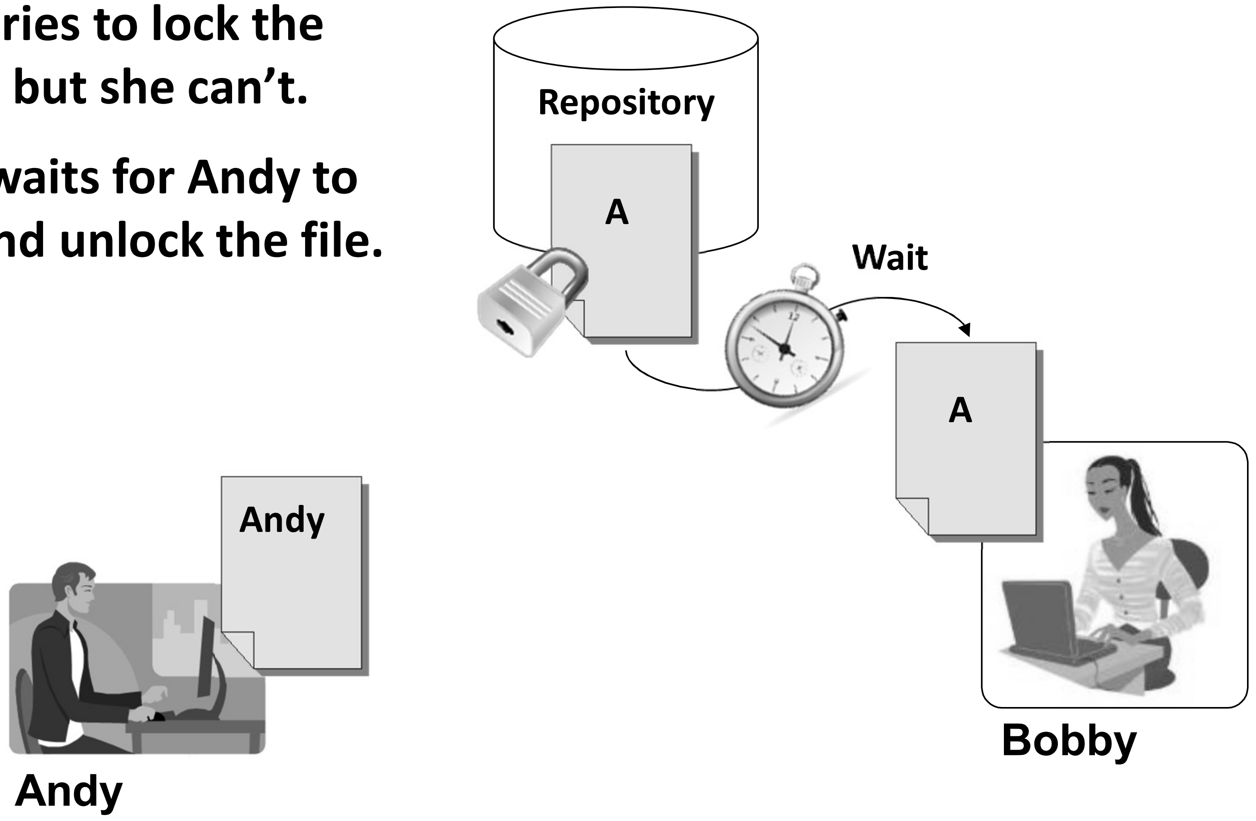
Andy locks file A and begins modifying it.



The Lock-Modify-Unlock Model (3)

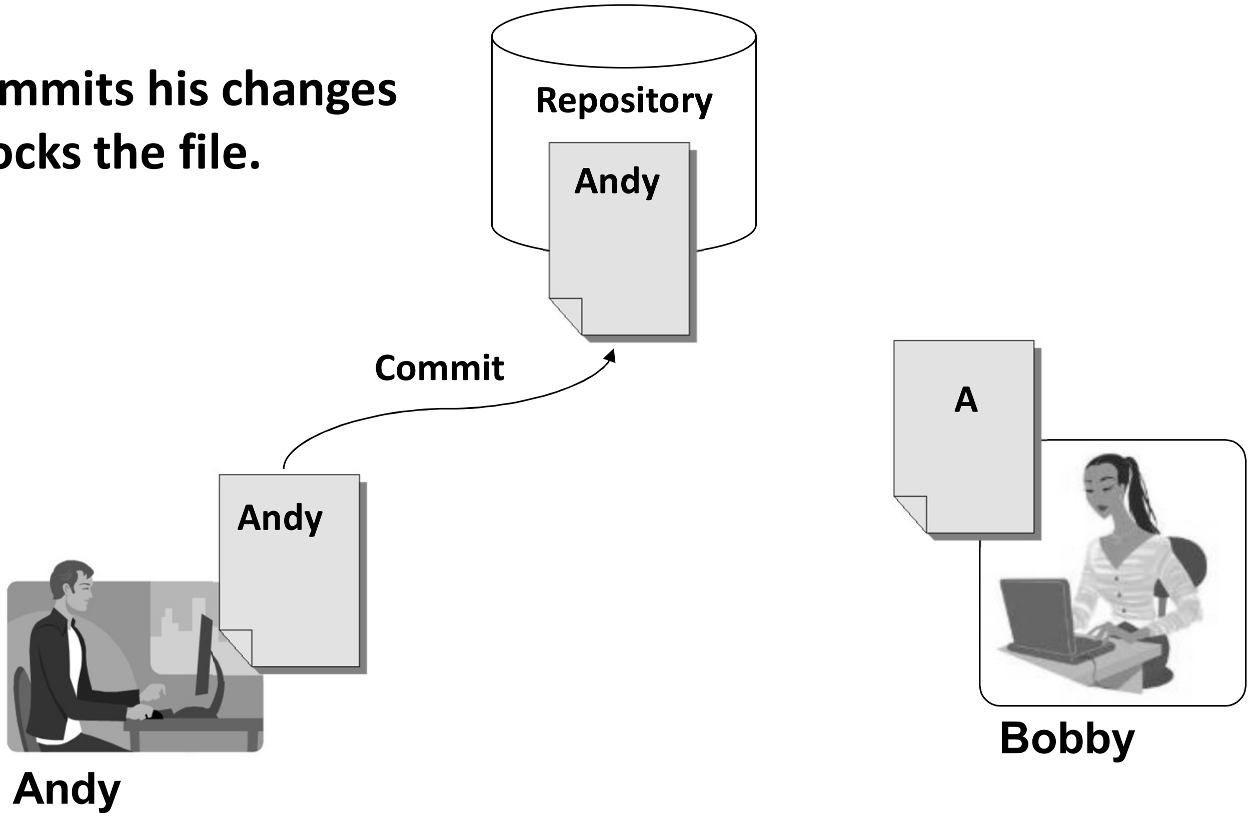
Bobby tries to lock the file too, but she can't.

Bobby waits for Andy to finish and unlock the file.



The Lock-Modify-Unlock Model (4)

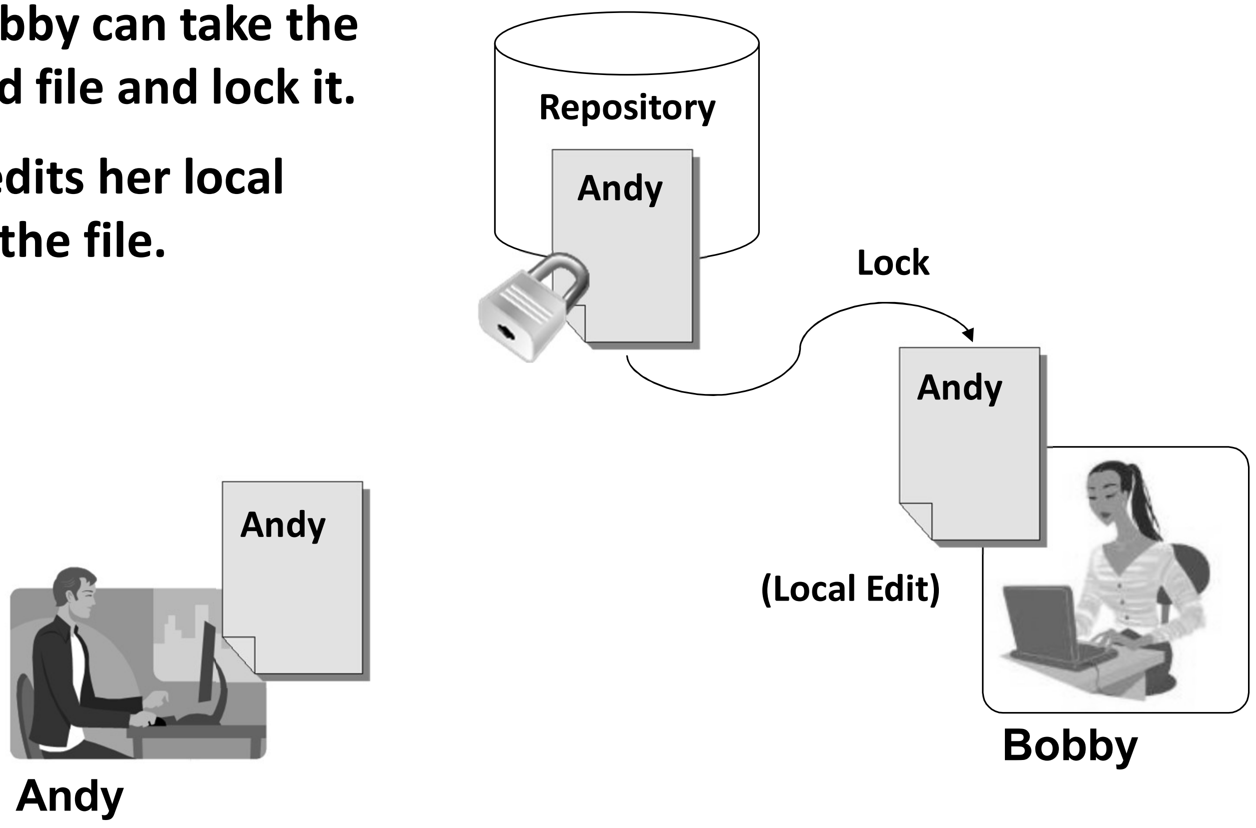
Andy commits his changes and unlocks the file.



