IT4490 - SOFTWARE DESIGN AND CONSTRUCTION

1. VERSION CONTROL

1

Outline

2

Why version control? (1/2)

- 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?

3

Why version control? (2/2)

- 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?

4

Version control for teams (1/2)

- 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?

5

Version control for teams (2/2)

- 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?

6

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

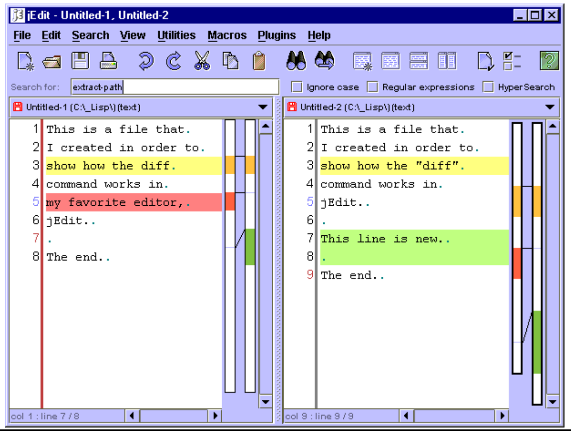
7

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

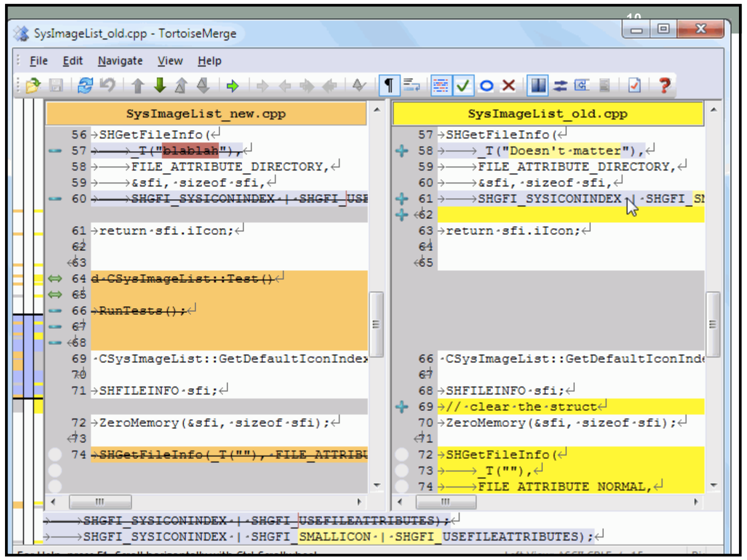
8

Tools: jDiff


jEdit - Untitled-1, Untitled-2
File Edit Search View Utilities Macros Plugins Help
Search for: extract-path
Ignore case
Regular expressions
HyperSearch
Untitled-1 (C:_Lisp\) (text)
Untitled-2 (C:_Lisp\) (text)
col 1 : line 7 / 8
col 9 : line 9 / 9

```
This is a file that.
I created in order to.
show how the diff.
command works in.
my favorite editor,.
jEdit..
.
The end..
```

```
This is a file that.
I created in order to.
show how the "diff".
command works in.
jEdit..
.
This line is new..
.
The end..
```


SysImageList_old.cpp - TortoiseMerge
File Edit Navigate View Help
SysImageList_new.cpp
SysImageList_old.cpp

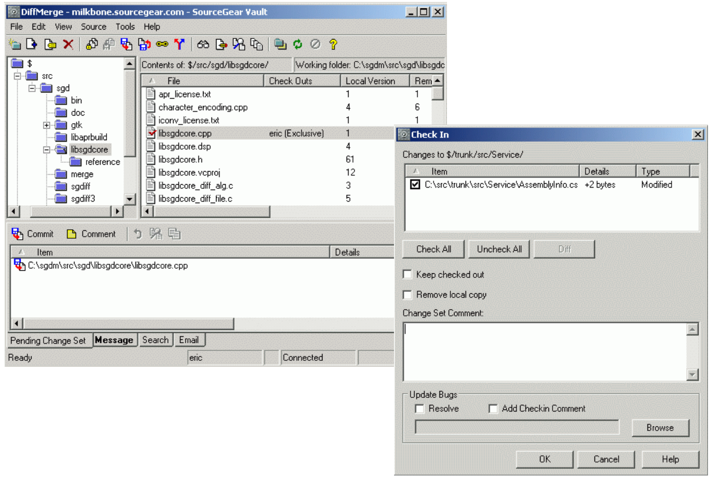
```
SHGetFileInfo(
        _T("blablah"),
        FILE_ATTRIBUTE_DIRECTORY,
        &sfi, sizeof sfi,
        SHGFI_SYSICONINDEX | SHGFI_USE

return sfi.iIcon;


d CSysImageList::Test()

RunTests();


CSysImageList::GetDefaultIconIndex

SHFILEINFO sfi;

ZeroMemory(&sfi, sizeof sfi);

SHGetFileInfo(_T(""), FILE_ATTRIBU
```

```
SHGetFileInfo(
        _T("Doesn't matter"),
        FILE_ATTRIBUTE_DIRECTORY,
        &sfi, sizeof sfi,
        SHGFI_SYSICONINDEX | SHGFI_SM

return sfi.iIcon;


CSysImageList::GetDefaultIconInde

SHFILEINFO sfi;
// clear the struct
ZeroMemory(&sfi, sizeof sfi);

SHGetFileInfo(
        _T(""),
        FILE_ATTRIBUTE_NORMAL,
```

```
        SHGFI_SYSICONINDEX | SHGFI_USEFILEATTRIBUTES);
        SHGFI_SYSICONINDEX | SHGFI_SMALLICON | SHGFI_USEFILEATTRIBUTES);
```

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



Contents of: $/src/sgd/libsgdcore/ Working folder: C:\sgdm\src\sgd\libsgdc

File	Check Outs	Local Version	Rem
apr_license.txt		1	1
character_encoding.cpp		4	6
iconv_license.txt		1	1
libsgdcore.cpp	eric (Exclusive)	1	
libsgdcore.dsp		4	
libsgdcore.h		61	
libsgdcore.vcproj		12	
libsgdcore_diff_alg.c		3	
libsgdcore_diff_file.c		5	

Item	Details
C:\sgdm\src\sgd\libsgdcore\libsgdcore.cpp	

Ready | eric | Connected

Changes to $/trunk/src/Service/

Item	Details	Type
☑ C:\src\trunk\src\Service\AssemblyInfo.cs	+2 bytes	Modified

☐ Keep checked out

☐ Remove local copy

Change Set Comment:

