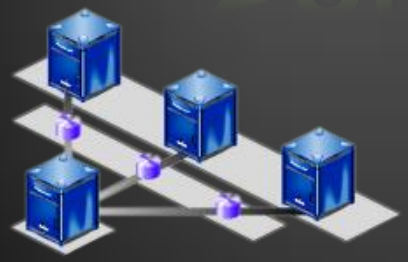
Telerik Academy

Source Control Systems



Source Control Repositories for Team Collaboration: Git

Software Engineering

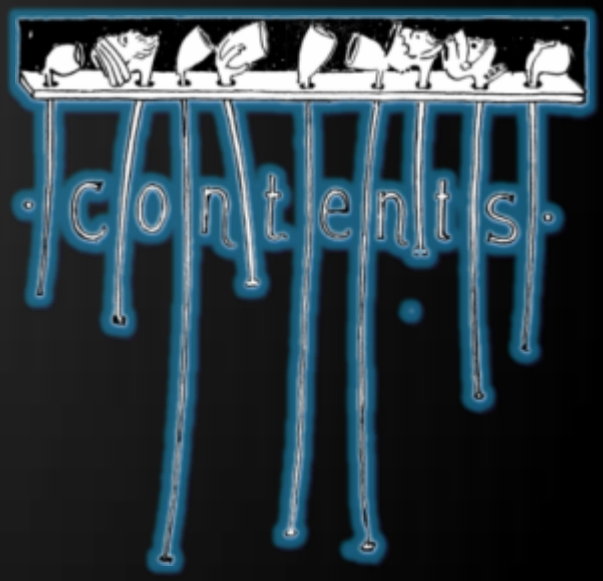
Telerik Software Academy

academy.telerik.com

Telerik Academy

Table of Contents

1. Software Configuration Management (SCM)
2. Version Control Systems: Philosophy
3. Versioning Models
   - Lock-Modify-Unlock
   - Copy-Modify-Merge
   - Distributed Version Control
4. Tags and Branching
5. Subversion, Git, TFS – Demo
6. Project Hosting Sites



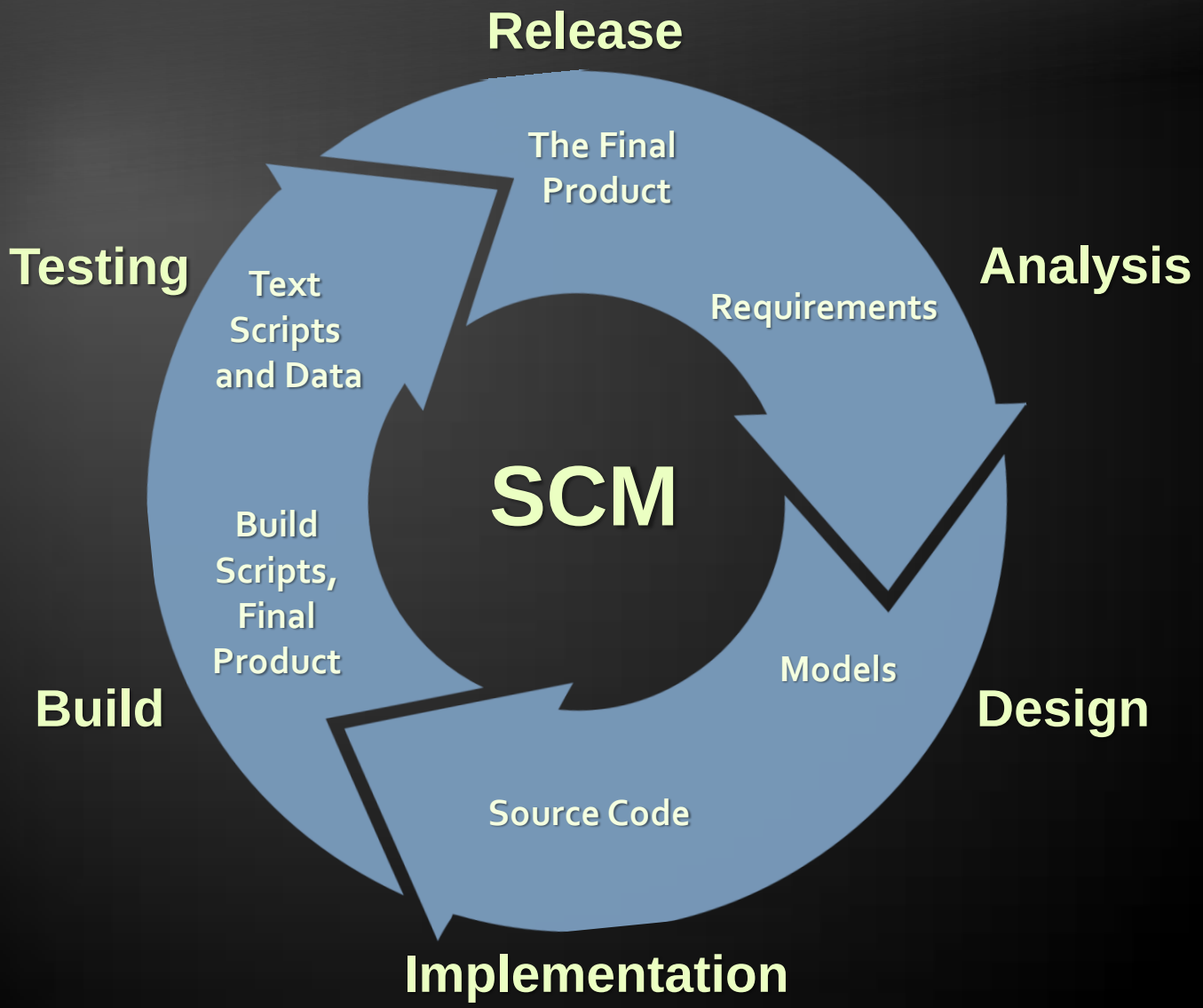
Telerik Academy

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
  - Solves conflicts in the changes

Telerik Academy

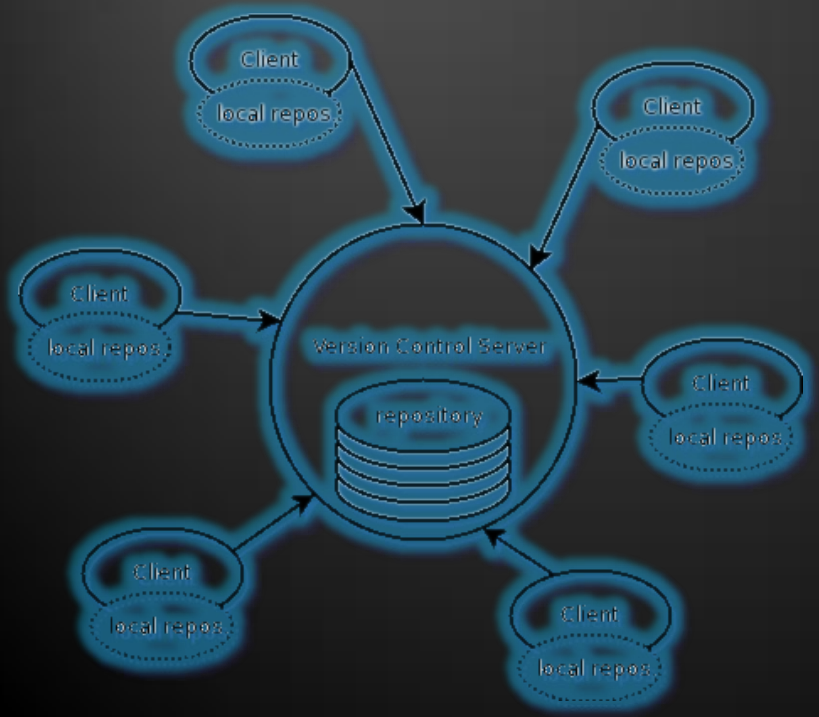
SCM and the Software Development Lifecycle



Telerik Academy

Version Control

Managing Different Version of the Same File / Document



Telerik Academy

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 – free

Version 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

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)

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)

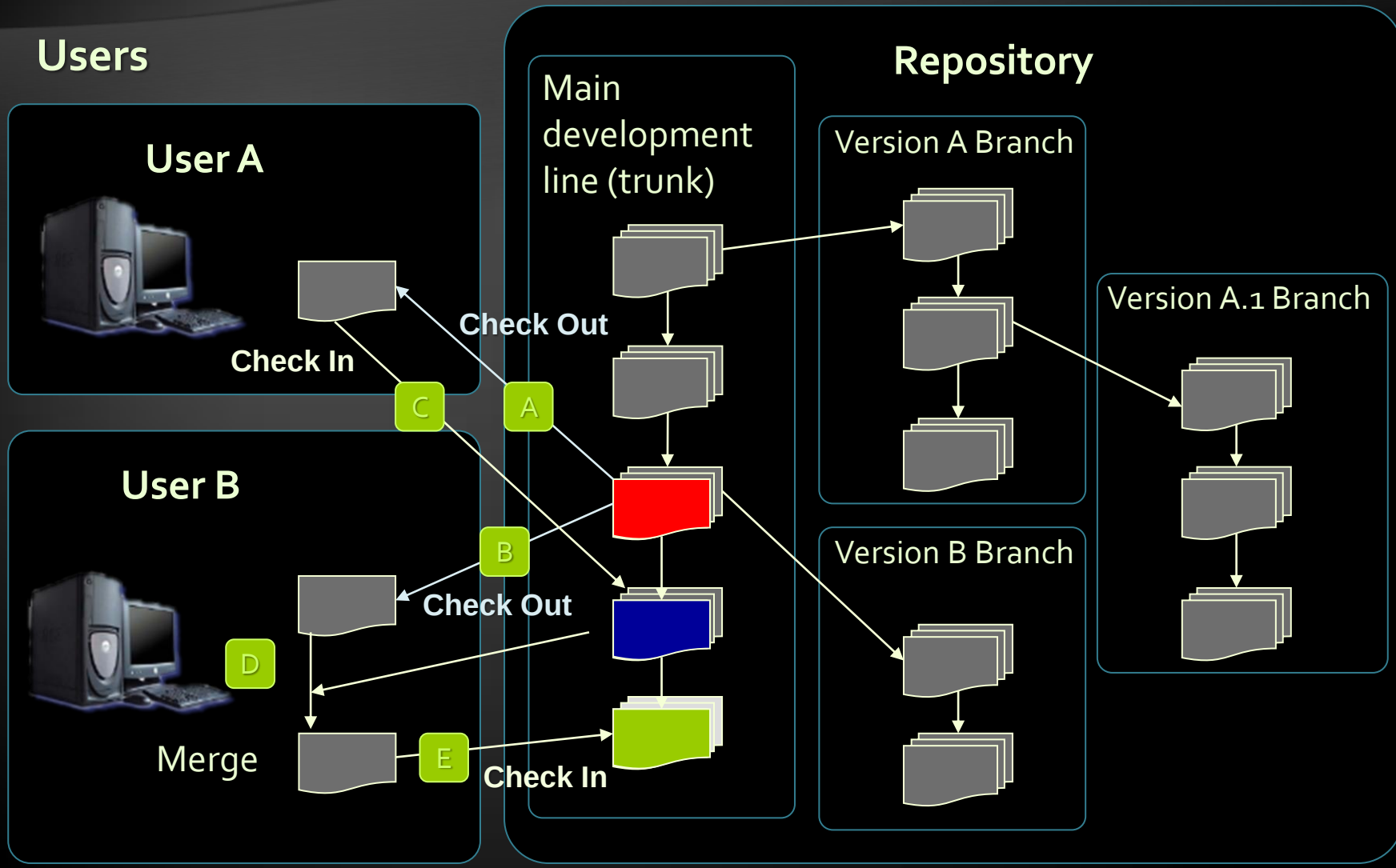
- Conflict
  - The simultaneous change to a certain file by multiple users
  - Can be solved automatically and manually
- Update, Get Latest Version, Fetch / Pull
  - Download the latest version of the files from the repository to a local working directory
- Undo Check-Out, Revert / Undo Changes
  - 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
- Branching
  - Division of the repositories in a number of separate work flows