The Lock-Modify-Unlock Model (5)

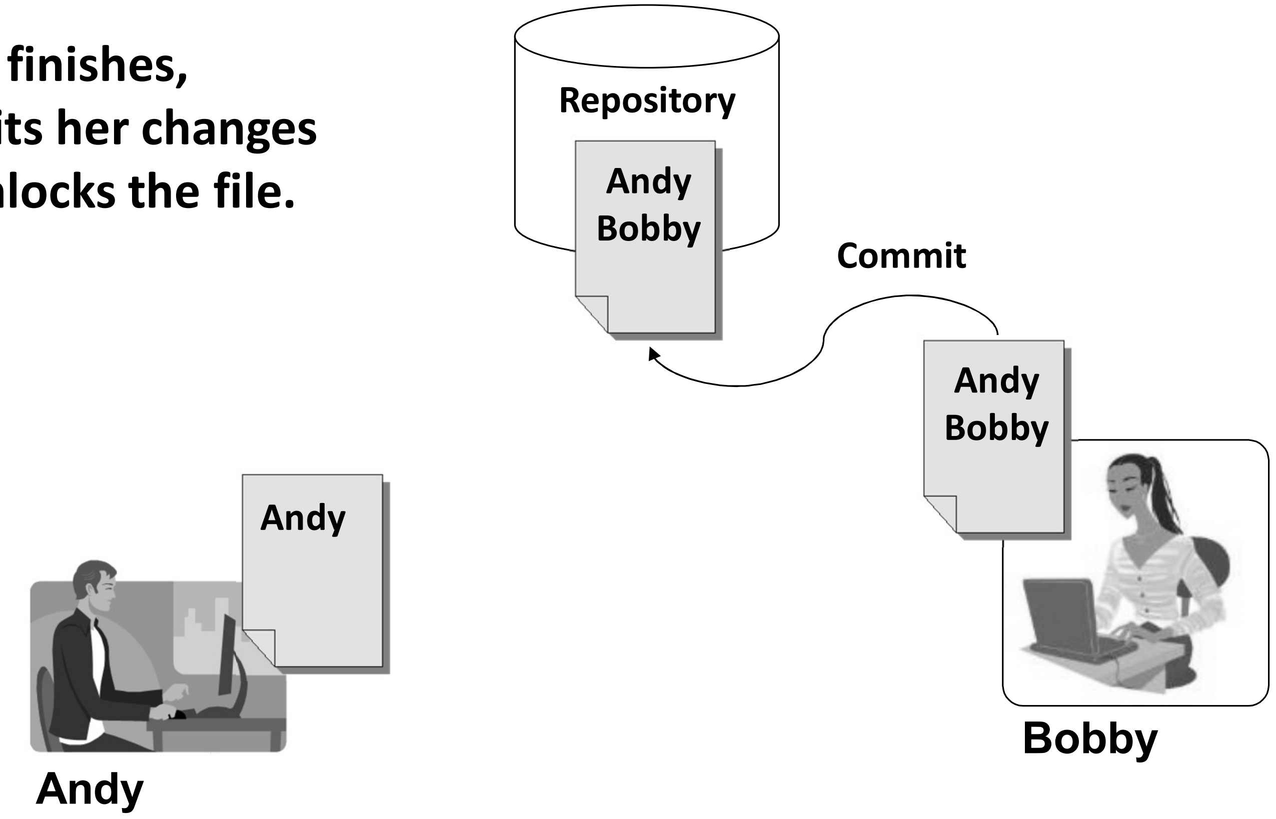
Now Bobby can take the modified file and lock it.

Bobby edits her local copy of the file.



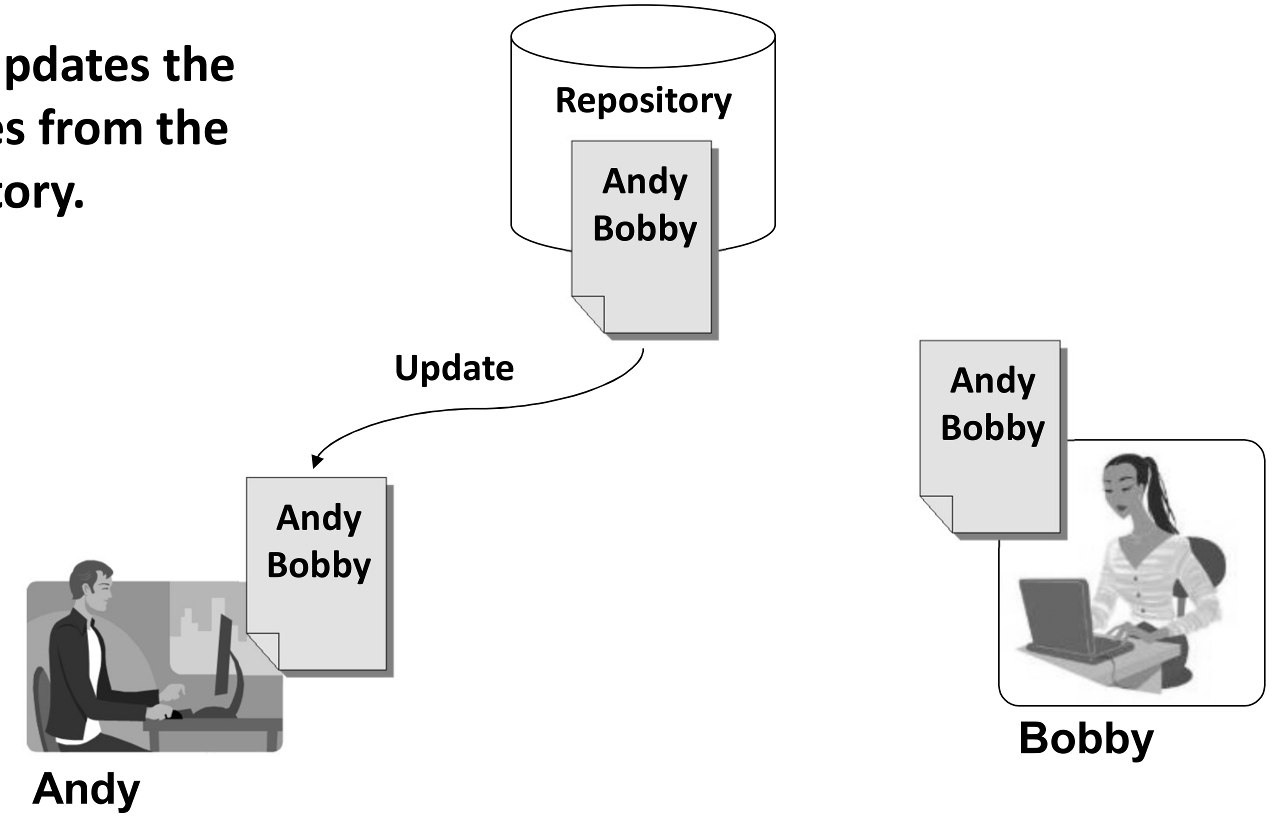
The Lock-Modify-Unlock Model (6)

Bobby finishes,
commits her changes
and unlocks the file.

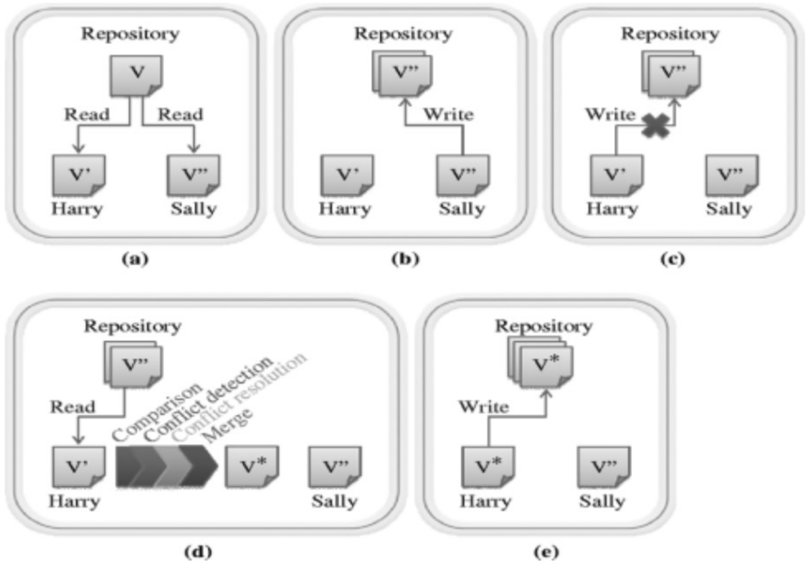


The Lock-Modify-Unlock Model (7)

Andy updates the changes from the repository.



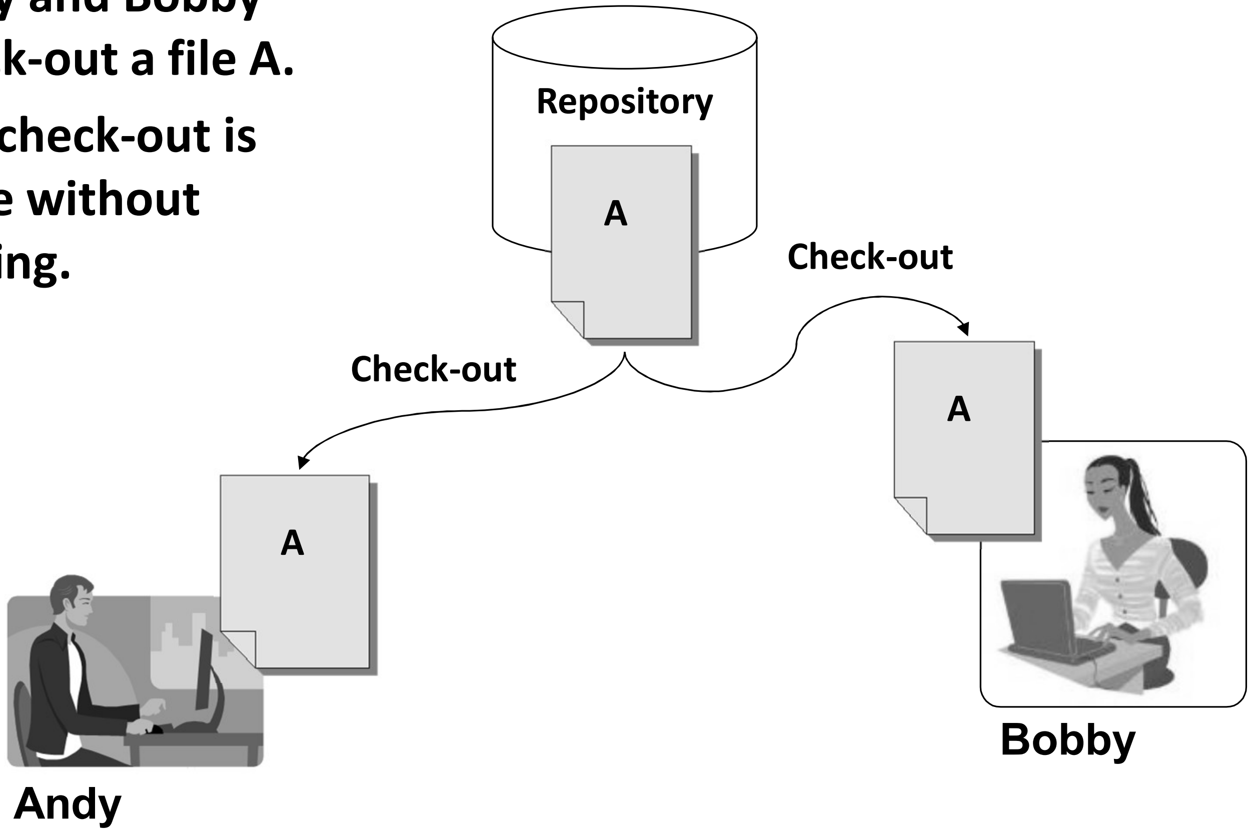
The "Copy-Modify-Merge" Model



The Copy-Modify-Merge Model (1)