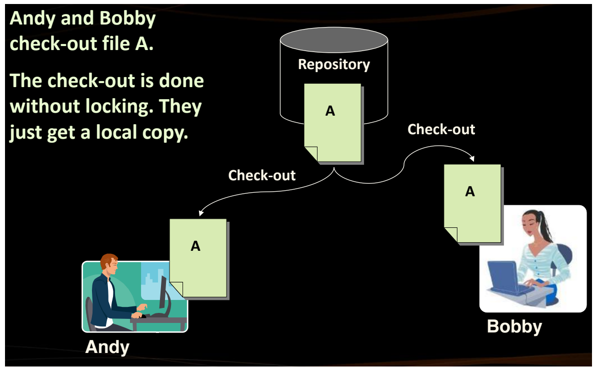
☐ Resolve ☐ Add Checkin Comment

Outline

2. Versioning Models

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

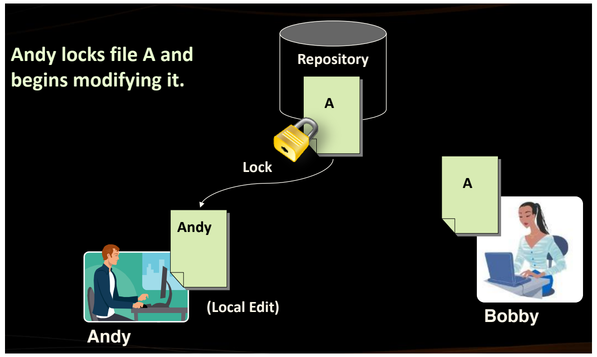
2.1. The Lock-Modify-Unlock Model



Andy and Bobby check-out file A.

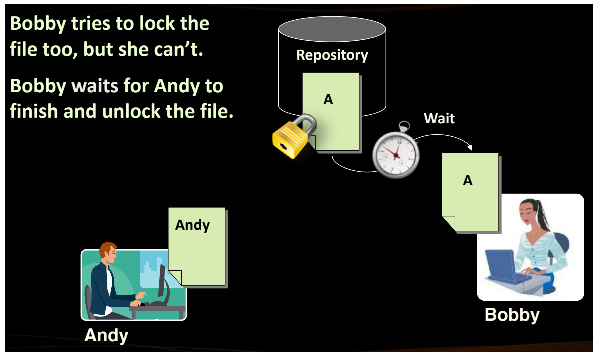
The check-out is done without locking. They just get a local copy.

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



Andy locks file A and begins modifying it.

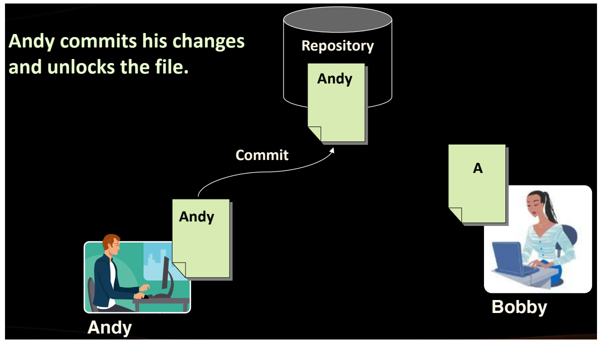
2.1. The Lock-Modify-Unlock Model (3)


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

Bobby waits for Andy to finish and unlock the file.

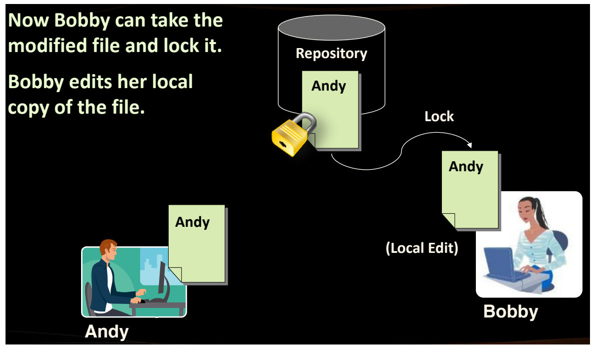
17

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


Andy commits his changes and unlocks the file.

18

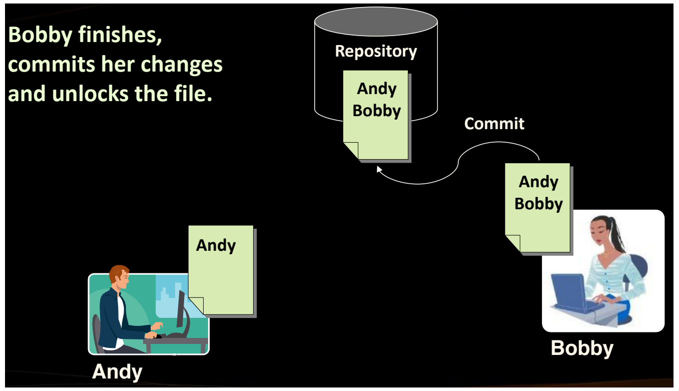
2.1. The Lock-Modify-Unlock Model (5)


Now Bobby can take the modified file and lock it.

Bobby edits her local copy of the file.

19

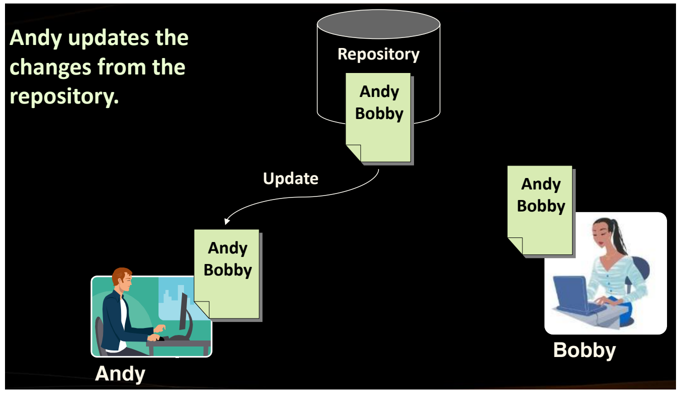
2.1. The Lock-Modify-Unlock Model (6)


Bobby finishes, commits her changes and unlocks the file.

20

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

Andy updates the changes from the repository.

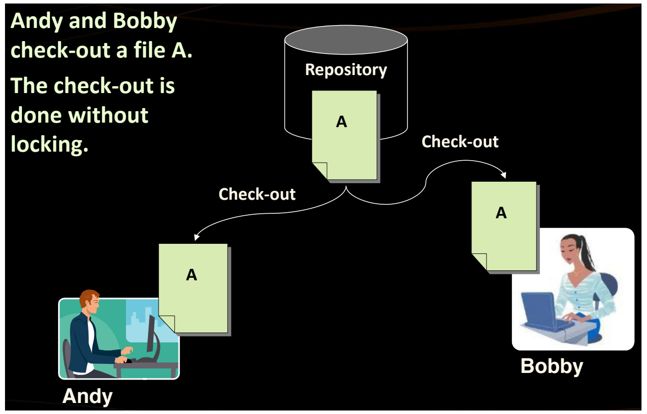


21

2.2. The Copy-Modify-Merge Model

Andy and Bobby check-out a file A.