Telerik Academy

Version Control: Typical Scenario

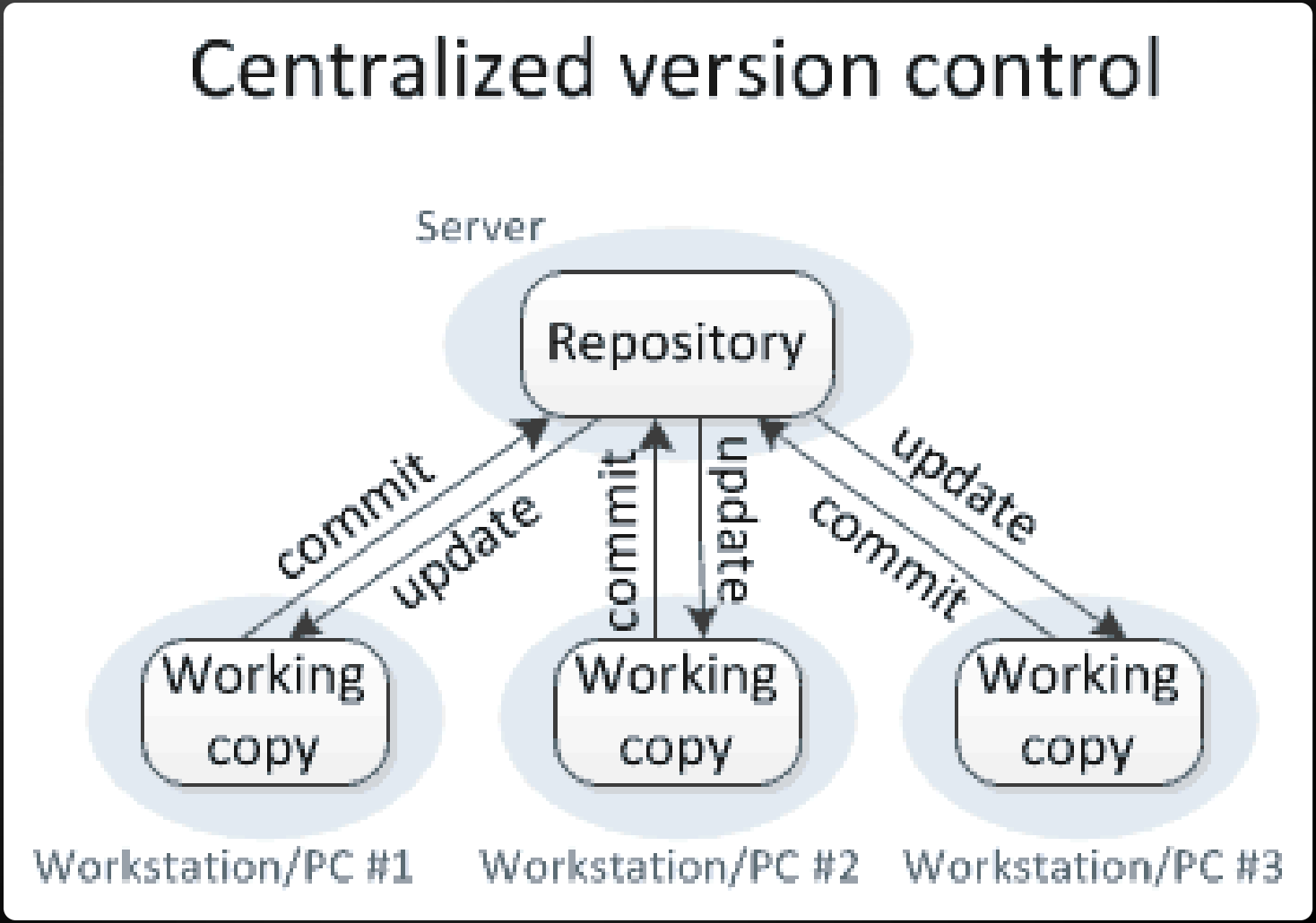


Telerik Academy

Versioning Models

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

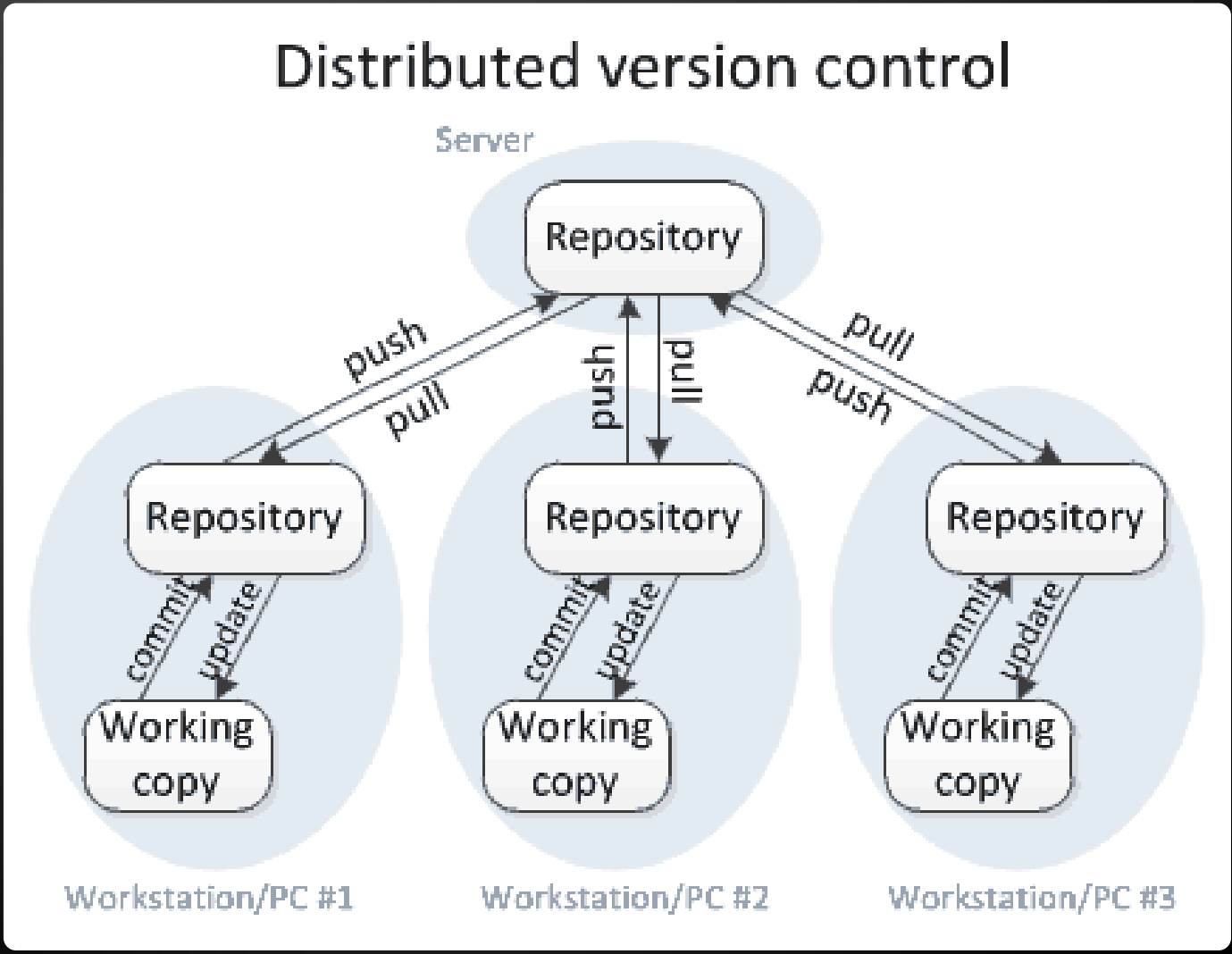
Centralized Version Control



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

Telerik Academy

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 (old fashioned)
    - TFS, SVN, Git (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, TFS, Git

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

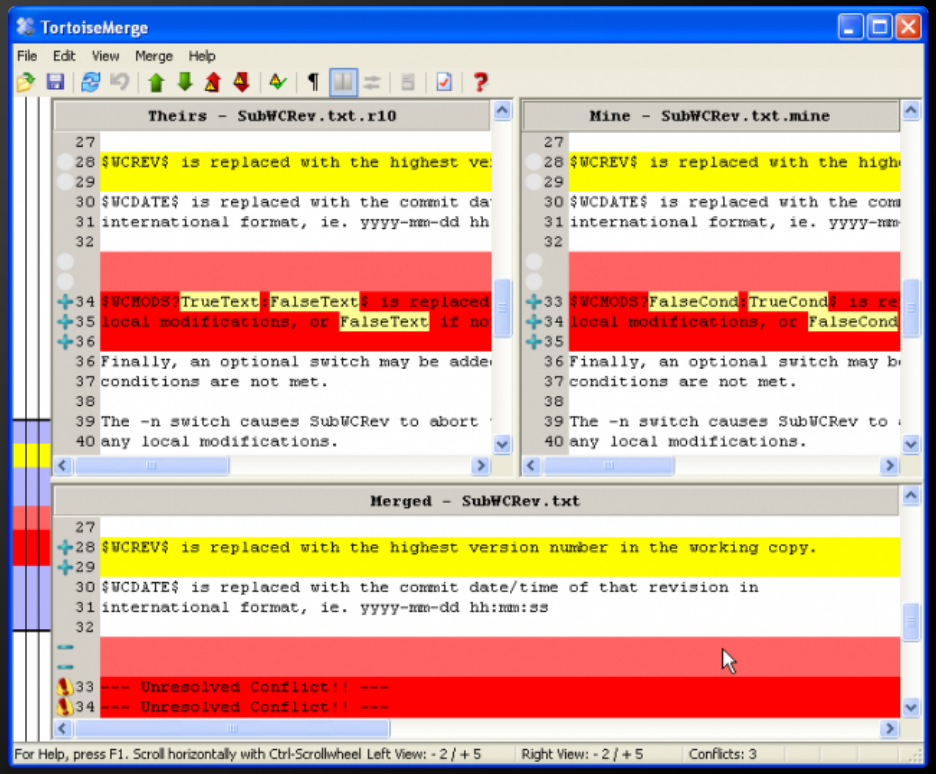
Merging Problems

- If a given file is concurrently modified, it is necessary to merge the changes
  - Merging is hard!
    - It is not always possible to do it automatically
- Responsibility and coordination 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 comments at each commit