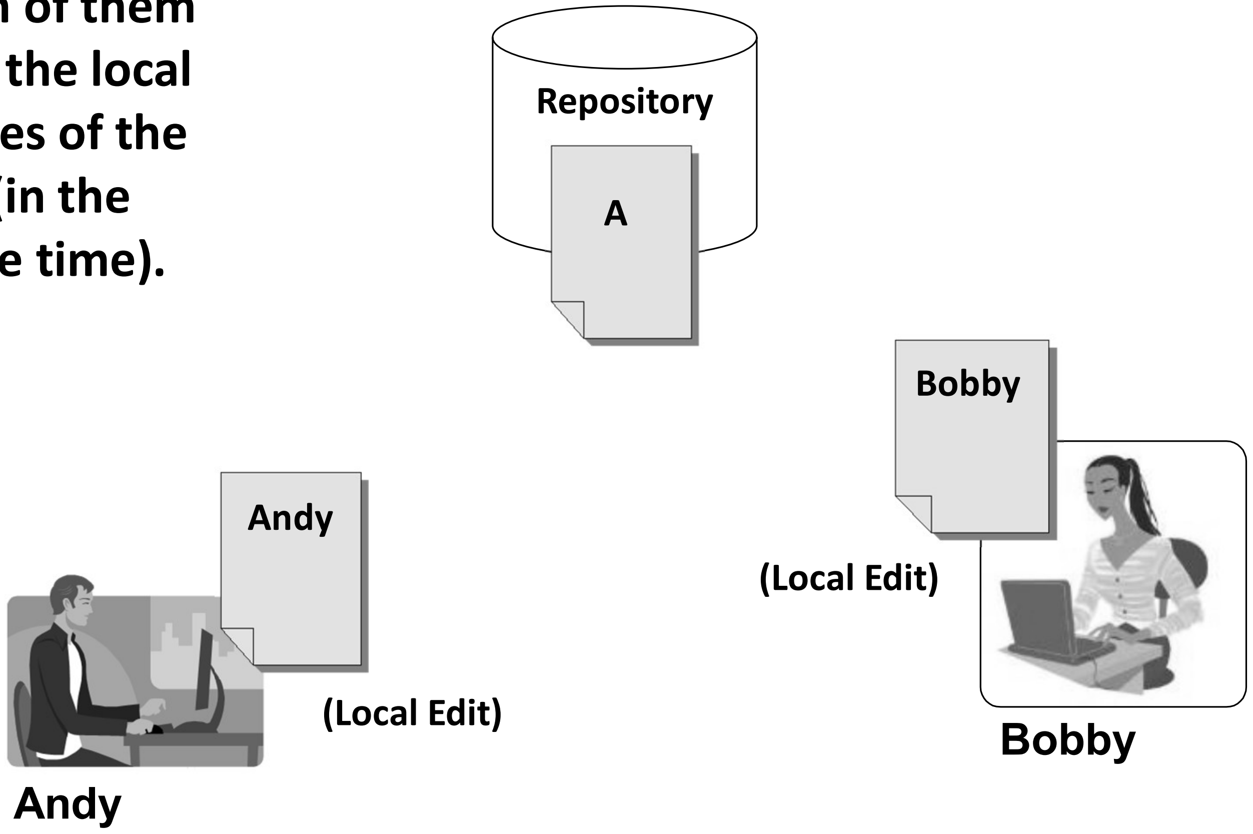
Andy and Bobby check-out a file A.

The check-out is done without locking.



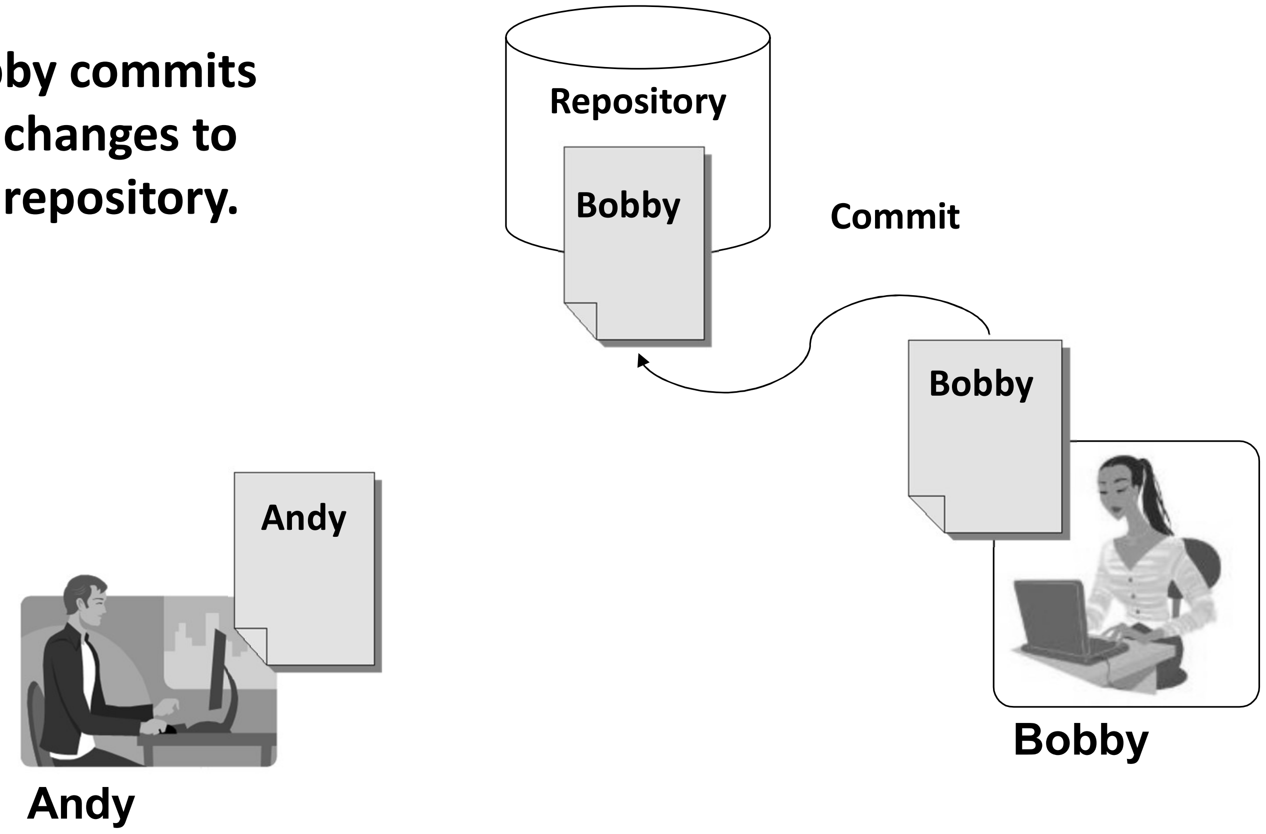
The Copy-Modify-Merge Model (2)

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



The Copy-Modify-Merge Model (3)

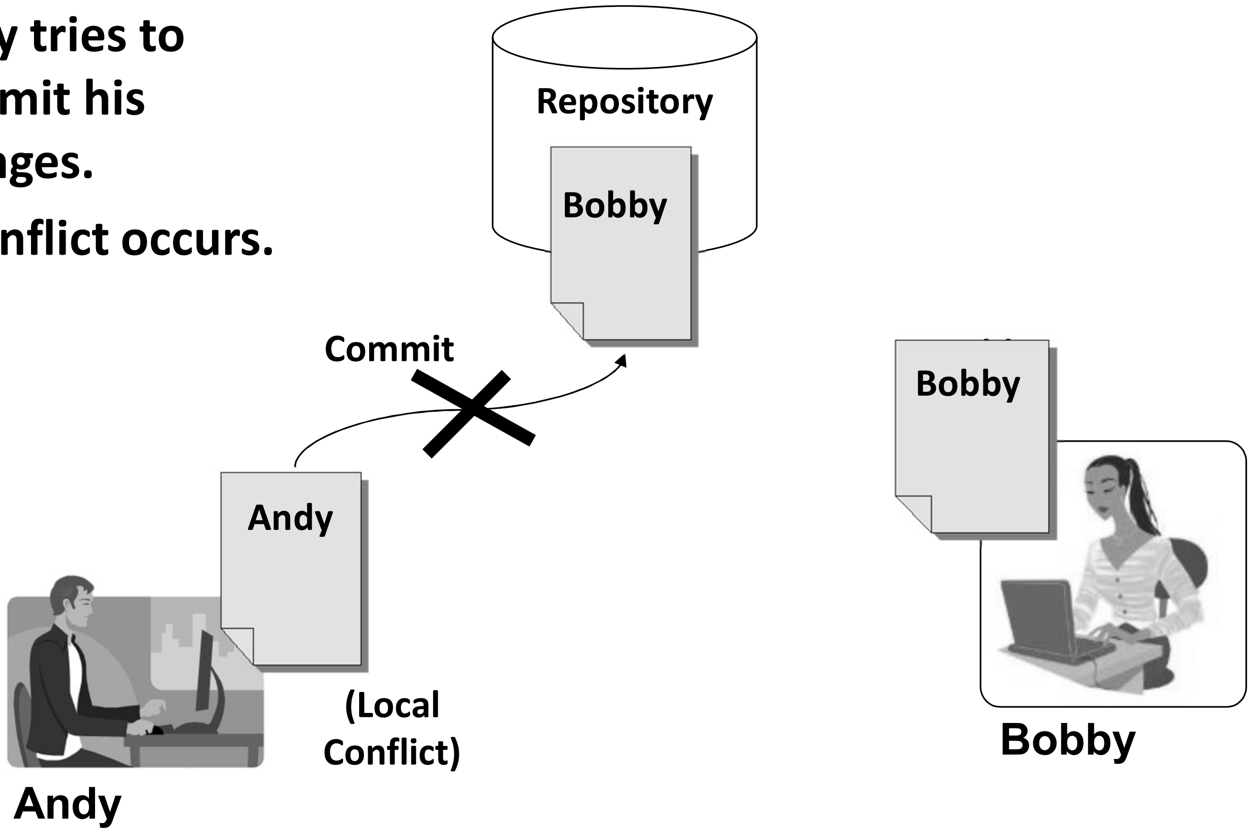
Bobby commits her changes to the repository.