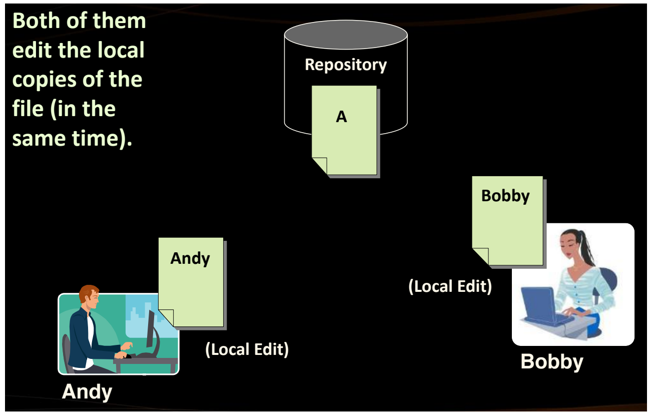
The check-out is done without locking.



22

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

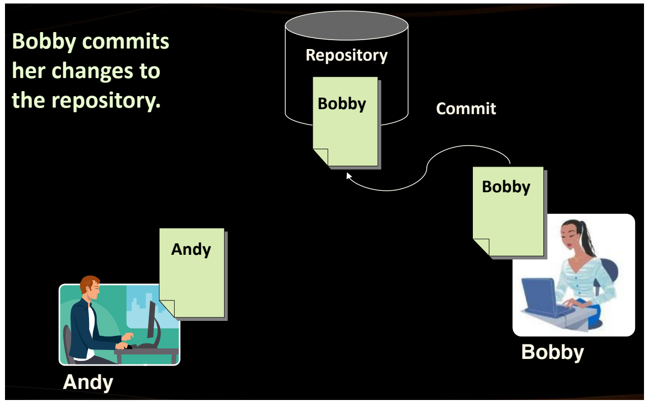
Both of them edit the local copies of the file (in the same time).



23

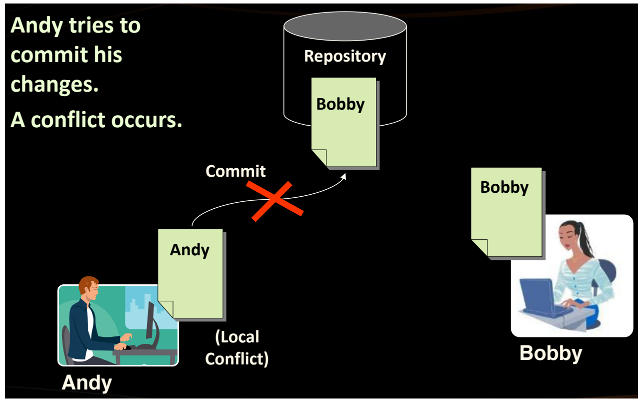
2.2. The Copy-Modify-Merge Model (3)

Bobby commits her changes to the repository.



24

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

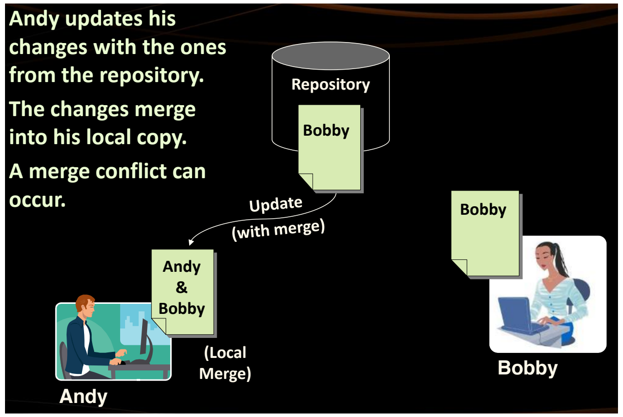


Andy tries to commit his changes.

A conflict occurs.

25

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



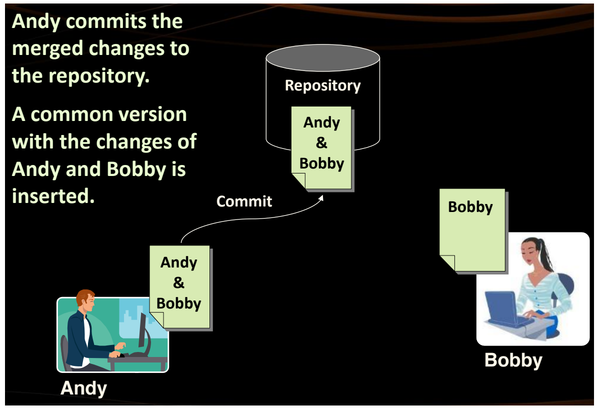
Andy updates his changes with the ones from the repository.

The changes merge into his local copy.

A merge conflict can occur.

26

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

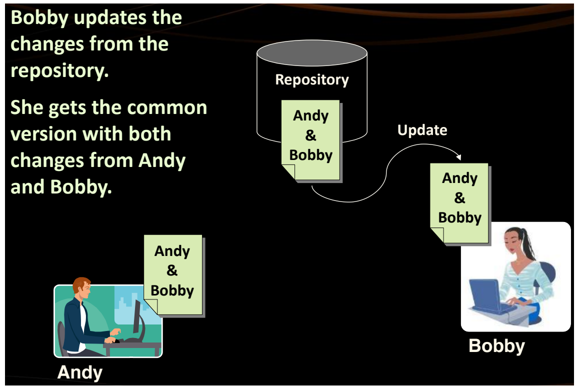


Andy commits the merged changes to the repository.

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

27

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



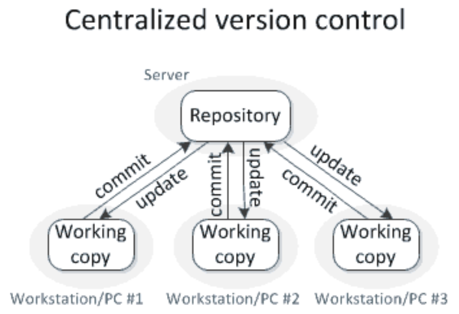
Bobby updates the changes from the repository.