Telerik Academy

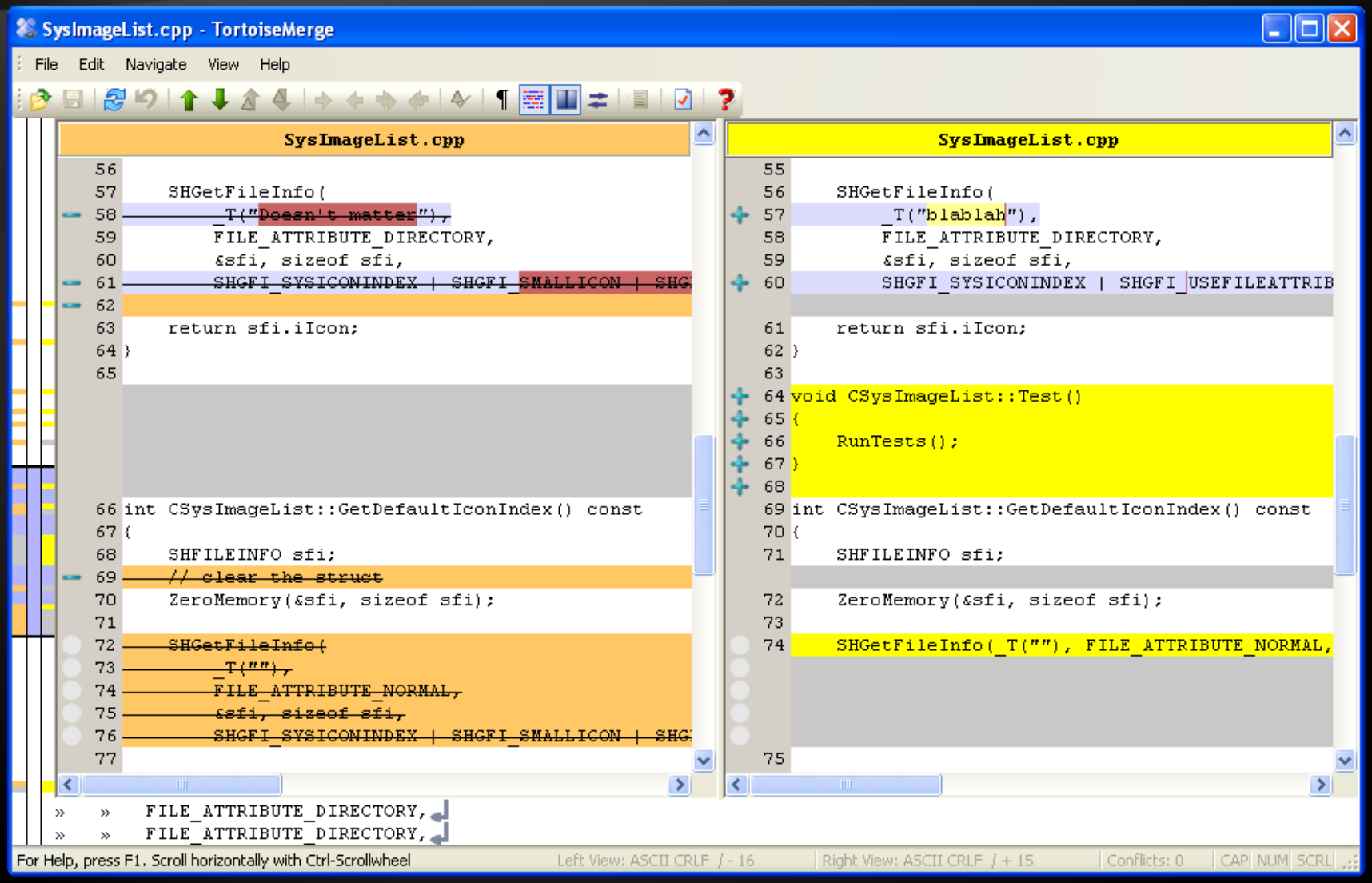
File Comparison / Merge Tools

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



Telerik Academy

File Comparison – Example



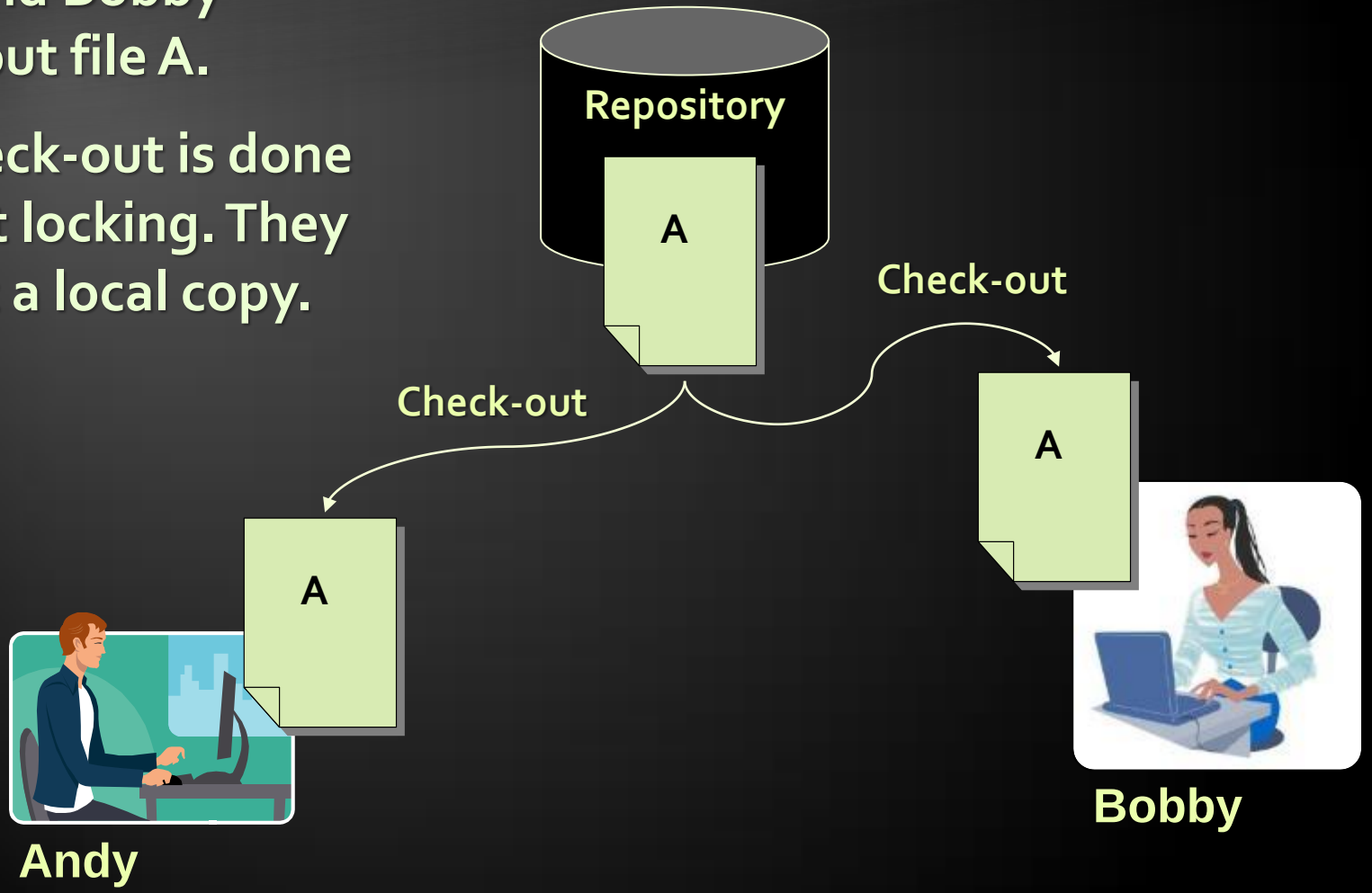
Telerik Academy

The "Lock-Modify-Unlock" Model

The Lock-Modify-Unlock Model (1)

Andy and Bobby check-out file A.

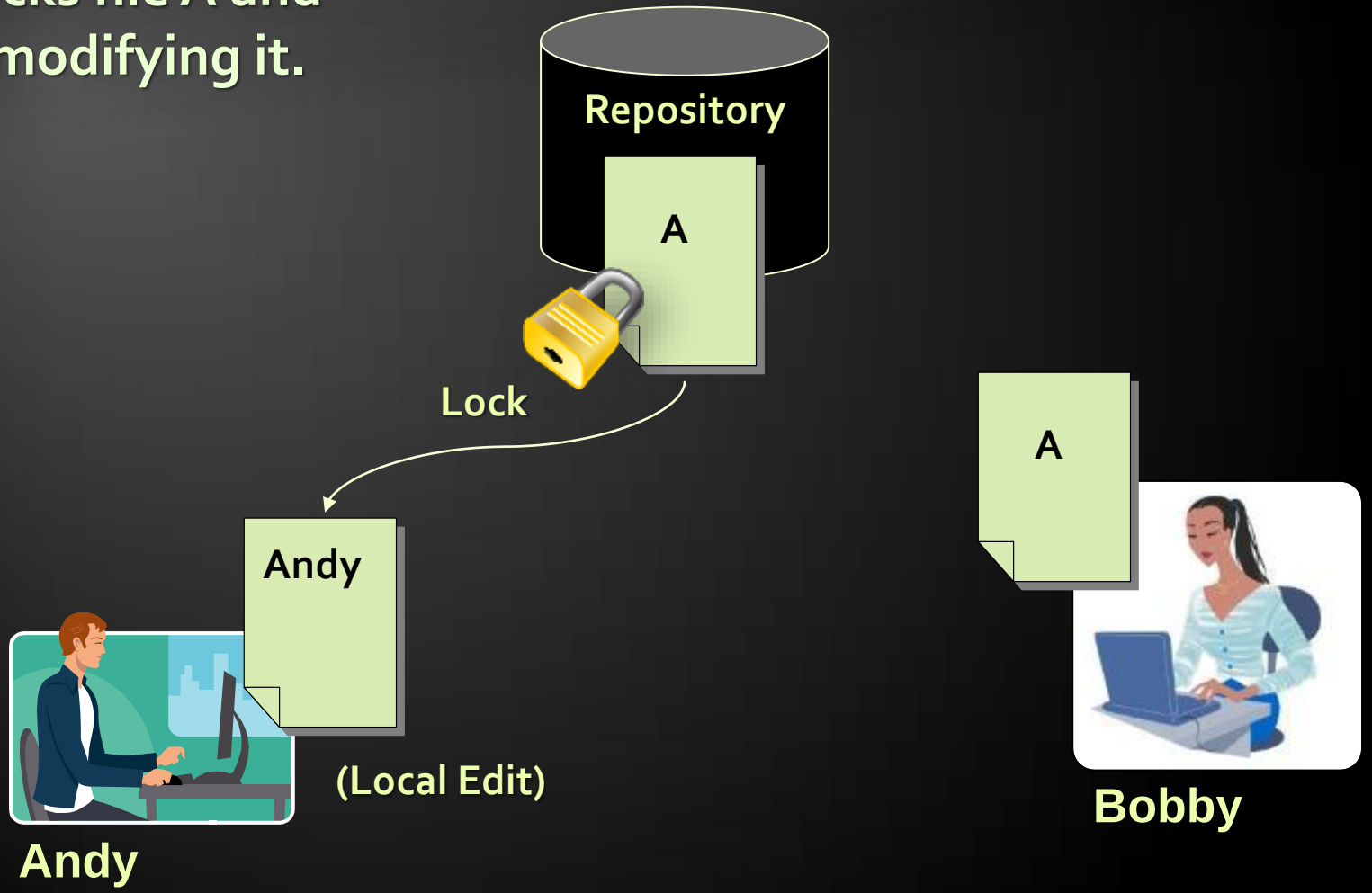
The check-out is done without locking. They just get a local copy.