The Copy-Modify-Merge Model (4)

Andy tries to commit his changes.

A conflict occurs.

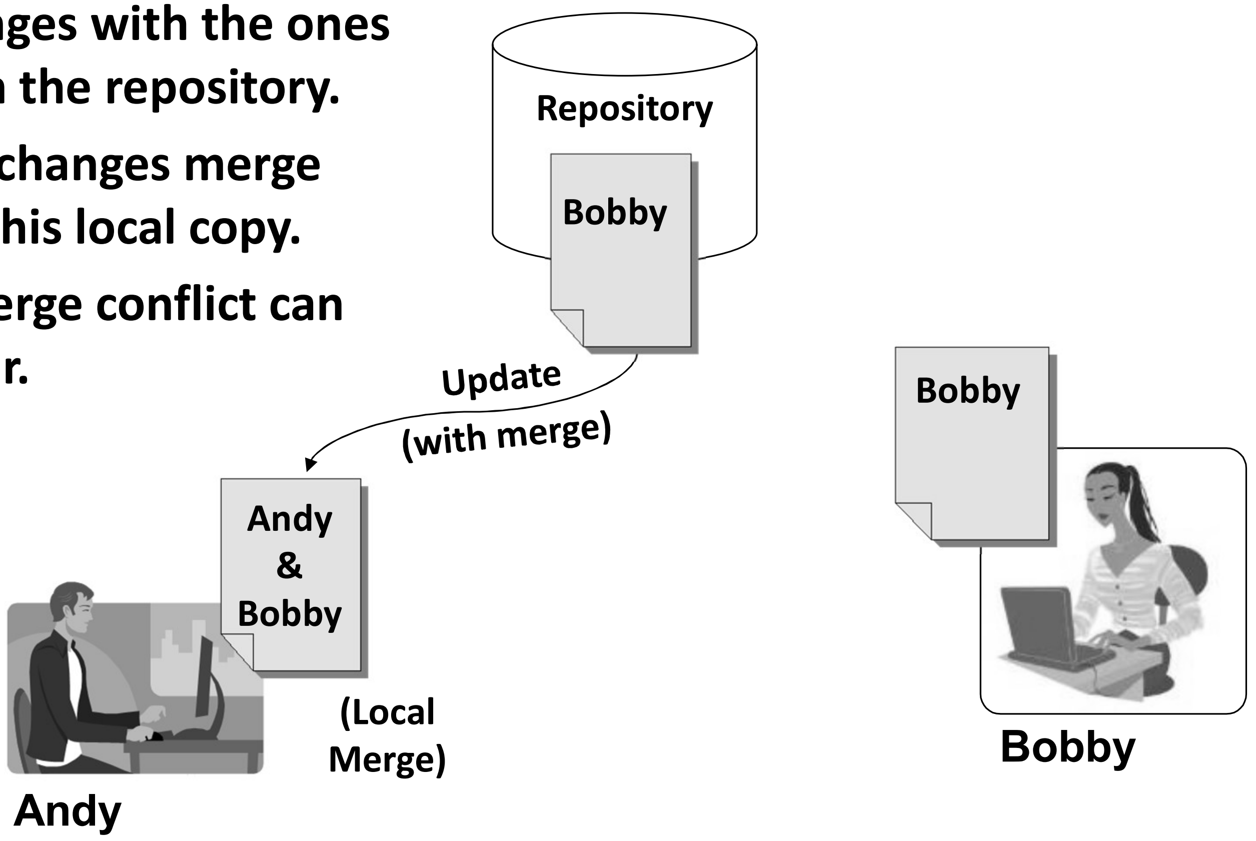


The Copy-Modify-Merge Model (5)

Andy updates his changes with the ones from the repository.

The changes merge into his local copy.

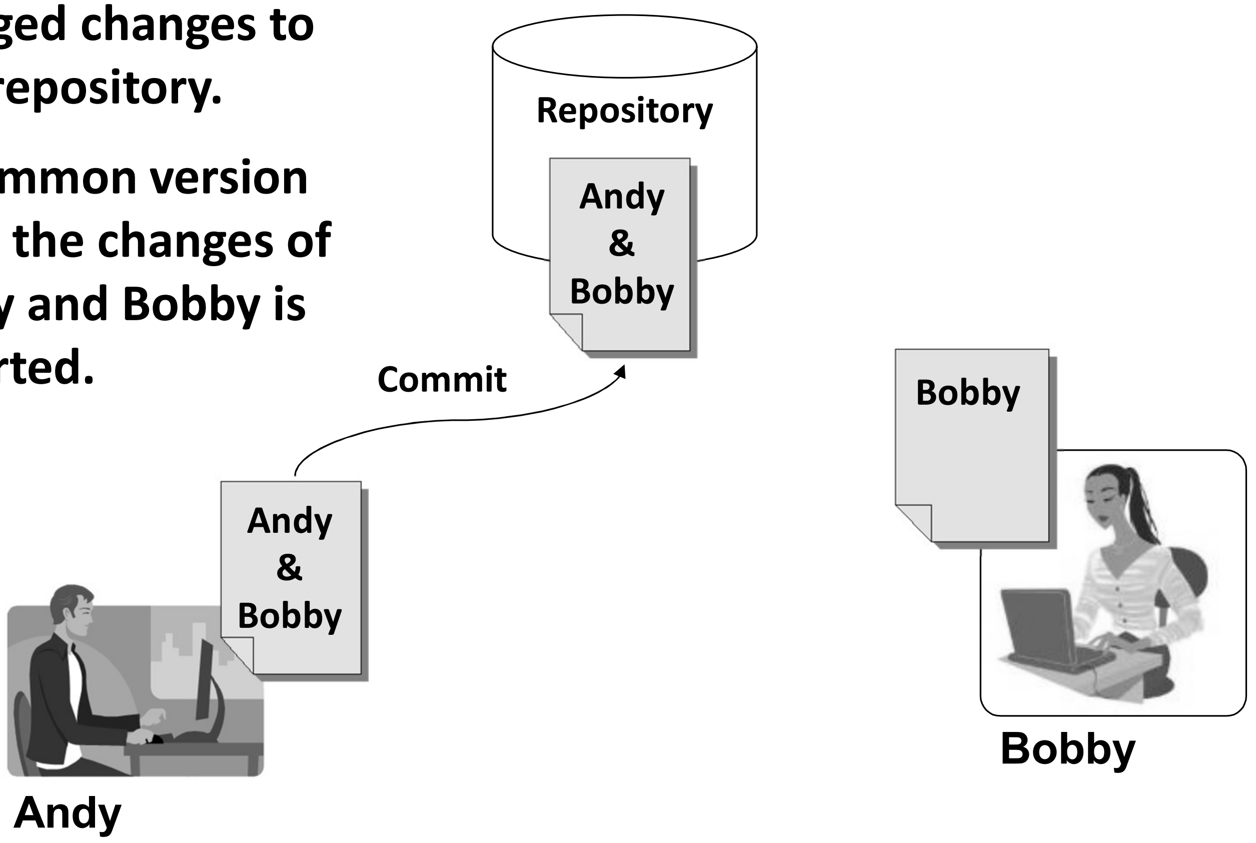
A merge conflict can occur.



The Copy-Modify-Merge Model (6)

Andy commits the merged changes to the repository.

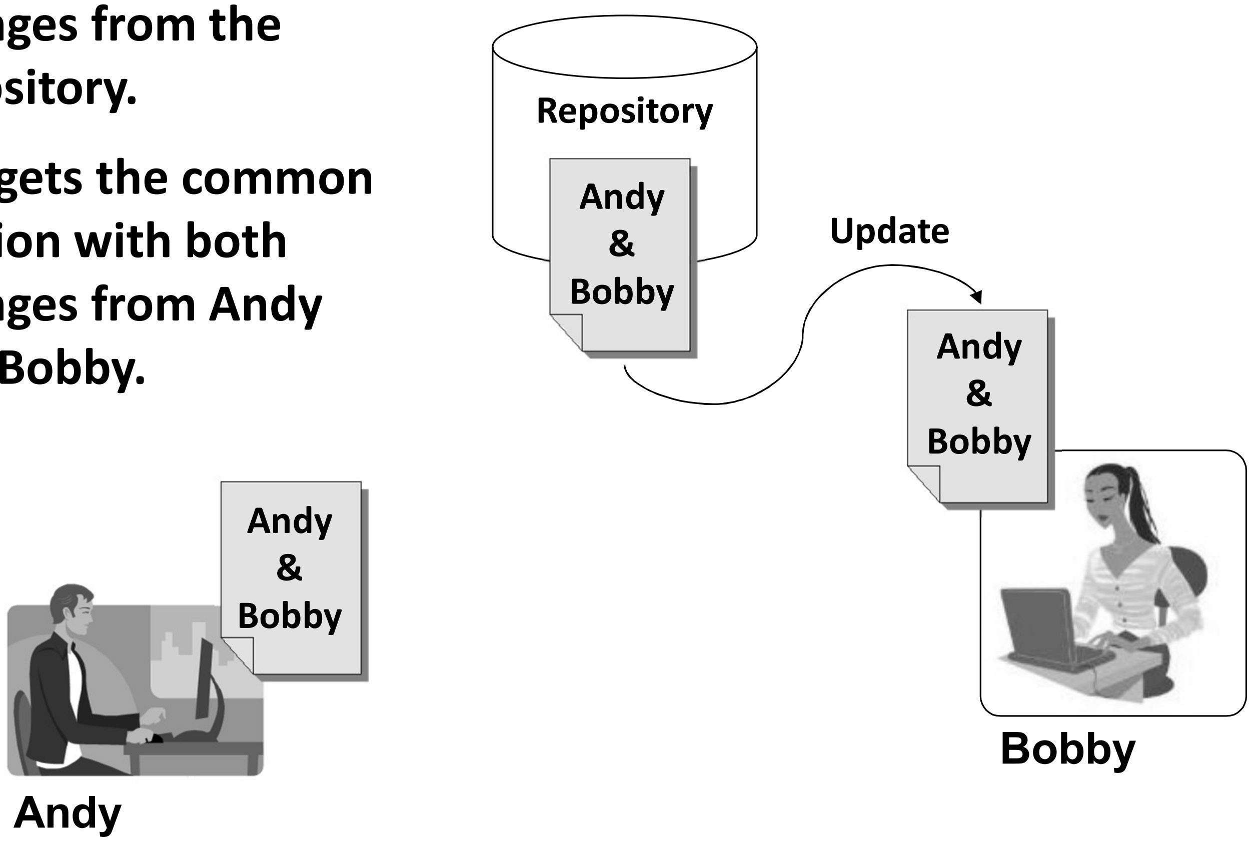
A common version with the changes of Andy and Bobby is inserted.