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

28

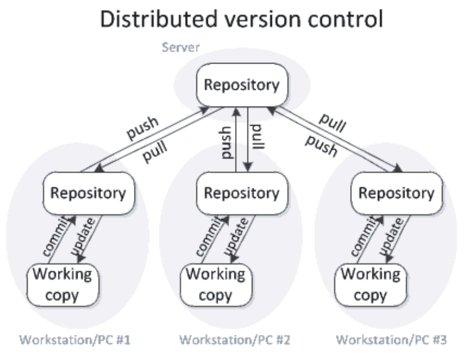
2.3. Distributed version control

- Compared to Central version control
  - Only one repository



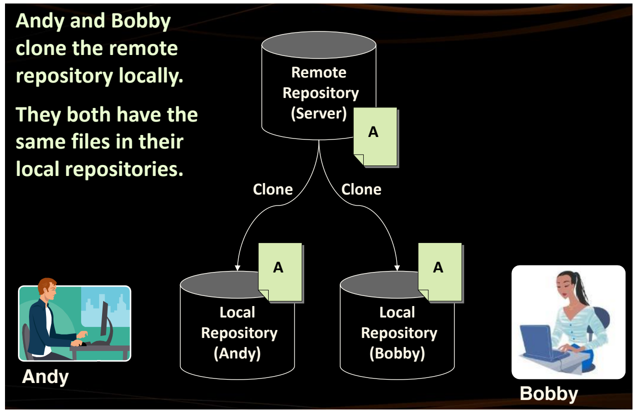
29

Distributed Version Control



30

Distributed Version Control (1)

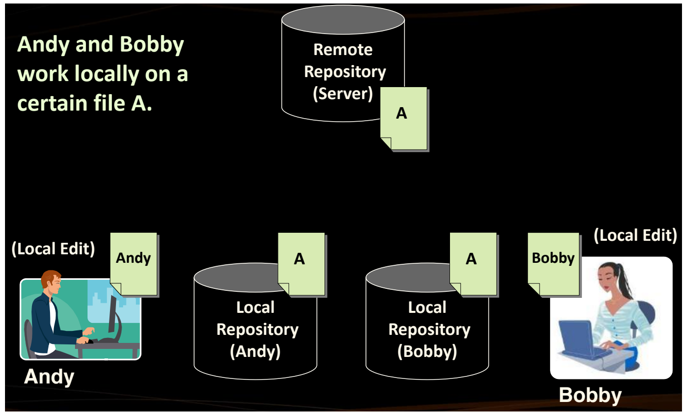


Andy and Bobby clone the remote repository locally.

They both have the same files in their local repositories.

31

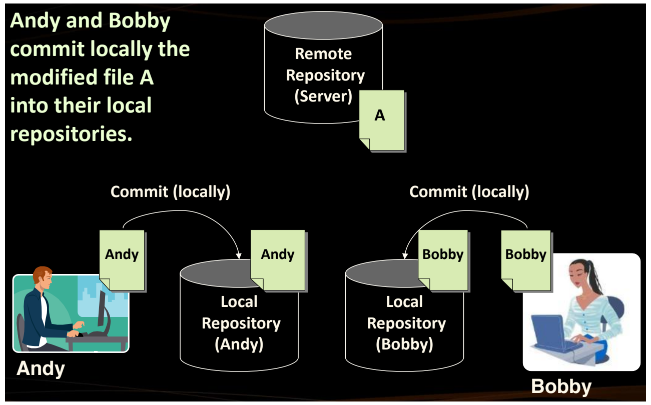
Distributed Version Control (2)



Andy and Bobby work locally on a certain file A.

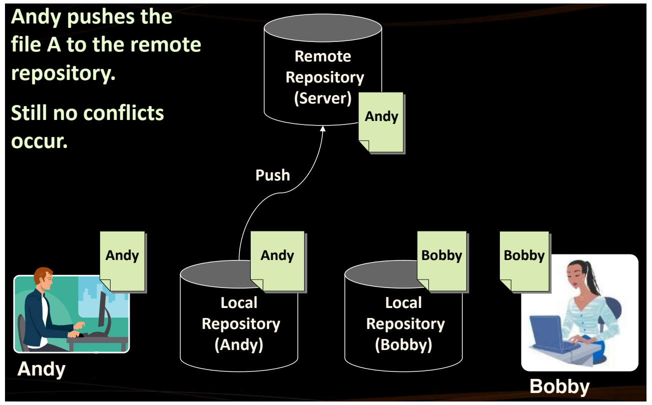
32

Distributed Version Control (3)


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

33

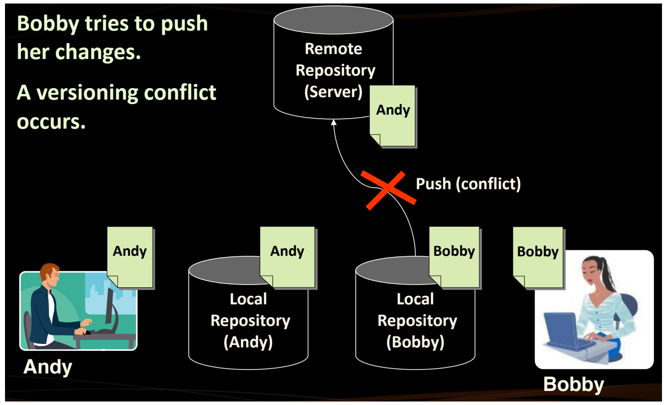
Distributed Version Control (4)


Andy pushes the file A to the remote repository.

Still no conflicts occur.

34

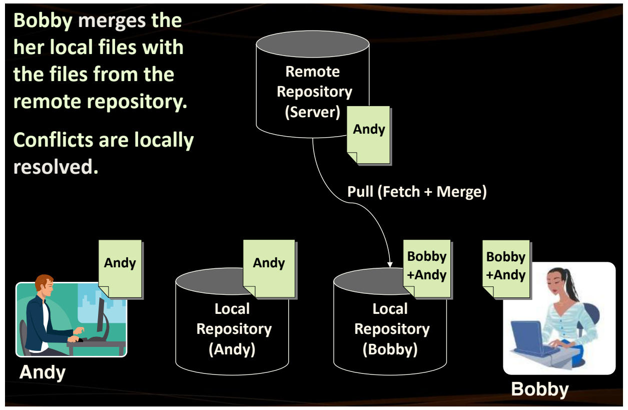
Distributed Version Control (5)


Bobby tries to push her changes.

A versioning conflict occurs.

35

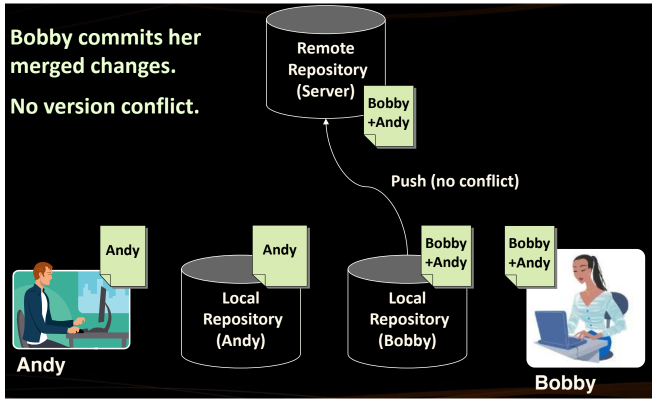
Distributed Version Control (6)


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

Conflicts are locally resolved.