Telerik Academy

The Lock-Modify-Unlock Model (2)

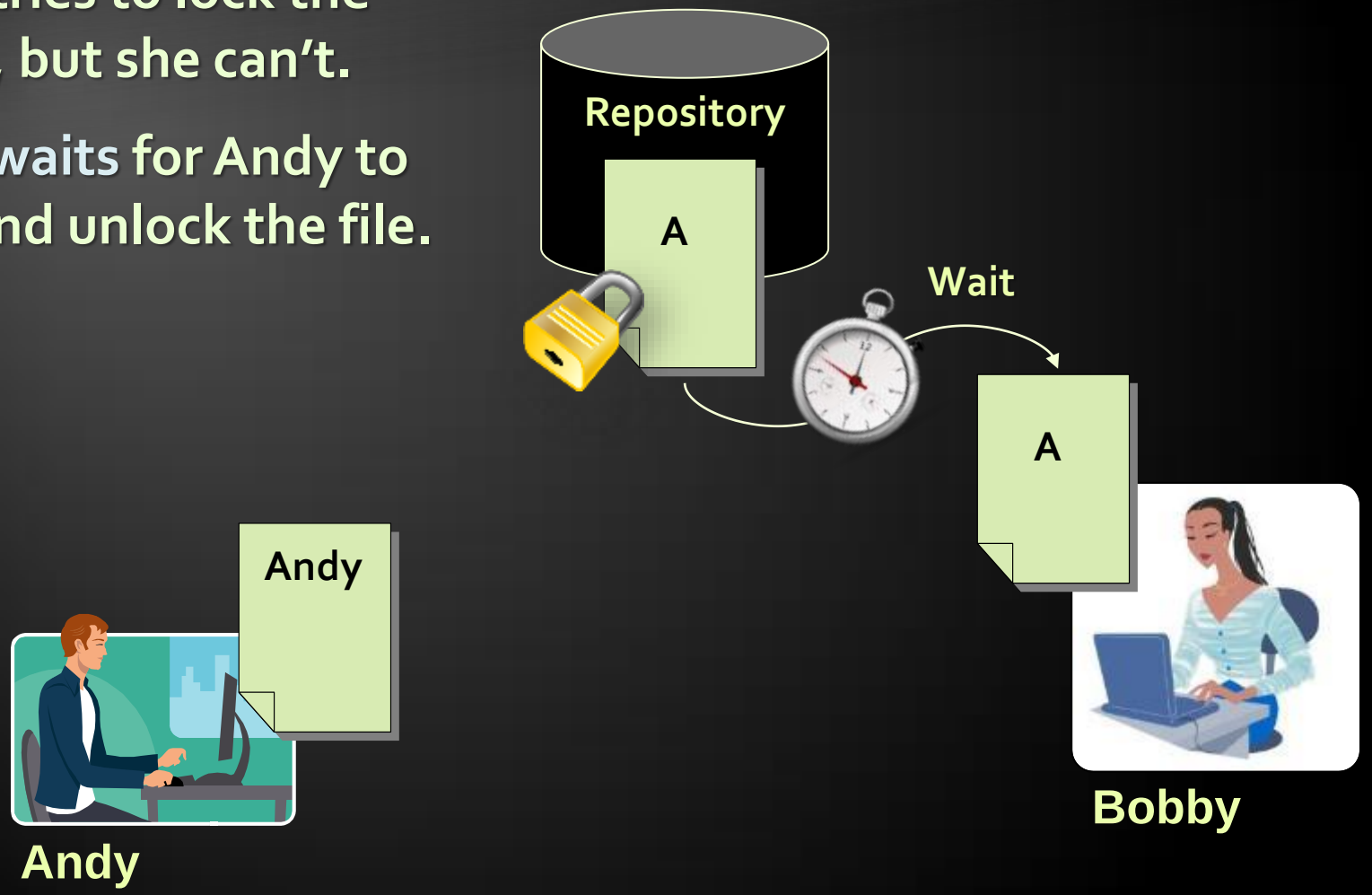
Andy locks file A and begins modifying it.



The Lock-Modify-Unlock Model (3)

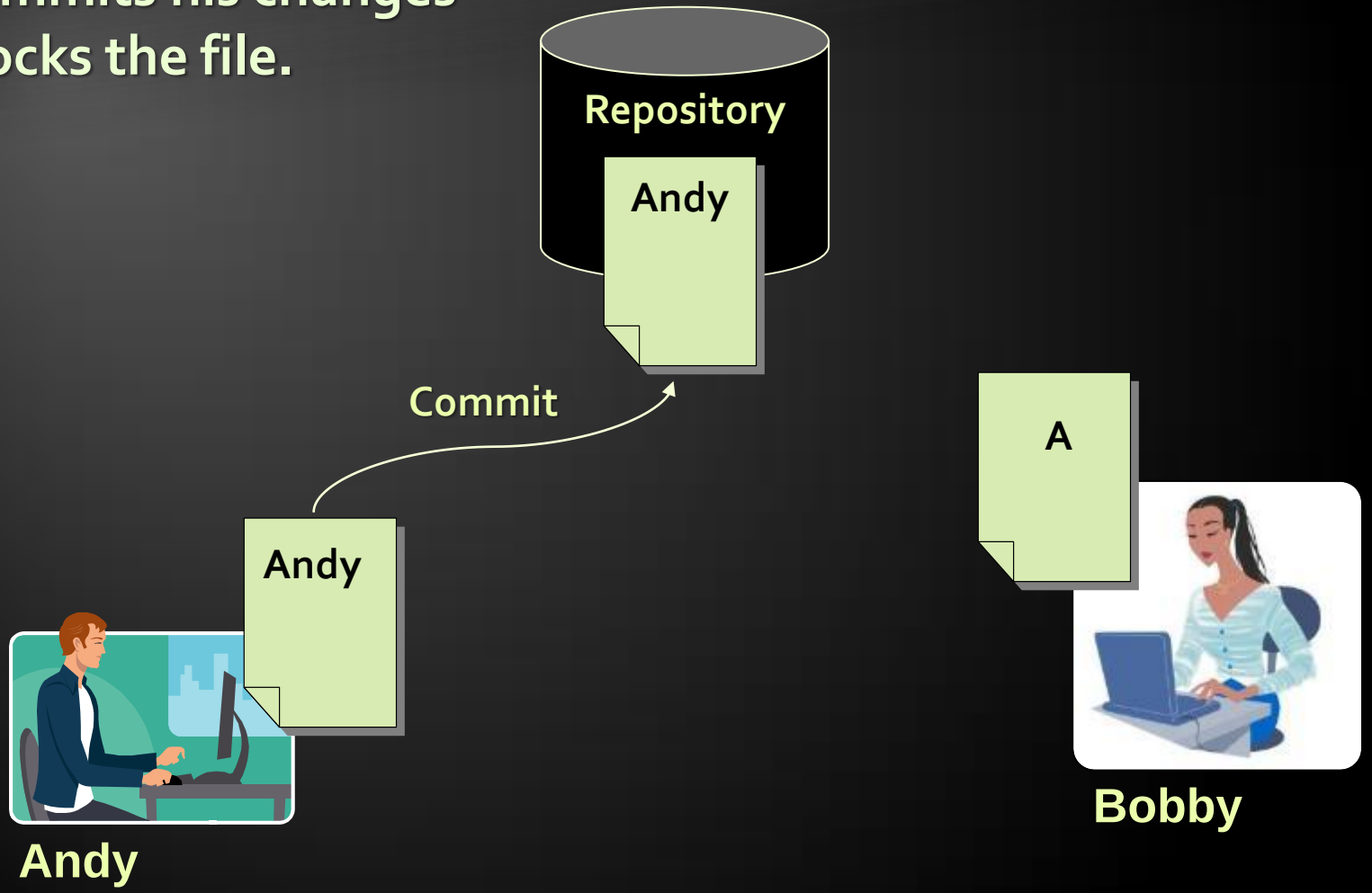
Bobby tries to lock the file too, but she can't.

Bobby waits for Andy to finish and unlock the file.



The Lock-Modify-Unlock Model (4)

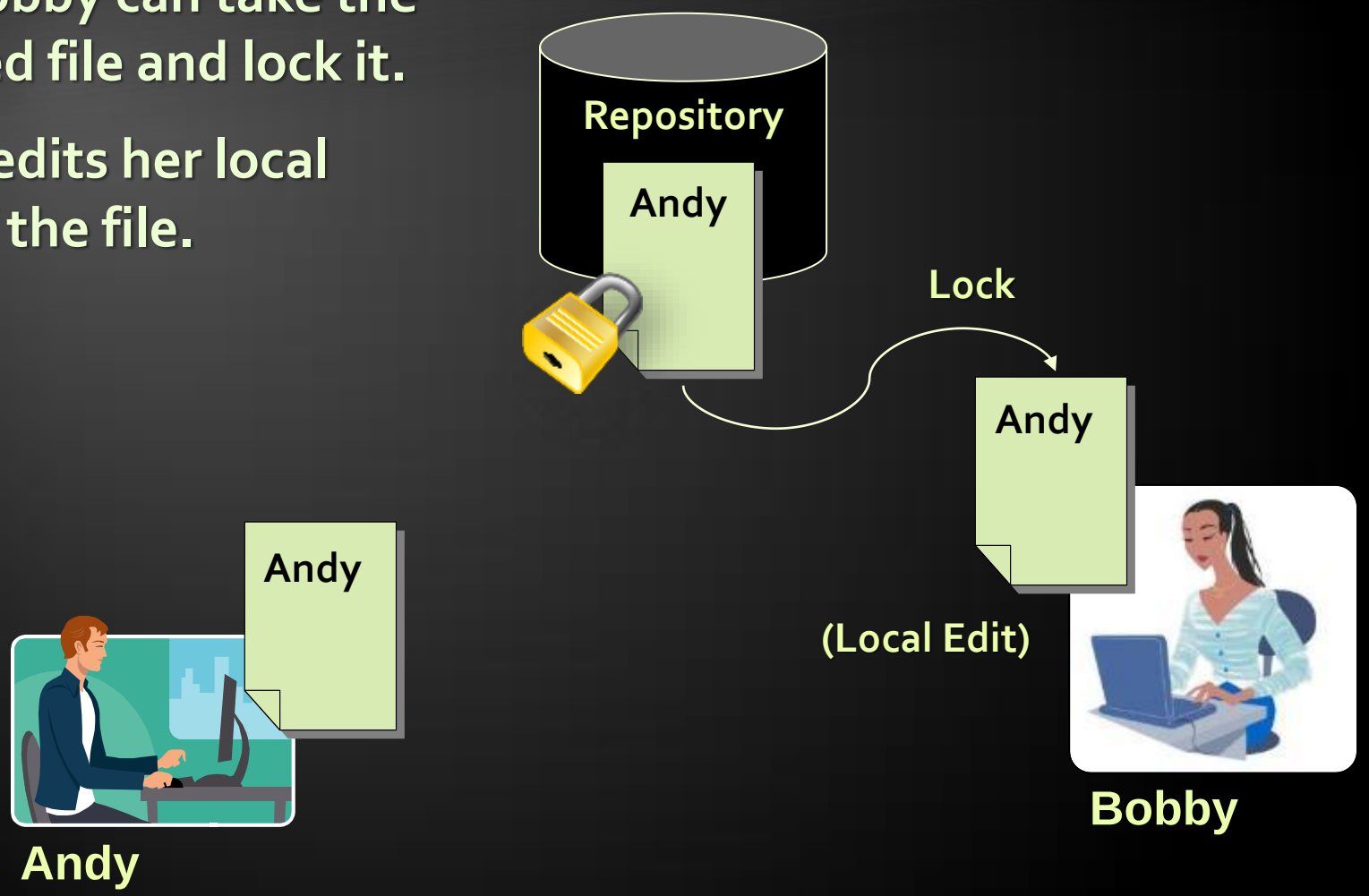
Andy commits his changes and unlocks the file.