The Copy-Modify-Merge Model (7)

Bobby updates the changes from the repository.

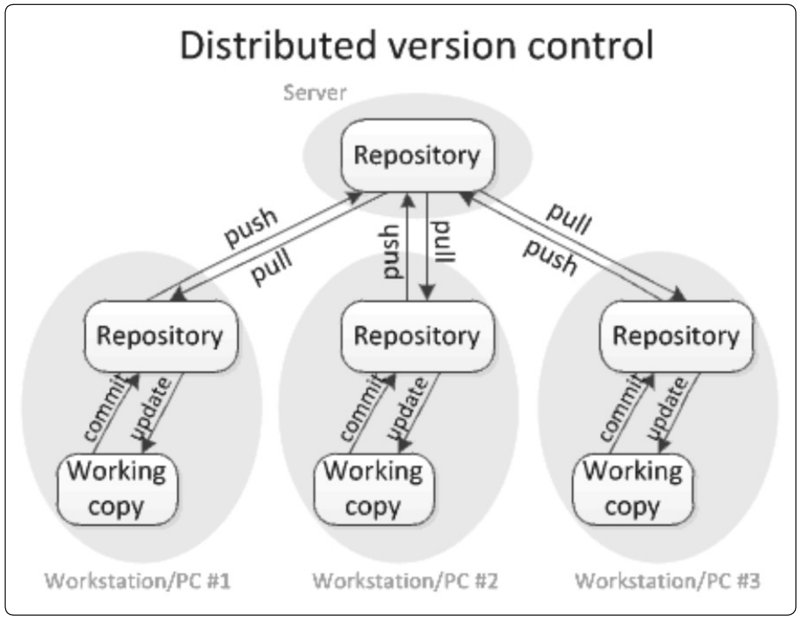
She gets the common version with both changes from Andy and Bobby.



FOUNDATION

The "Distributed Version Control" Versioning Model

git

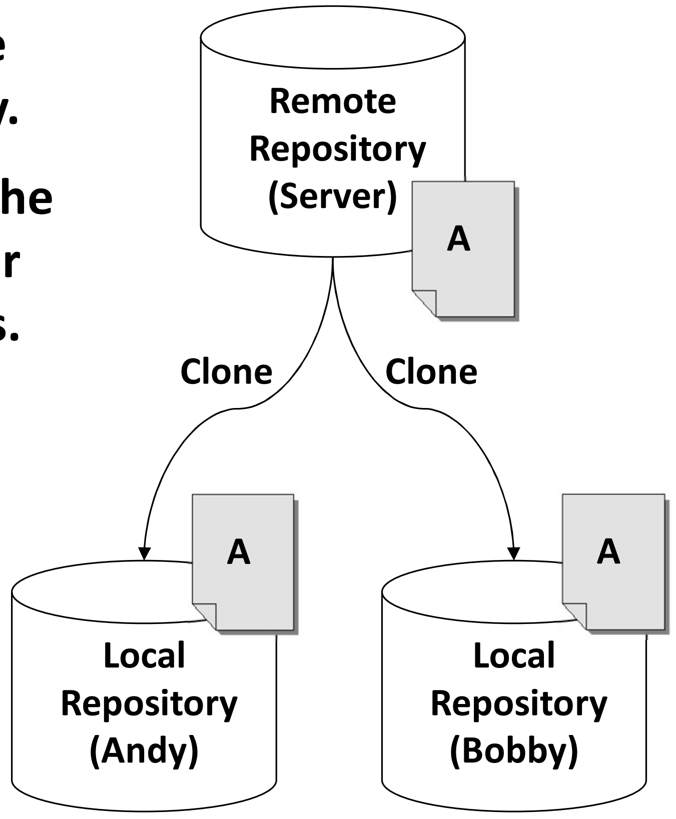


mercurial

Distributed Version Control (1)

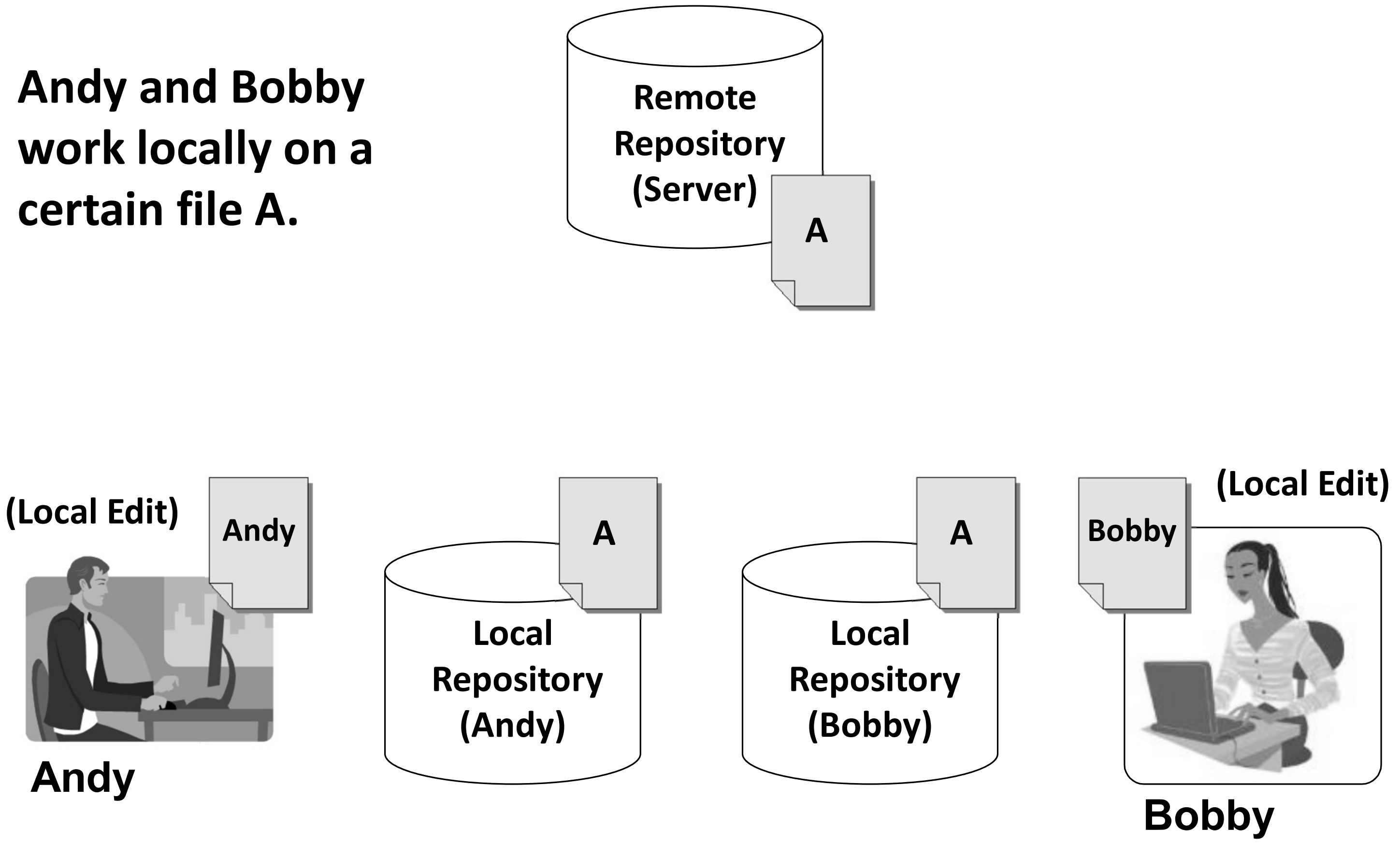
Andy and Bobby clone the remote repository locally.

They both have the same files in their local repositories.



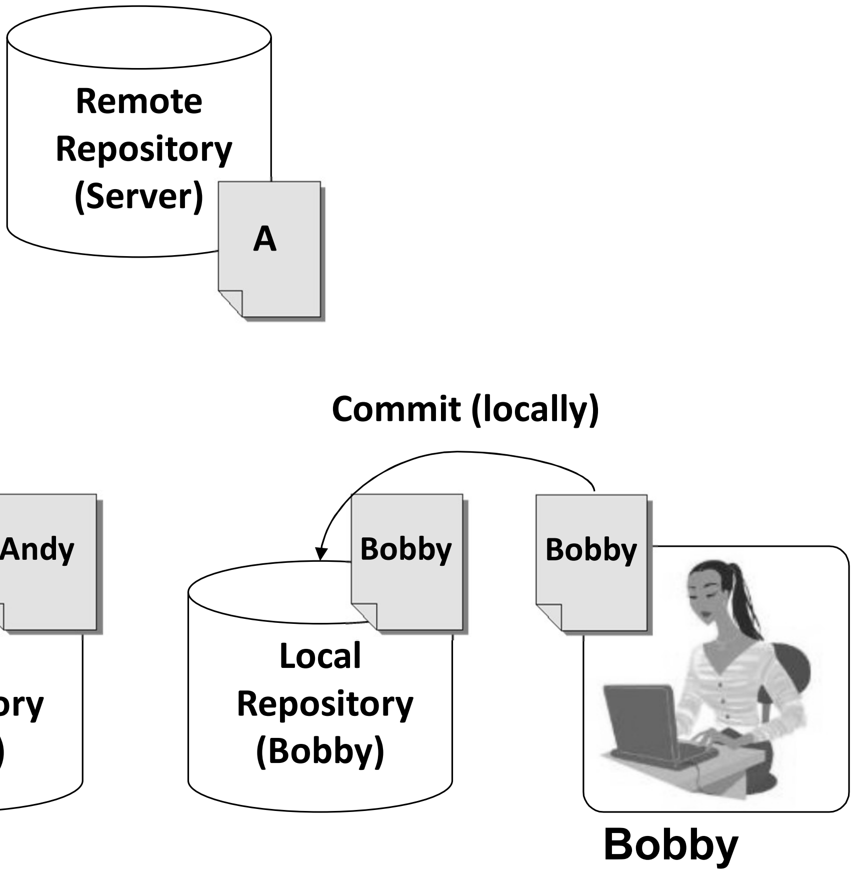
Distributed Version Control (2)

Andy and Bobby work locally on a certain file A.