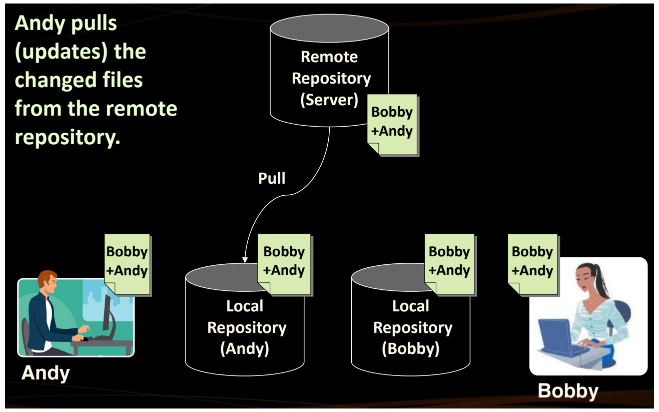
36

Distributed Version Control (7)



37

Distributed Version Control (8)



38

Outline

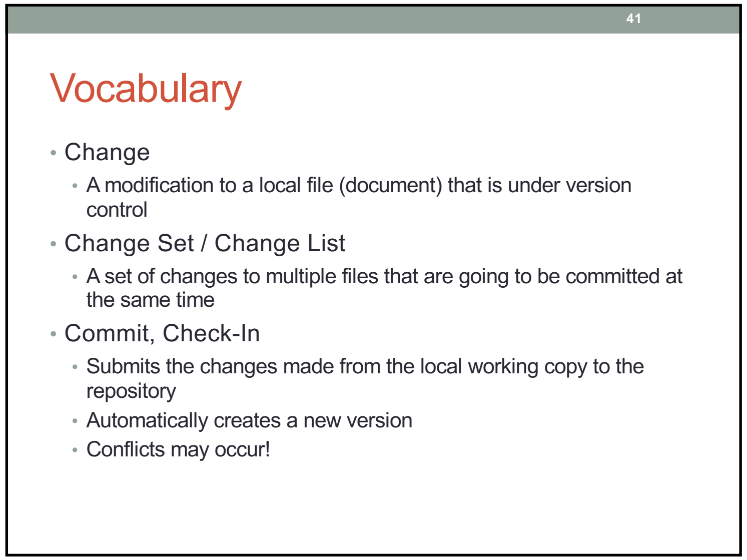
1. Introduction
2. Versioning Models
3. **Vocabulary**
4. Tools

39

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

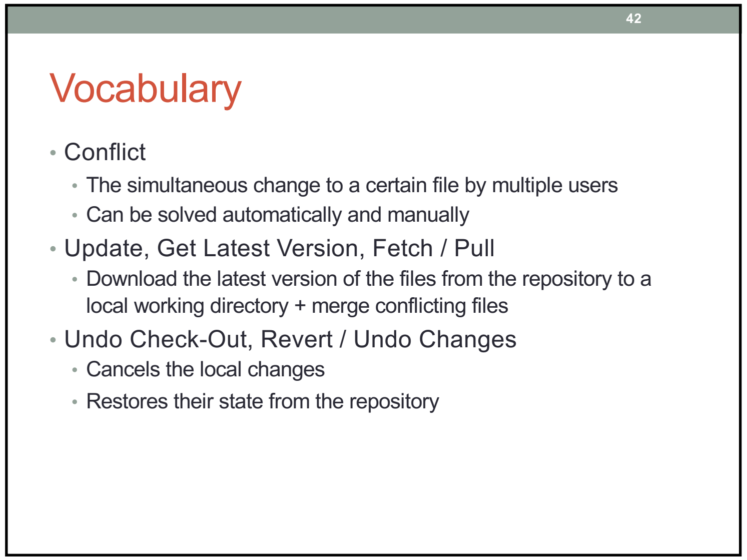
40



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!

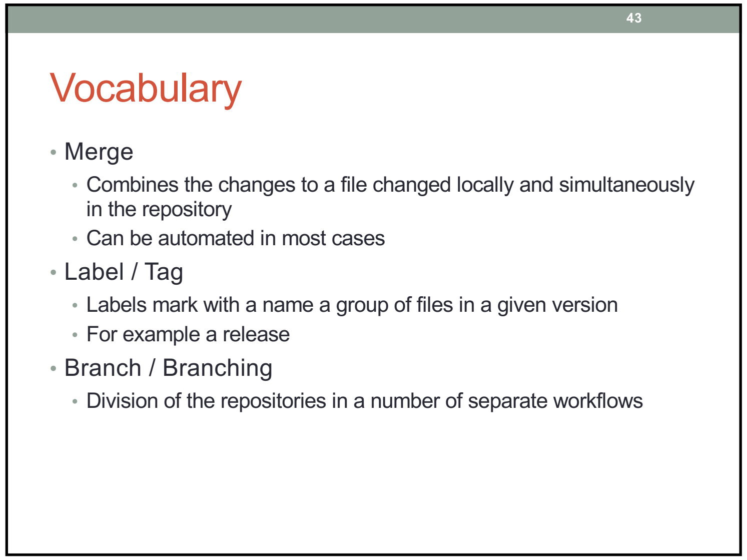
41



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
  - Restores their state from the repository