The Lock-Modify-Unlock Model (5)

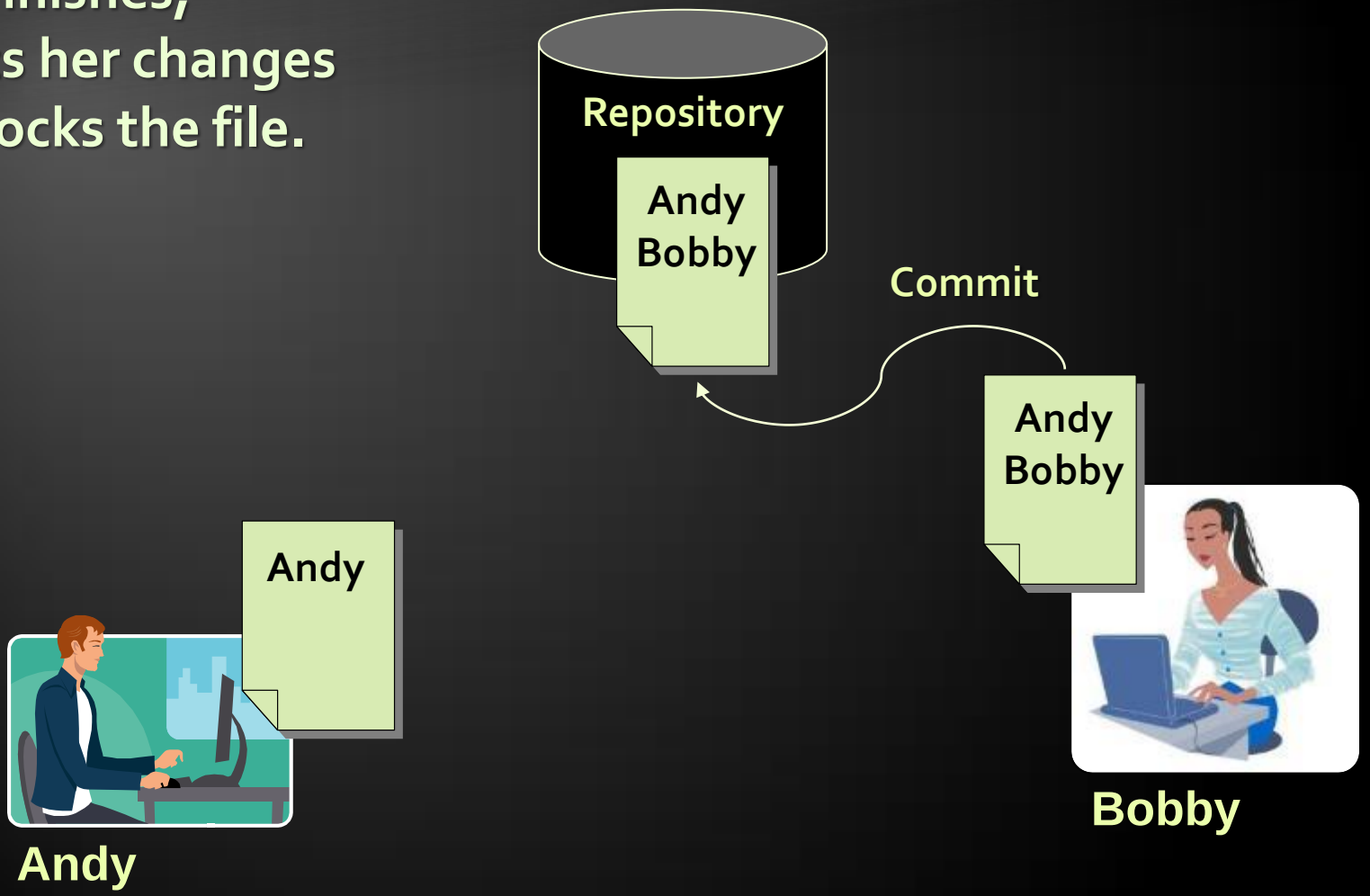
Now Bobby can take the modified file and lock it.

Bobby edits her local copy of the file.



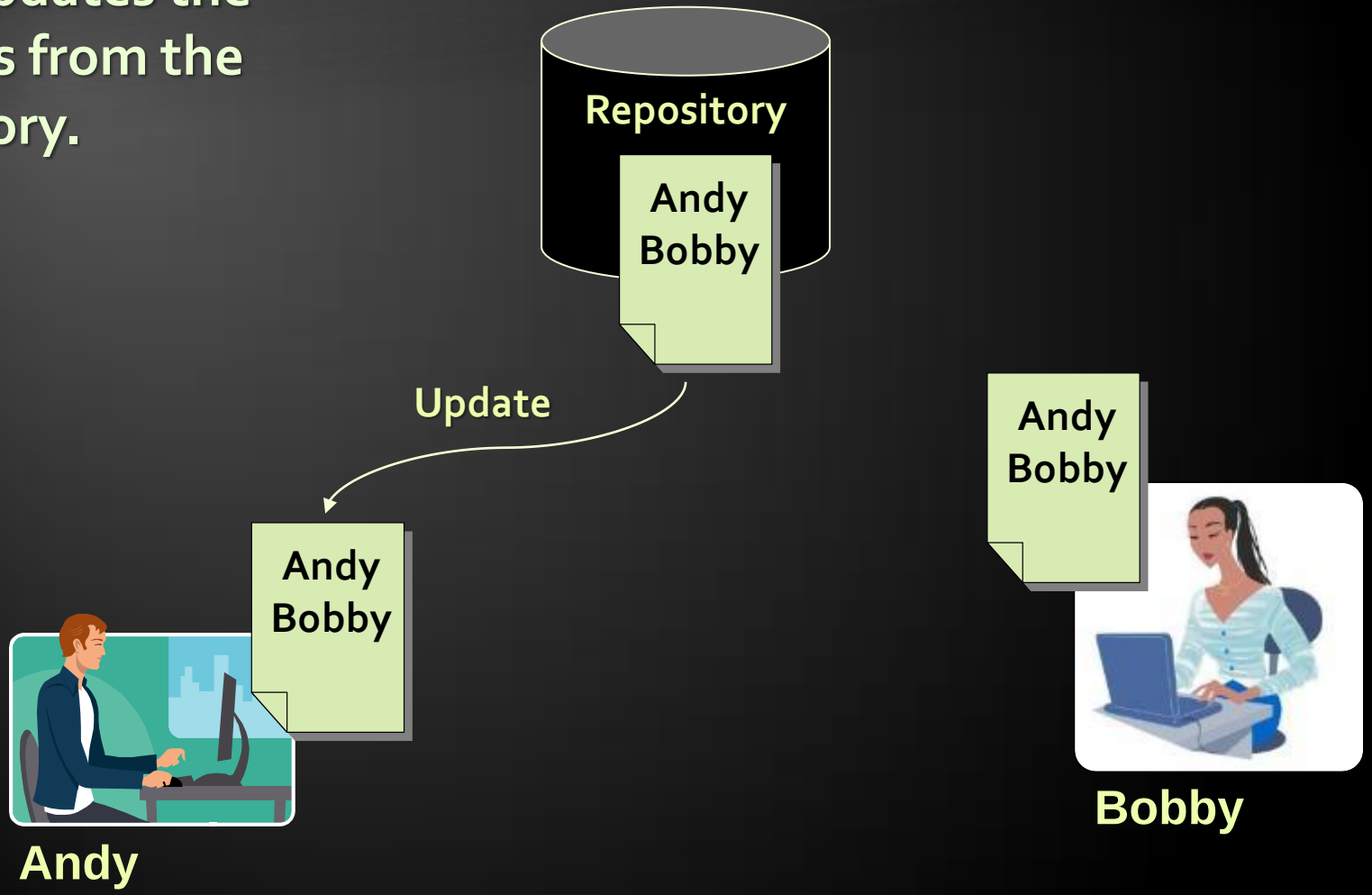
The Lock-Modify-Unlock Model (6)

Bobby finishes, commits her changes and unlocks the file.



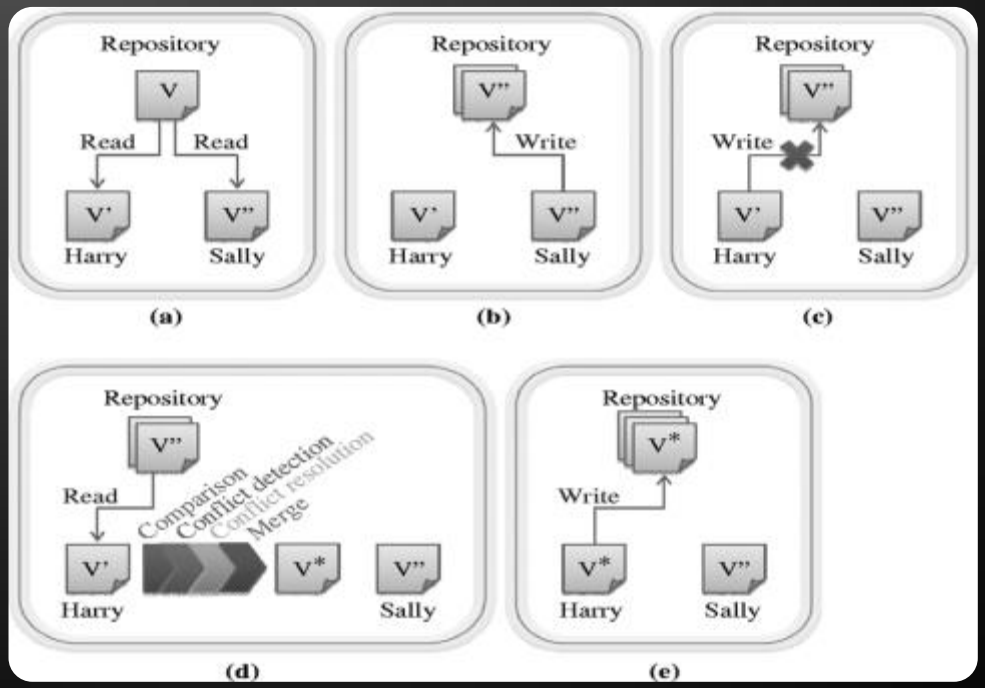
The Lock-Modify-Unlock Model (7)

Andy updates the changes from the repository.