Distributed Version Control (3)

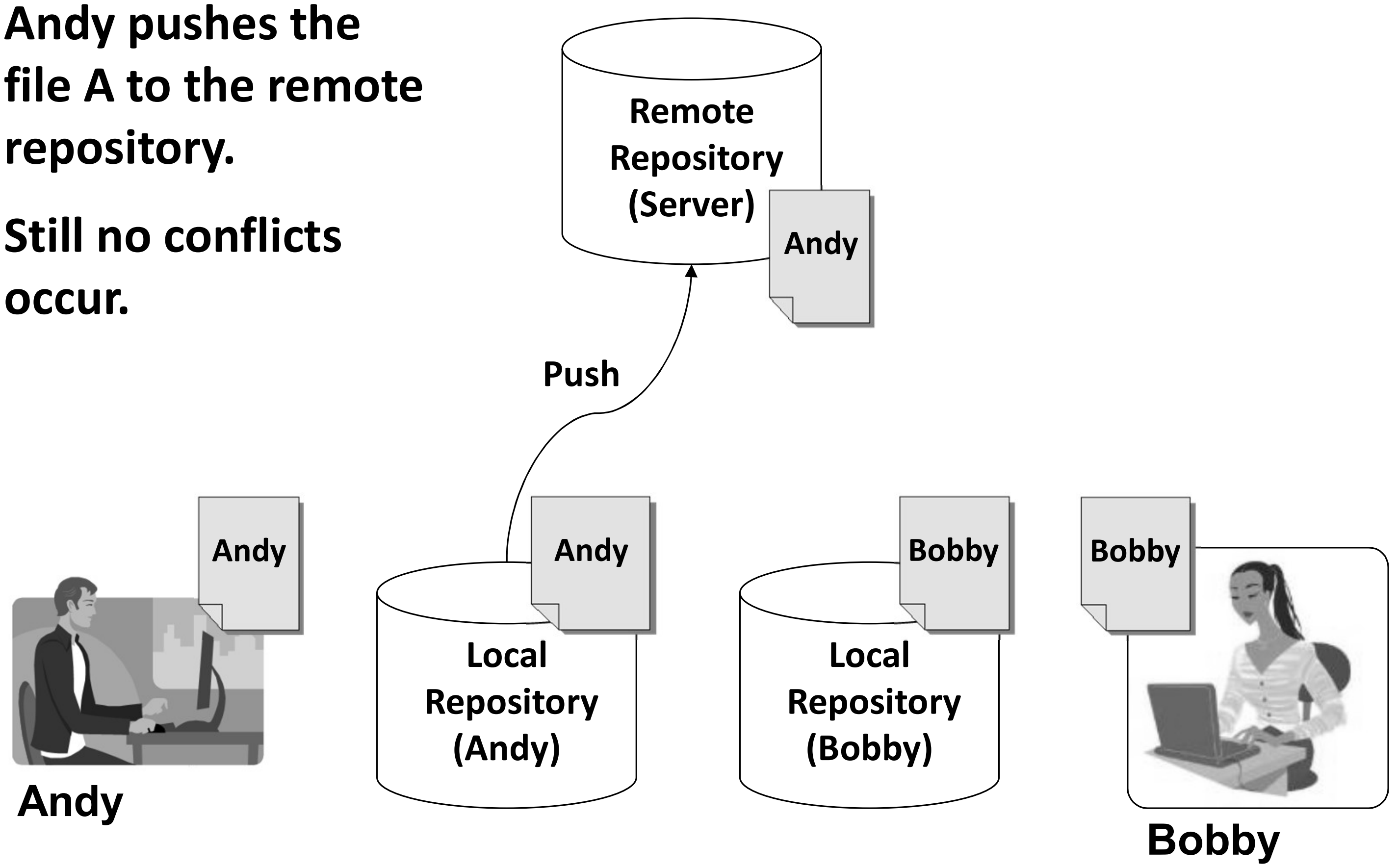
Andy and Bobby commit locally the modified file A into their local repositories.



Distributed Version Control (4)

Andy pushes the file A to the remote repository.

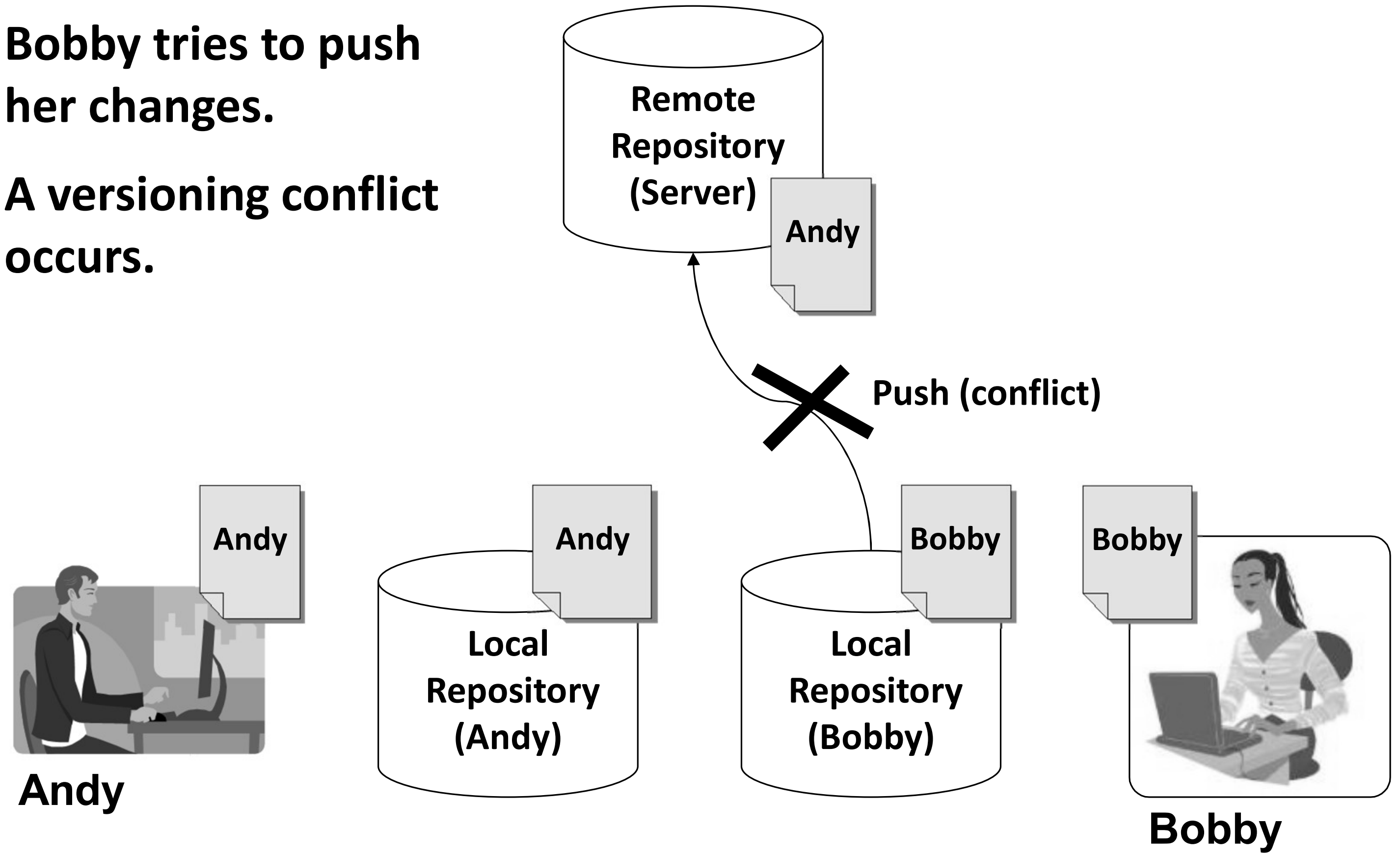
Still no conflicts occur.



Distributed Version Control (5)

Bobby tries to push her changes.