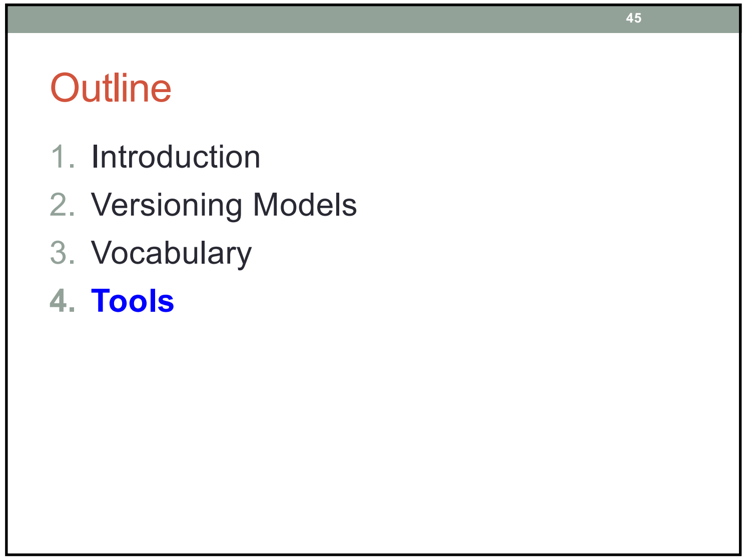
42



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

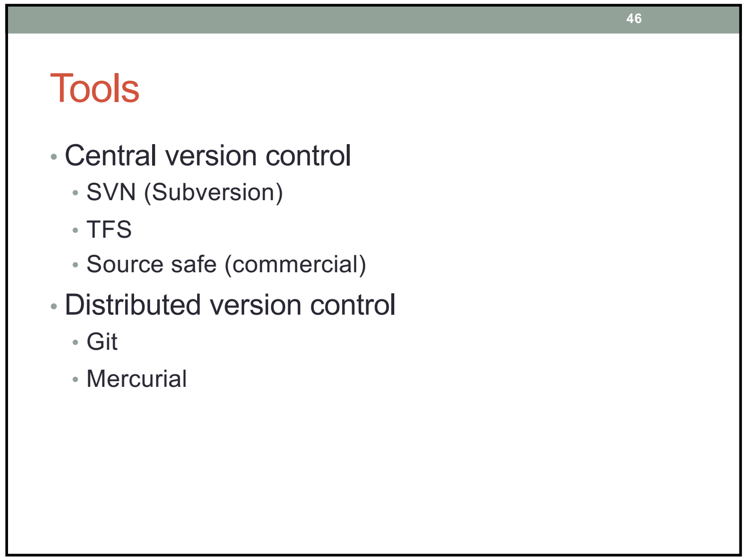
43



Outline

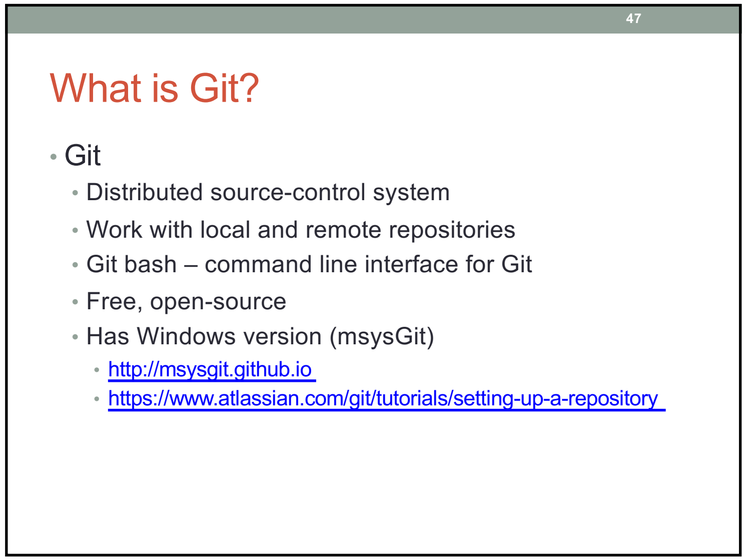
1. Introduction
2. Versioning Models
3. Vocabulary
4. **Tools**

45



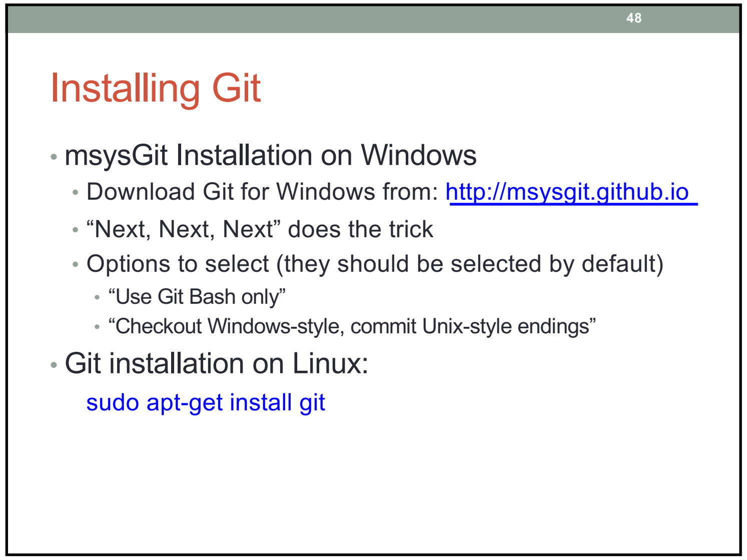
Tools

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



What is Git?

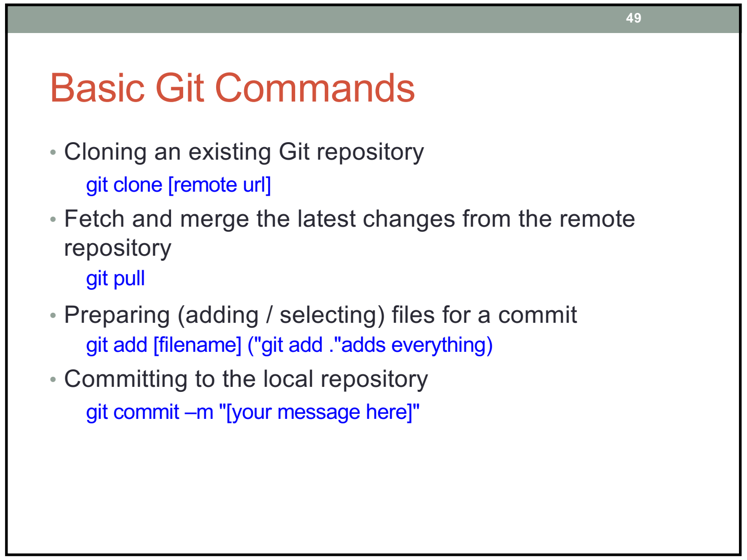
- Git
  - Distributed source-control system
  - Work with local and remote repositories
  - Git bash – command line interface for Git
  - Free, open-source
  - Has Windows version (msysGit)
    - http://msysgit.github.io
    - https://www.atlassian.com/git/tutorials/setting-up-a-repository