Telerik Academy

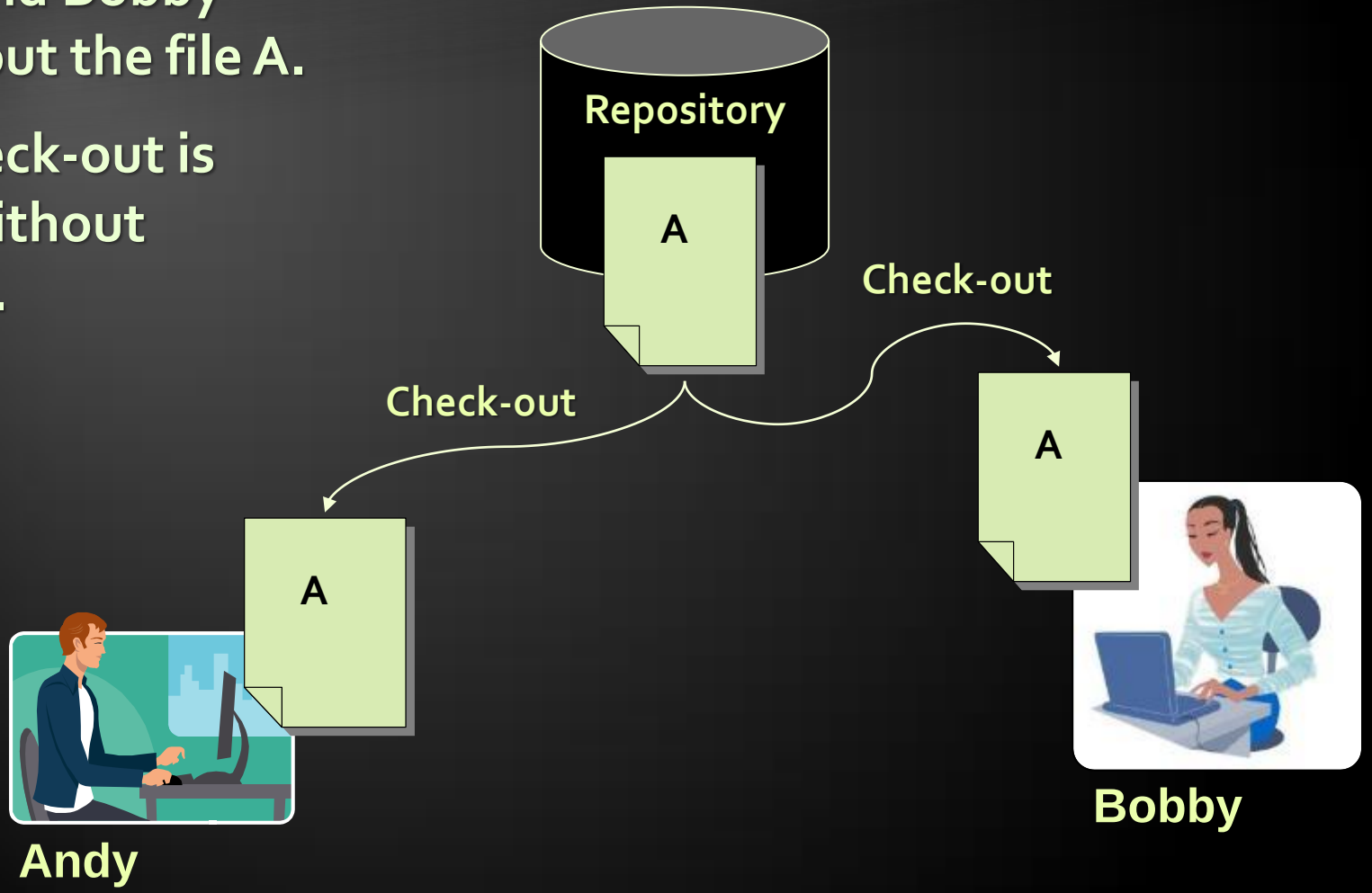
The "Copy-Modify-Merge" Model



The Copy-Modify-Merge Model (1)

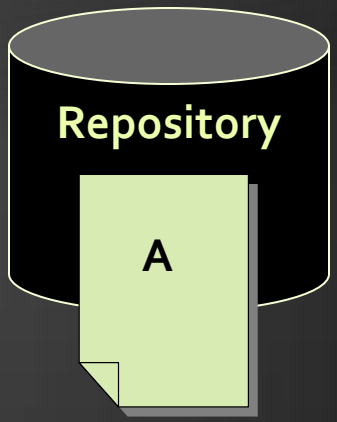
Andy and Bobby check-out the file A.

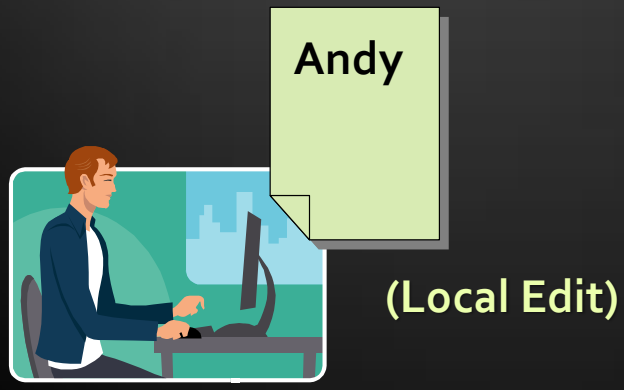
The check-out is done without locking.

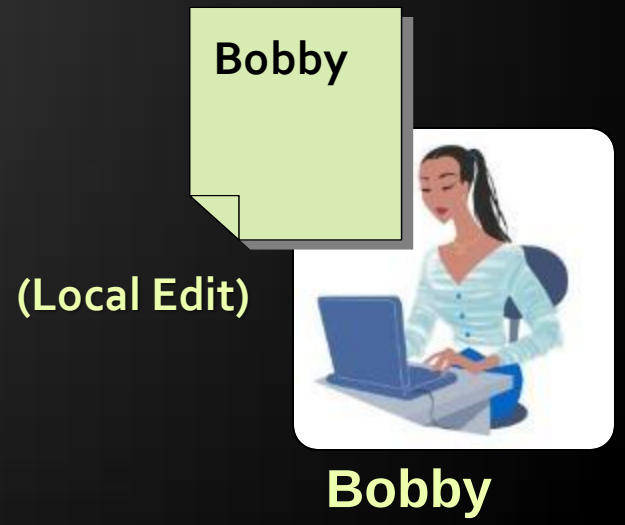


The Copy-Modify-Merge Model (2)

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

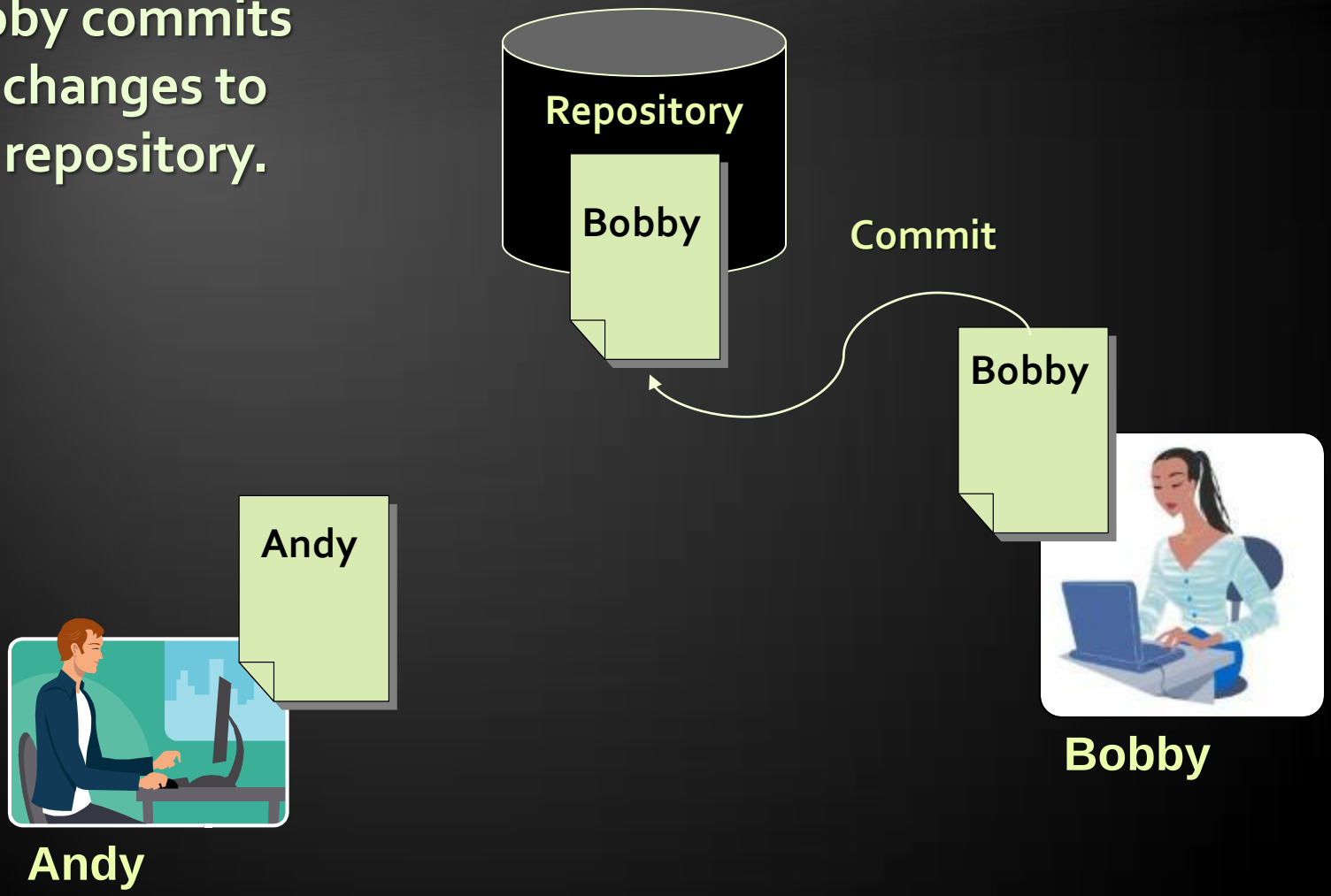






The Copy-Modify-Merge Model (3)