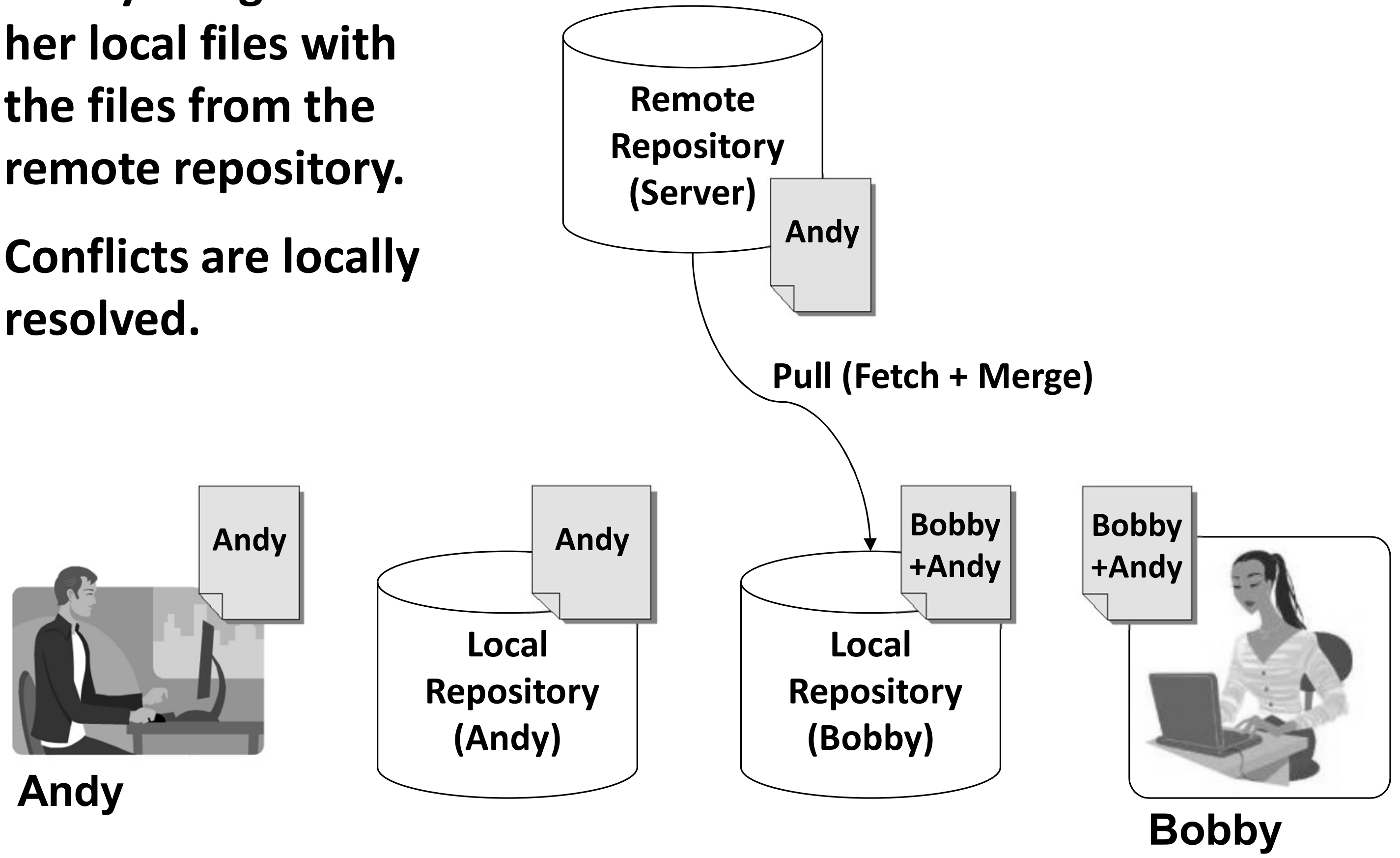
A versioning conflict occurs.



Distributed Version Control (6)

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

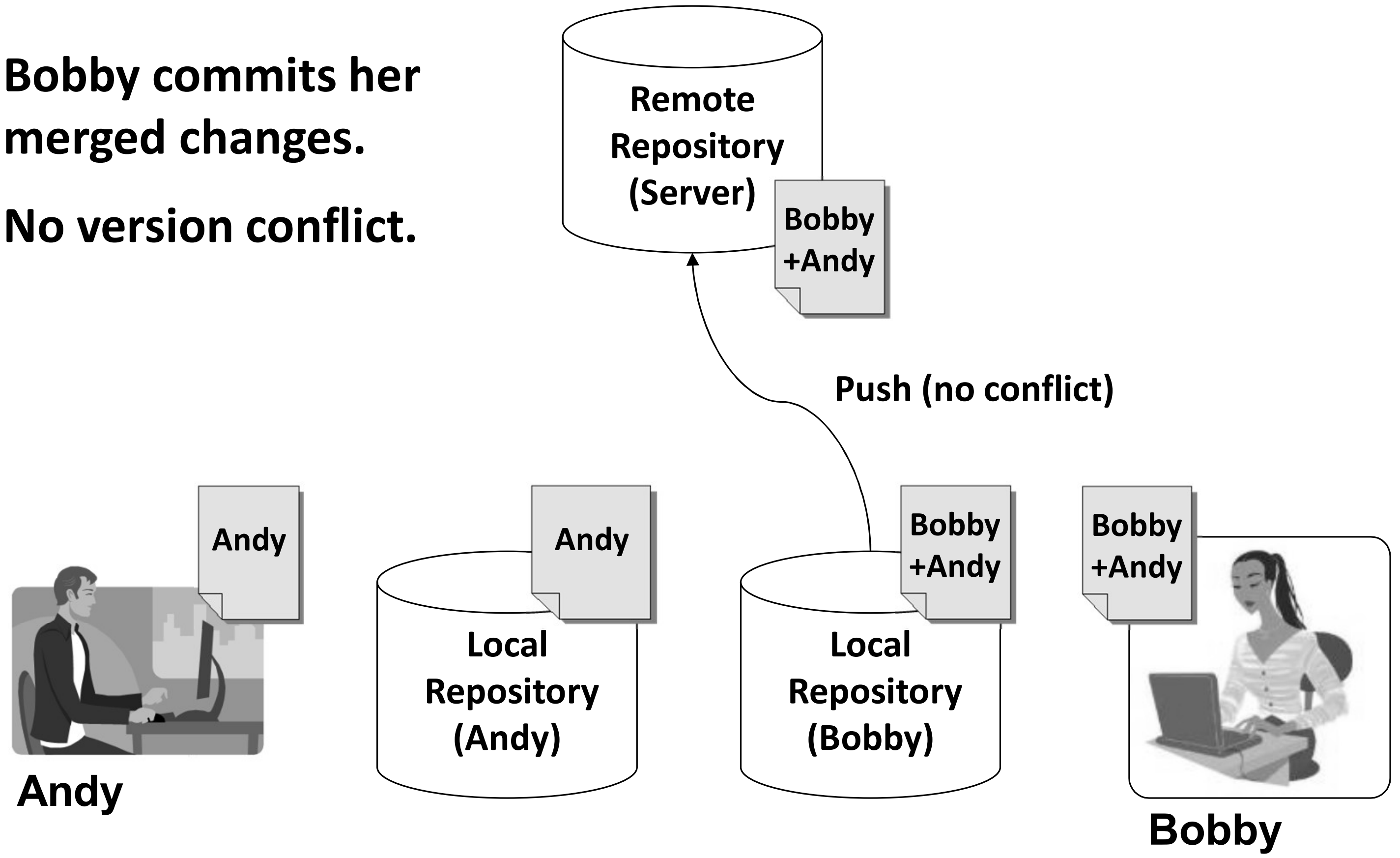
Conflicts are locally resolved.



Distributed Version Control (7)

Bobby commits her merged changes.

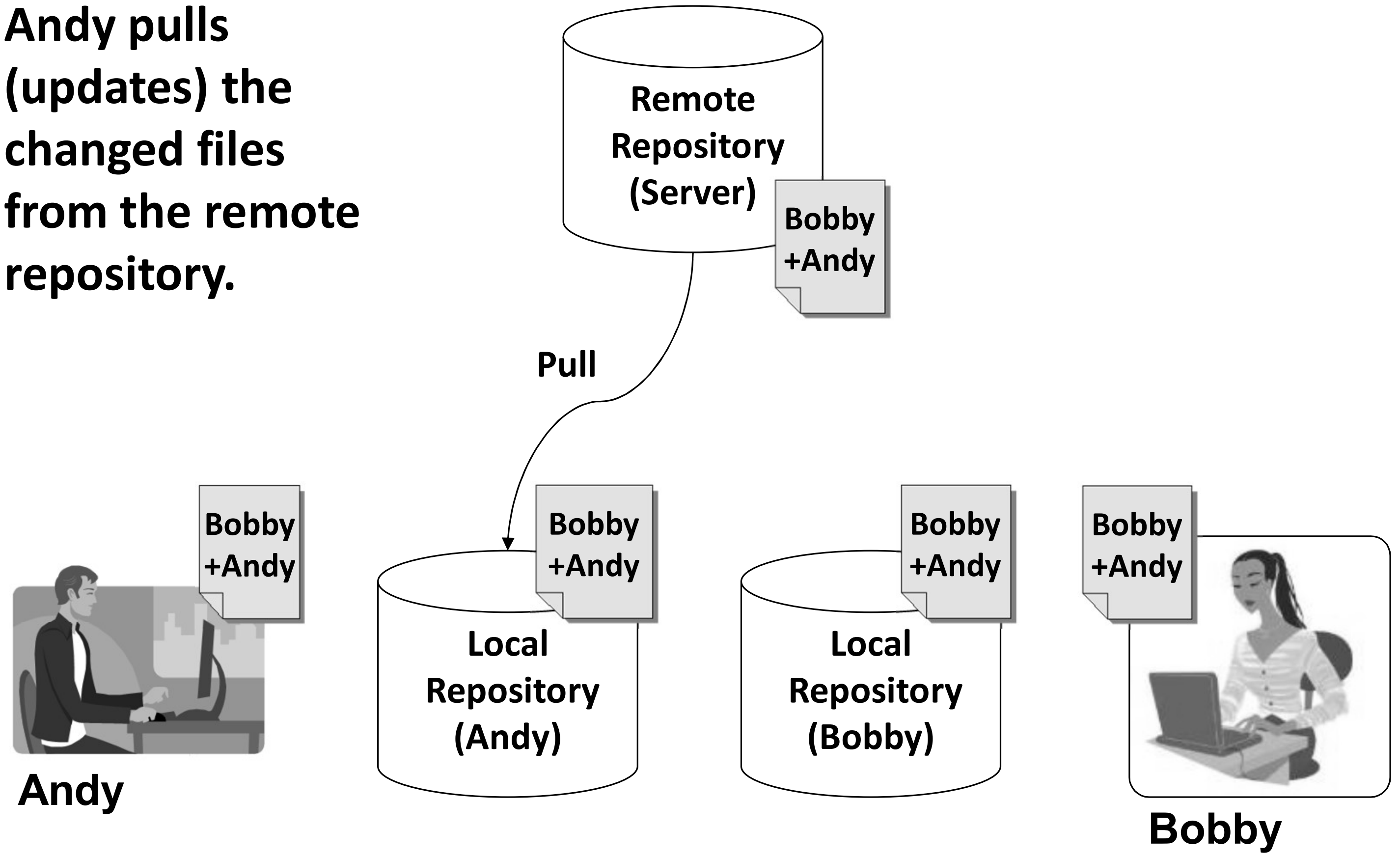
No version conflict.



Distributed Version Control (8)

FOUNDATION

Andy pulls (updates) the changed files from the remote repository.