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

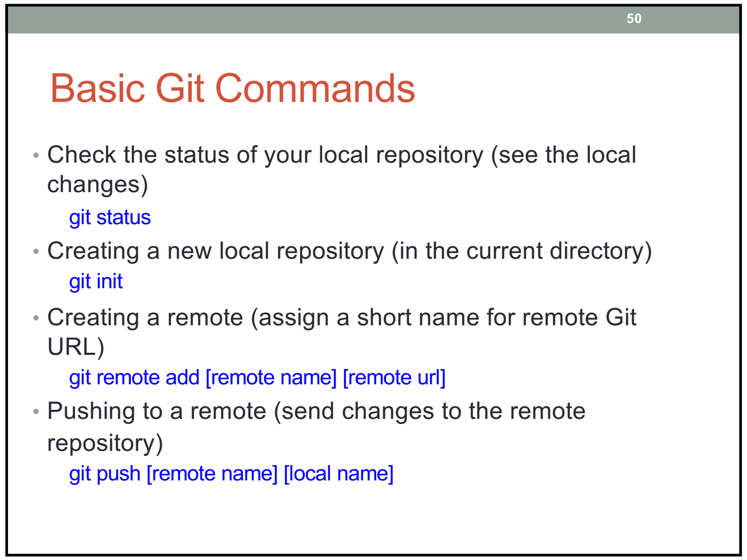
```
git pull
```

- Preparing (adding / selecting) files for a commit

```
git add [filename] ("git add ."adds everything)
```

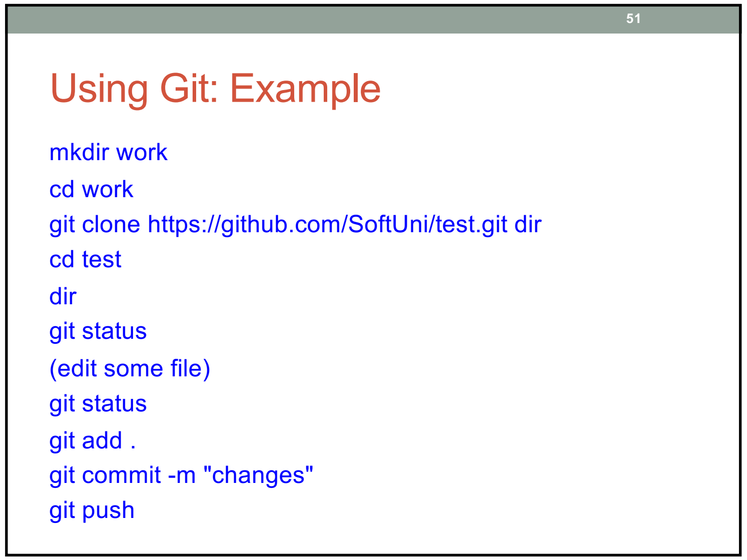
- Committing to the local repository

```
git commit –m "[your message here]"
```



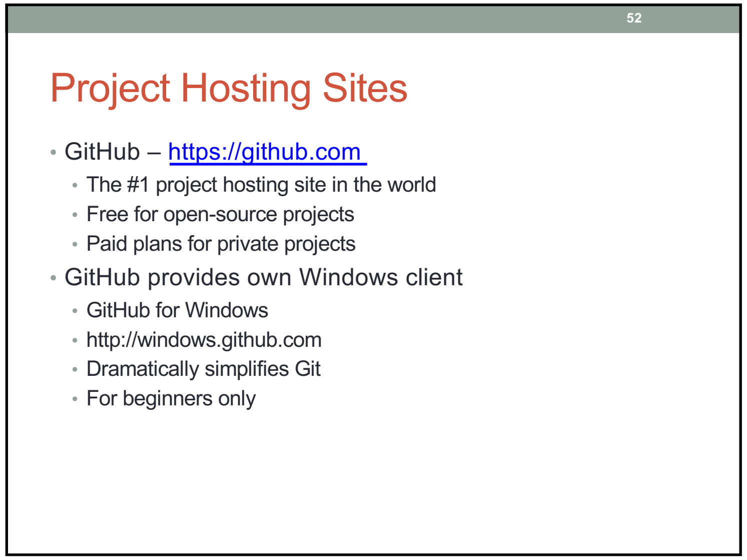
Basic Git Commands

- 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]



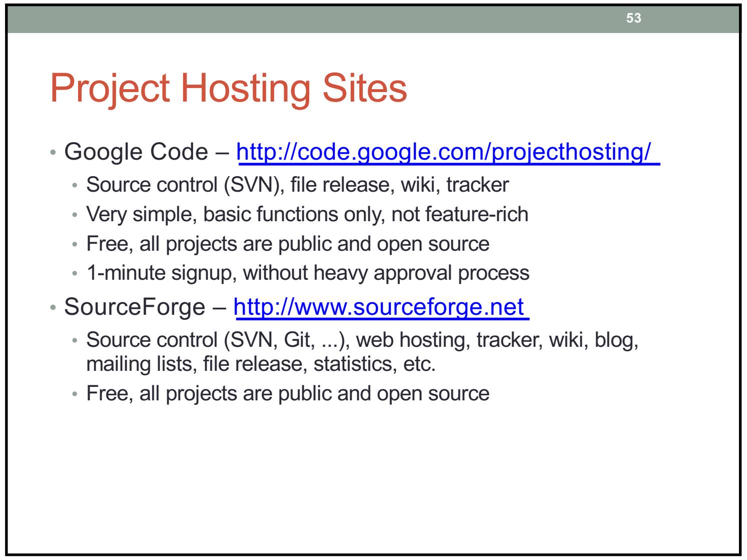
Using Git: Example

```
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
```