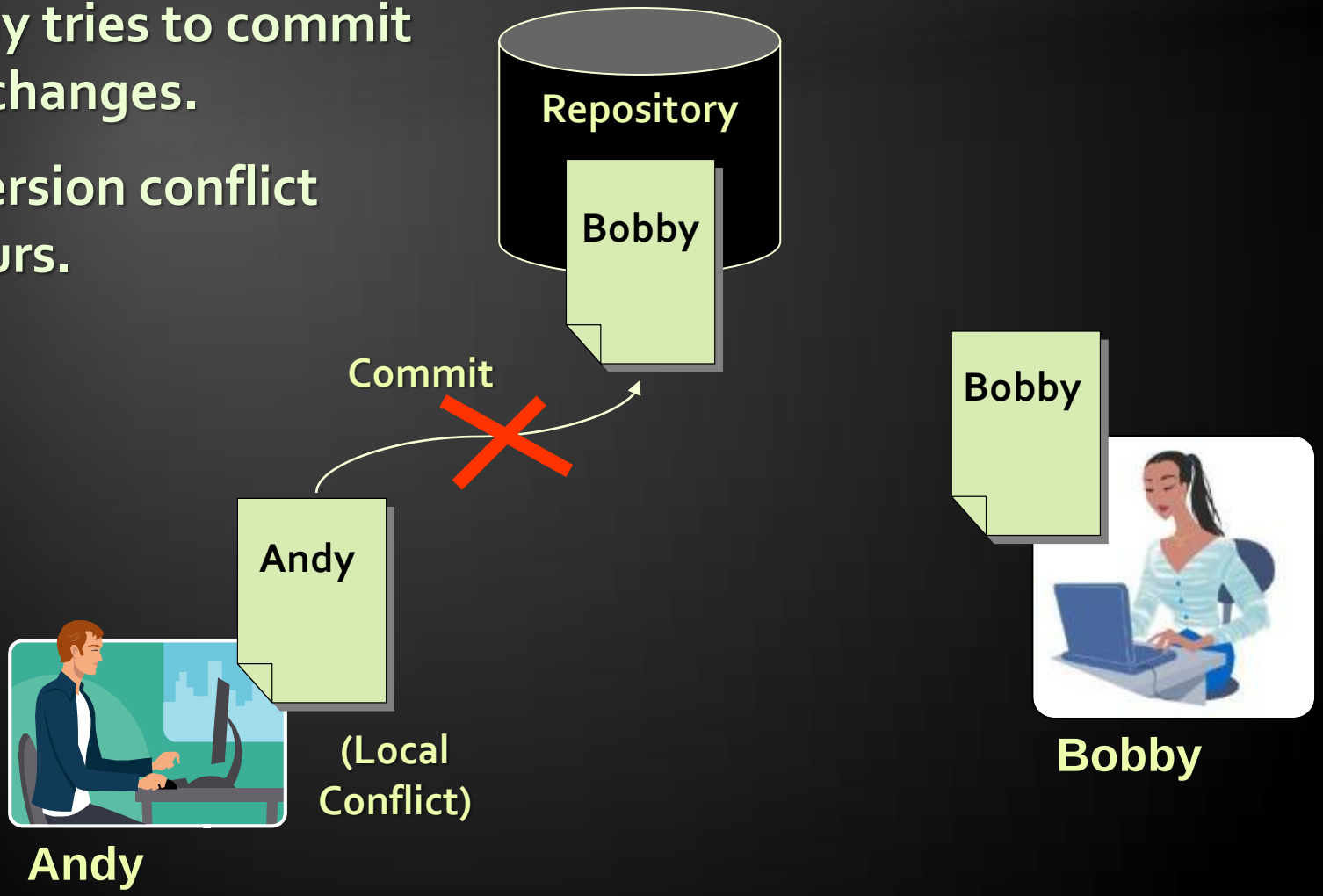
Bobby commits her changes to the repository.



The Copy-Modify-Merge Model (4)

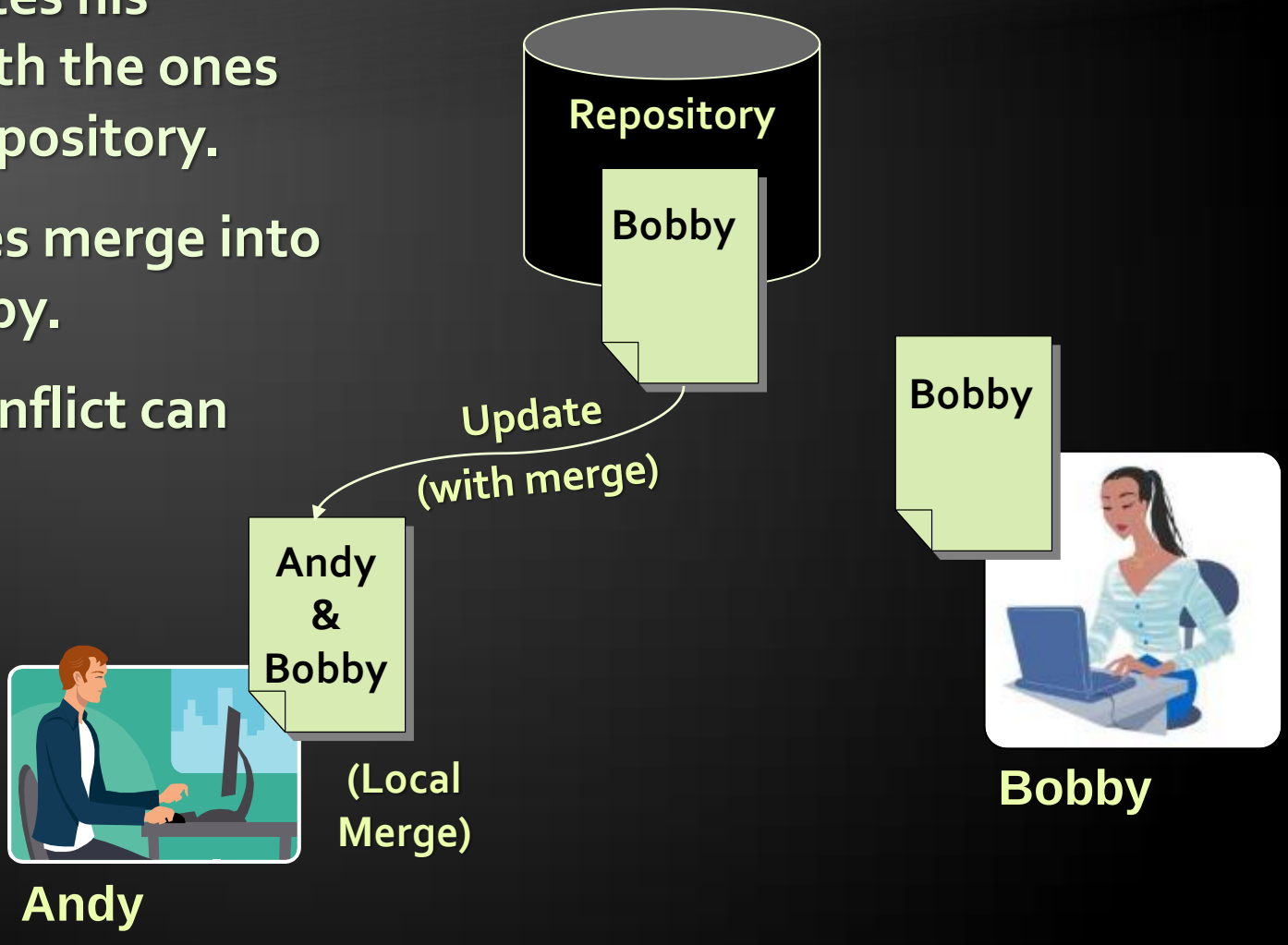
Andy tries to commit his changes.

A version conflict occurs.



The Copy-Modify-Merge Model (5)

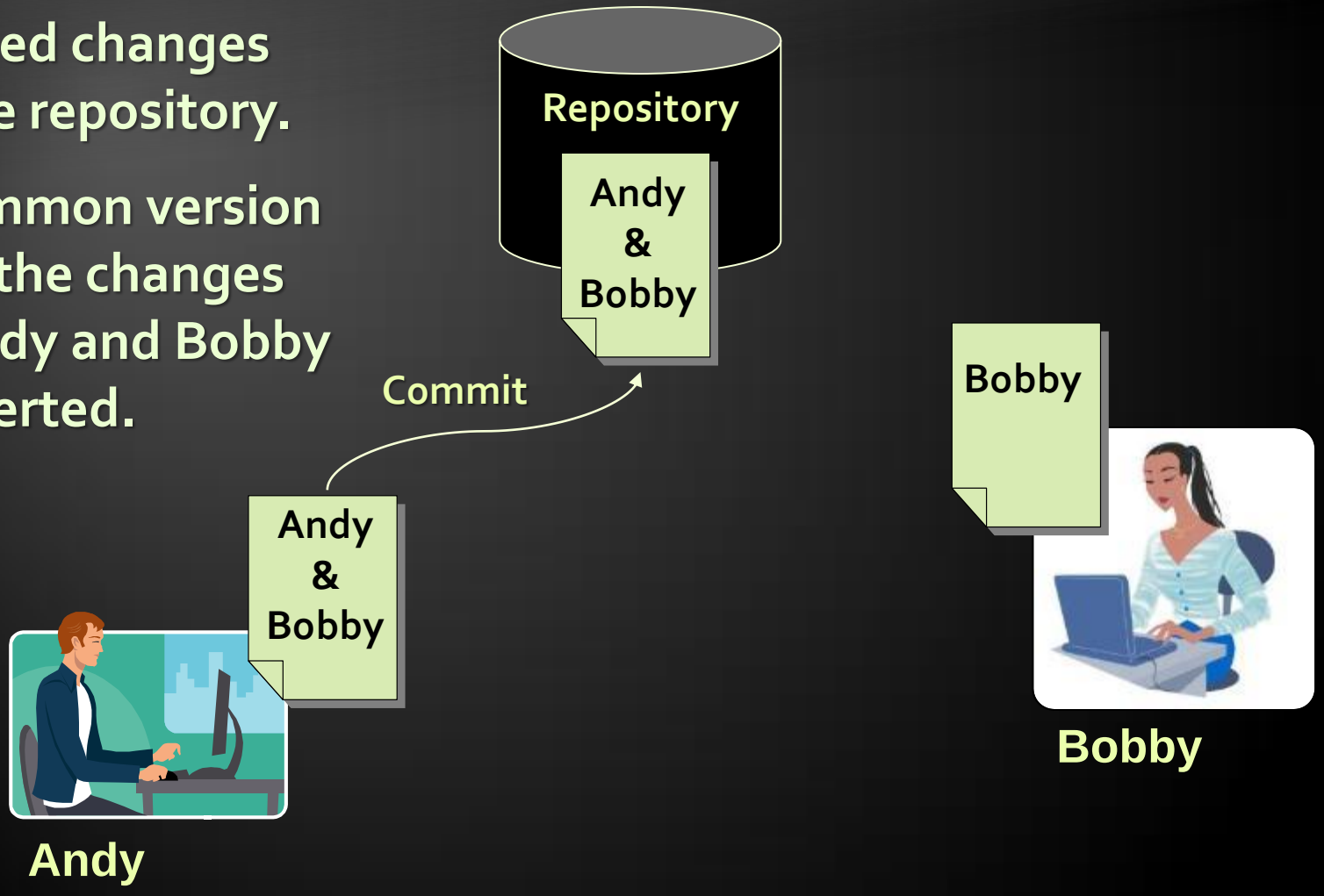
- Andy updates his changes with the ones from the repository.
- The changes merge into his local copy.
- A merge conflict can occur.