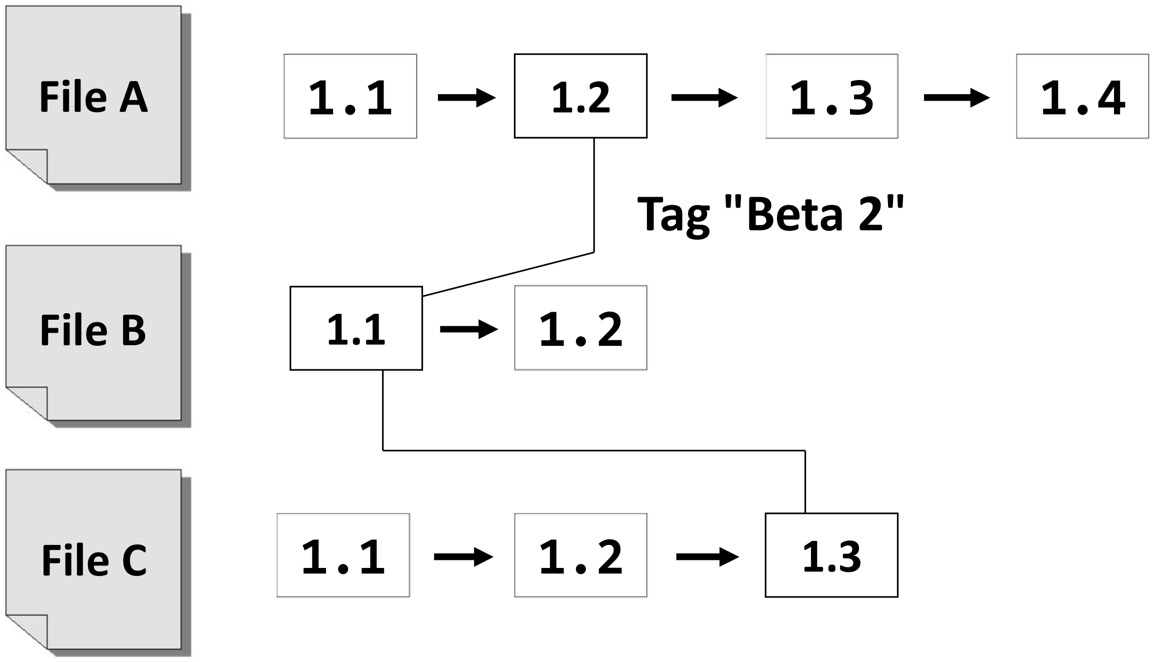


FOUNDATION

Tags and Branches

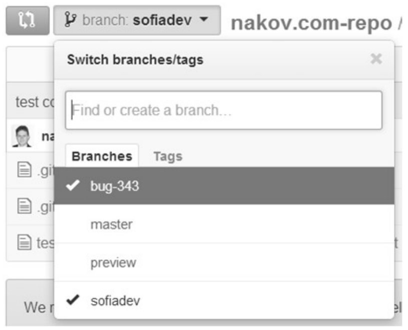
Tags

- Allows us to give a name to a group of files in a certain version



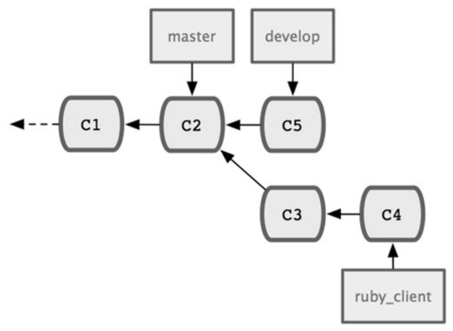
Branching

- Branching allows splitting the development line into separate branches
  - Different developers work in different branches
- Branching is suitable for:
  - Development of new feature or fix in a new version of the product (for example version 2.0)
    - Features are invisible in the main development line
    - Until merged with it
  - You can still make changes in the older version (for example version 1.0.1)

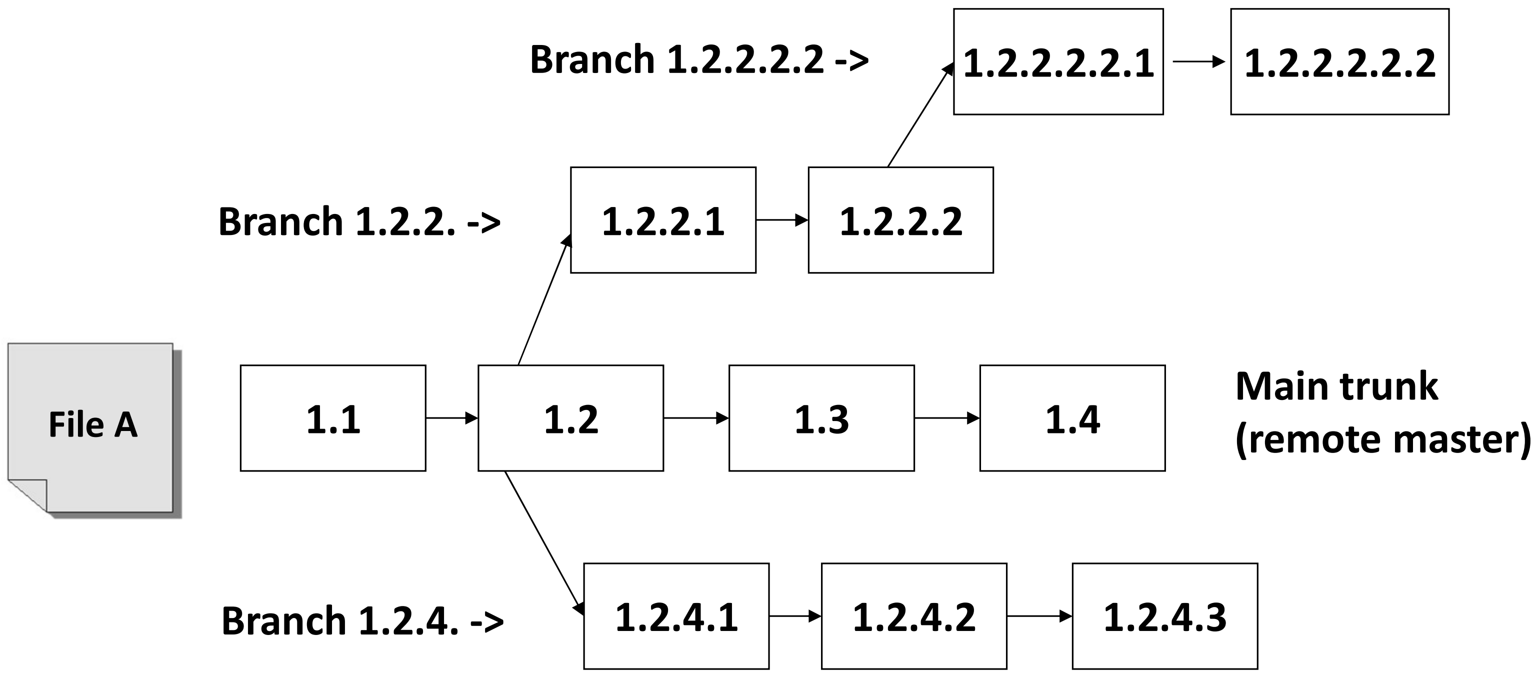


Merging Branches

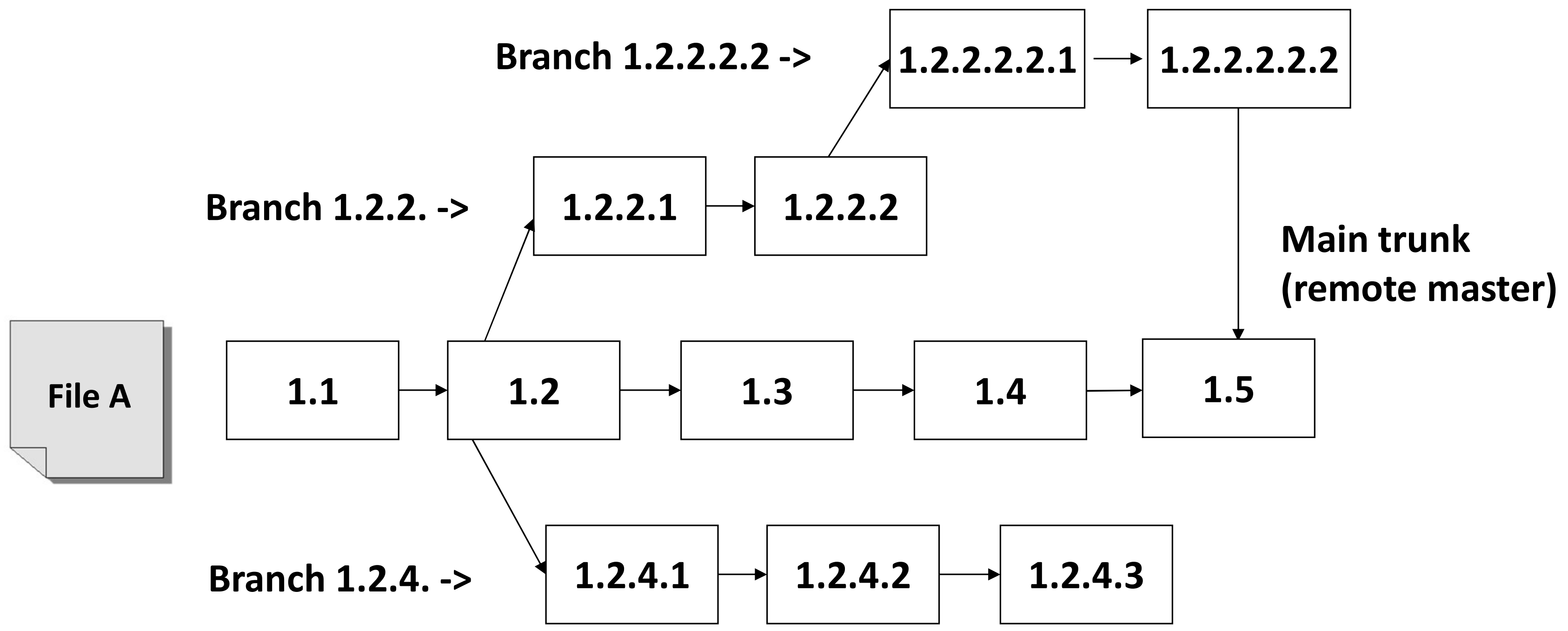
- Some companies work in separate branches
  - For each new feature / fix / task
- Once a feature / fix / task is completed
  - It is tested locally and committed in its branch
- Finally it is merged into the main development line
  - Merging is done locally
  - Conflicts are resolved locally
  - If the merge is tested and works well, it is integrated back in the main development line



Branching – Example



Merging Branches – Example


Branch 1.2.2.2.2 ->
1.2.2.2.2.1
1.2.2.2.2.2
Branch 1.2.2. ->
1.2.2.1
1.2.2.2
Main trunk
(remote master)
File A
1.1
1.2
1.3
1.4
1.5
Branch 1.2.4. ->
1.2.4.1
1.2.4.2
1.2.4.3