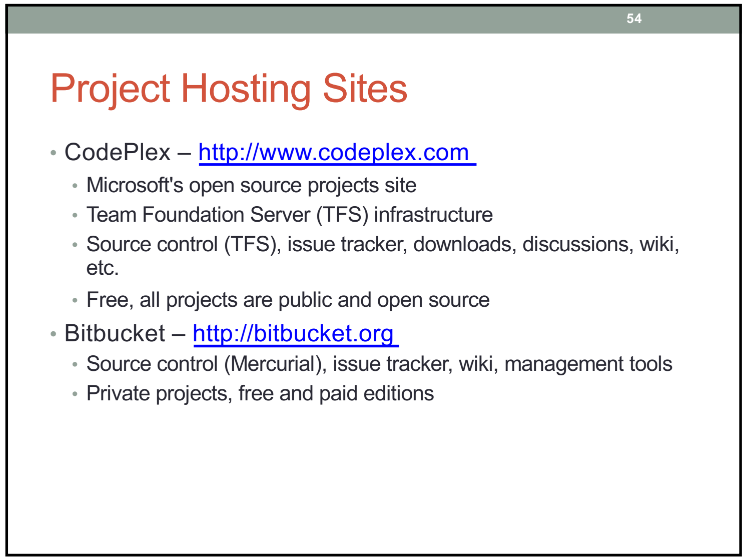
Project Hosting Sites

- 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

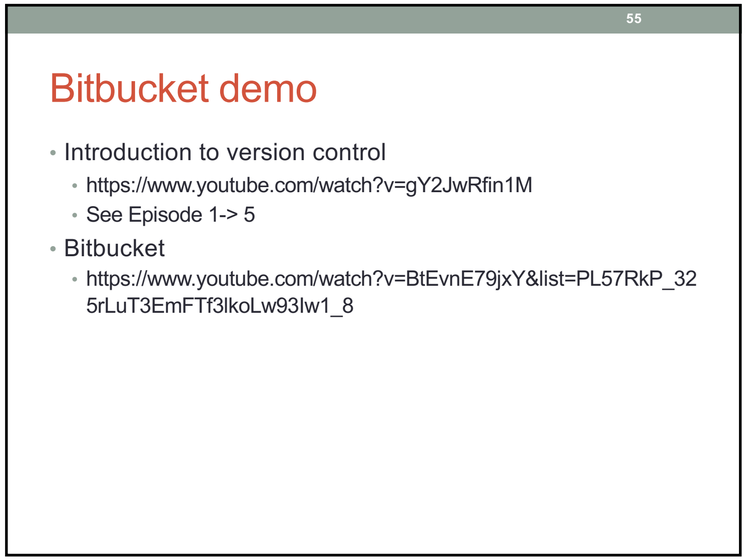
- Google Code – http://code.google.com/projecthosting/
  - Source control (SVN), file release, wiki, tracker
  - Very simple, basic functions only, not feature-rich
  - Free, all projects are public and open source
  - 1-minute signup, without heavy approval process
- SourceForge – http://www.sourceforge.net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



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

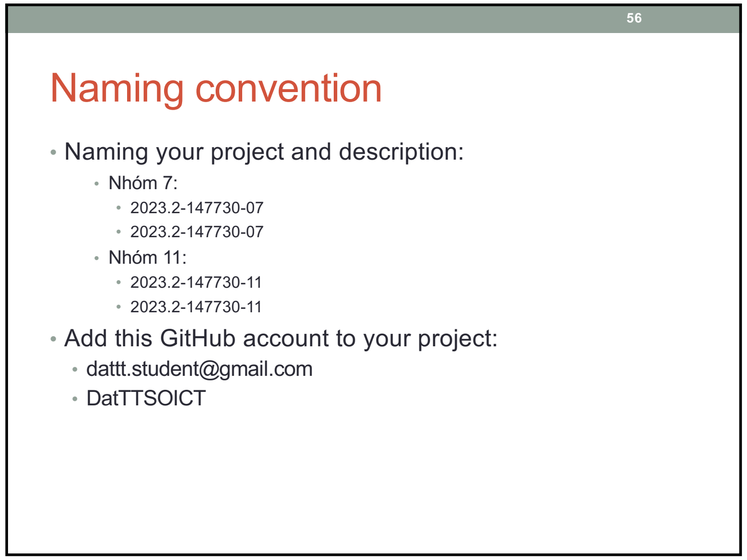
54



Bitbucket demo

- 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_325rLuT3EmFTf3lkoLw93lw1_8

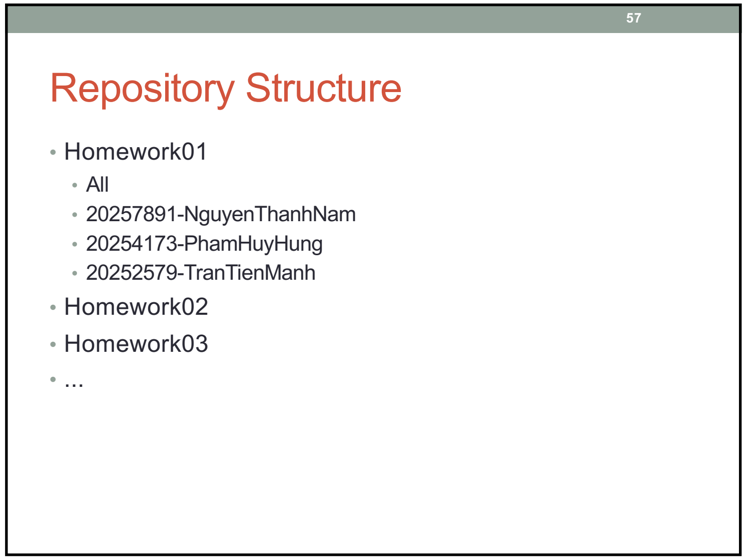
55



Naming convention

- Naming your project and description:
  - Nhóm 7:
    - 2023.2-147730-07
    - 2023.2-147730-07
  - Nhóm 11:
    - 2023.2-147730-11
    - 2023.2-147730-11
- Add this GitHub account to your project:
  - dattt.student@gmail.com
  - DatTTSOICT

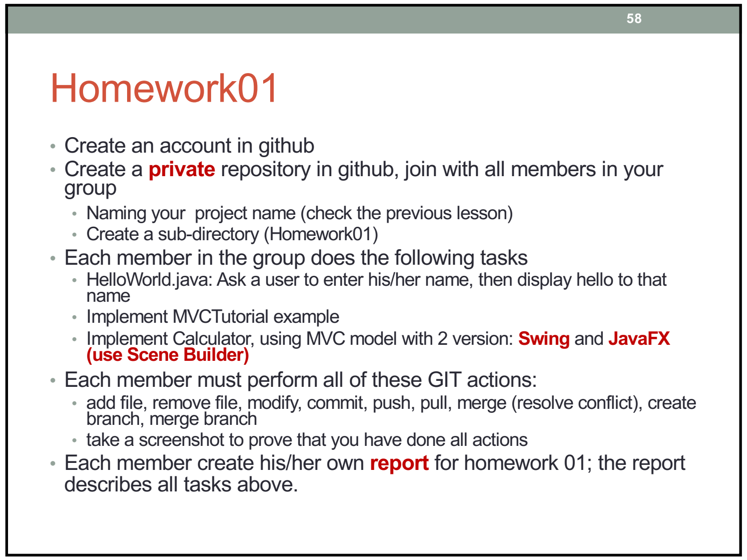
56



Repository Structure

- Homework01
  - All
  - 20257891-NguyenThanhNam
  - 20254173-PhamHuyHung
  - 20252579-TranTienManh
- Homework02
- Homework03
- ...

57



Homework01

- Create an account in github
- Create a **private** repository in github, join with all members in your group
  - Naming your project name (check the previous lesson)
  - Create a sub-directory (Homework01)
- Each member in the group does the following tasks
  - HelloWorld.java: Ask a user to enter his/her name, then display hello to that name
  - Implement MVCTutorial example
  - Implement Calculator, using MVC model with 2 version: **Swing** and **JavaFX (use Scene Builder)**
- Each member must perform all of these GIT actions:
  - add file, remove file, modify, commit, push, pull, merge (resolve conflict), create branch, merge branch
  - take a screenshot to prove that you have done all actions
- Each member create his/her own **report** for homework 01; the report describes all tasks above.

58