The Copy-Modify-Merge Model (6)

Andy commits the merged changes to the repository.

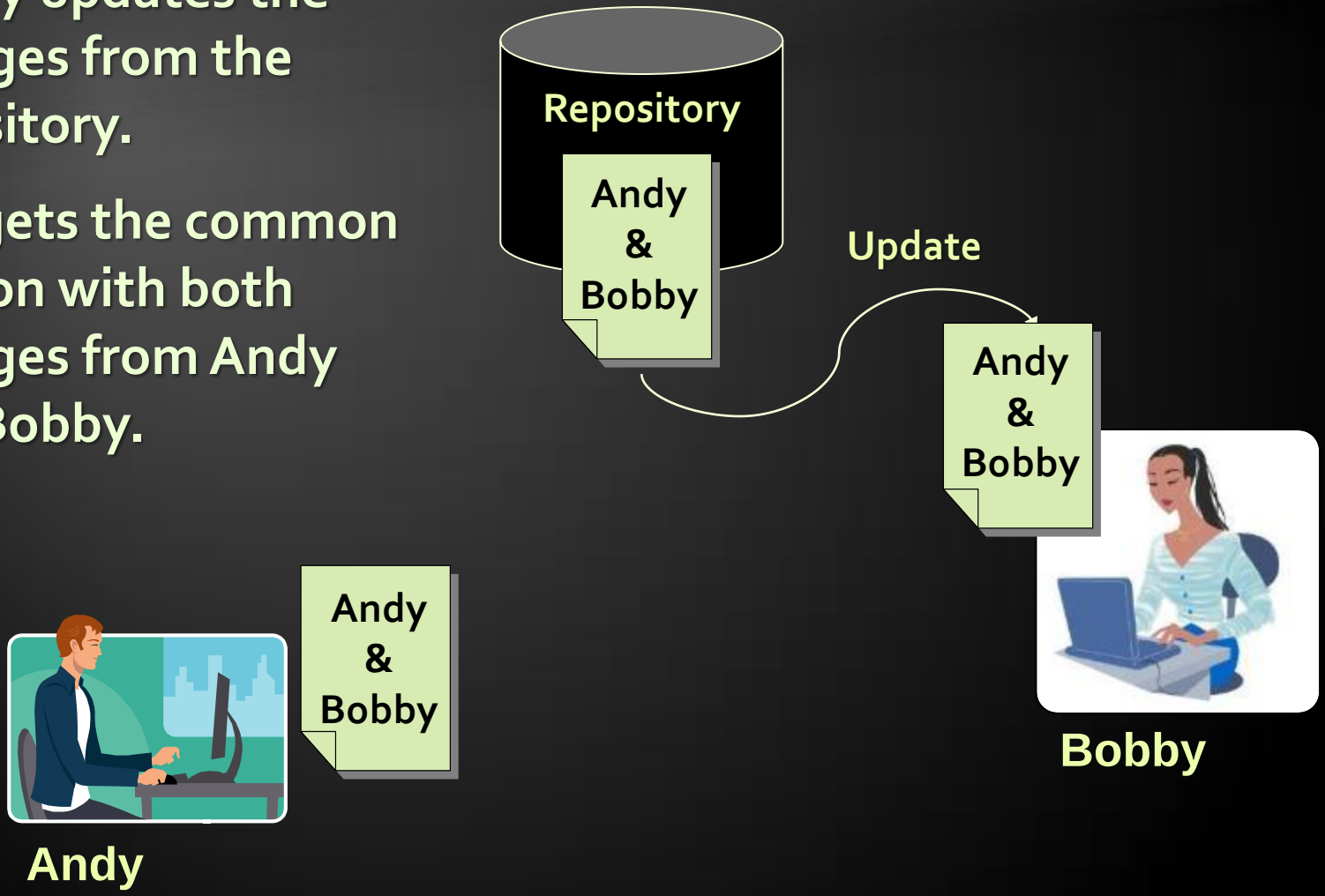
A common version with the changes of Andy and Bobby is inserted.



The Copy-Modify-Merge Model (7)

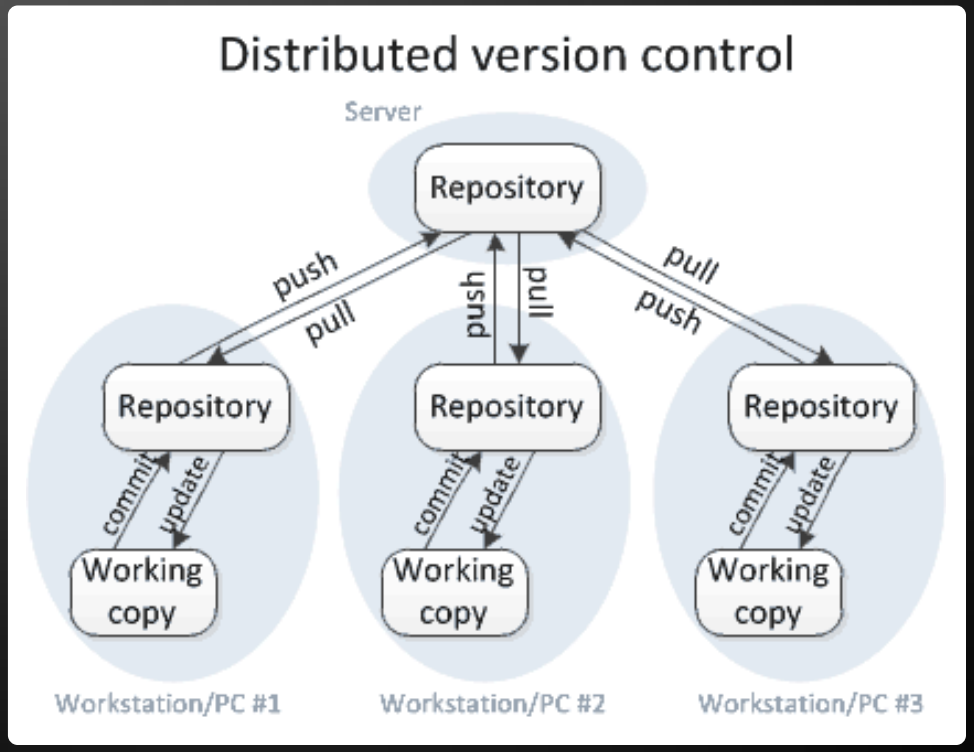
Bobby updates the changes from the repository.

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



Telerik Academy

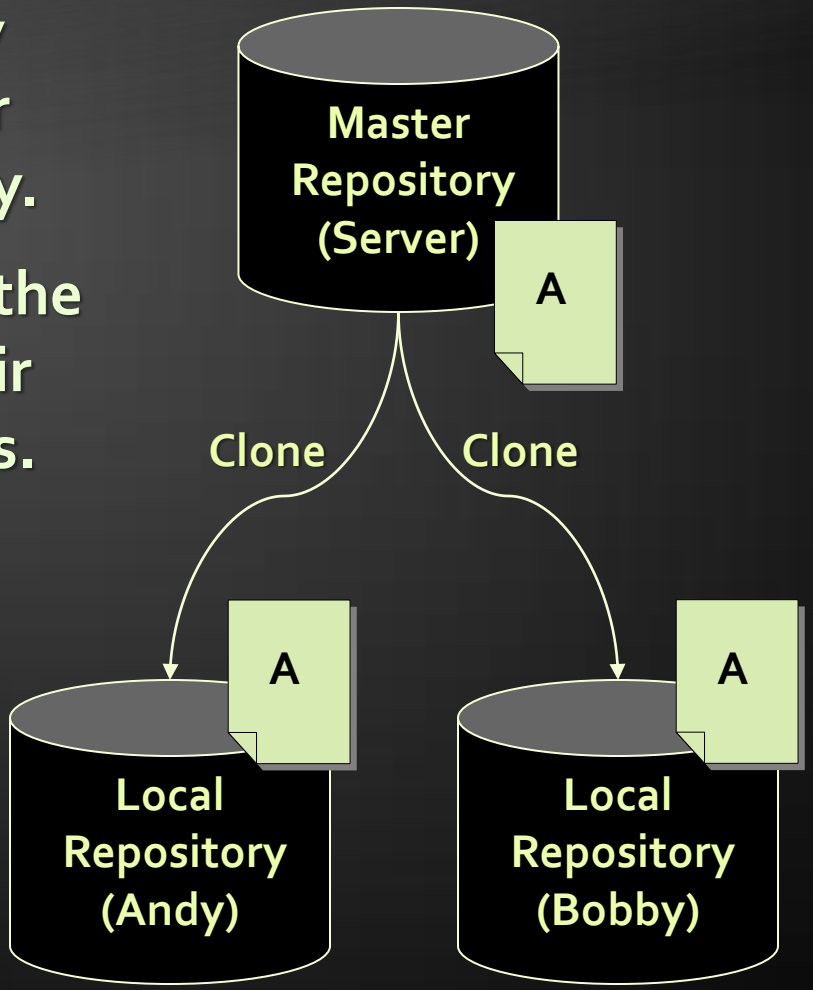
The "Distributed Version Control" Versioning Model



Distributed Version Control (1)

Andy and Bobby clone the master repository locally.

They both have the same files in their local repositories.

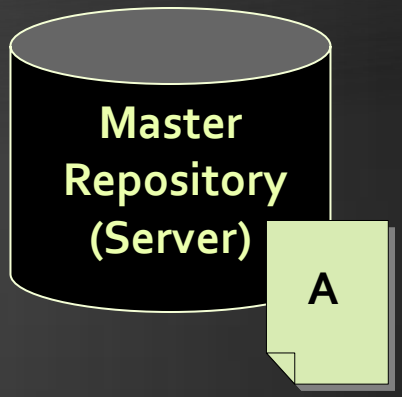


Andy

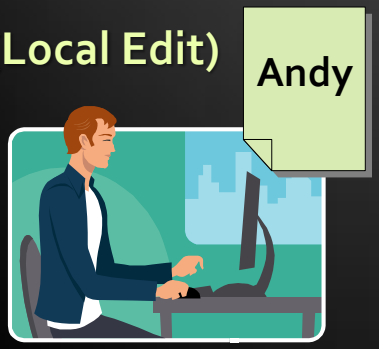
Bobby

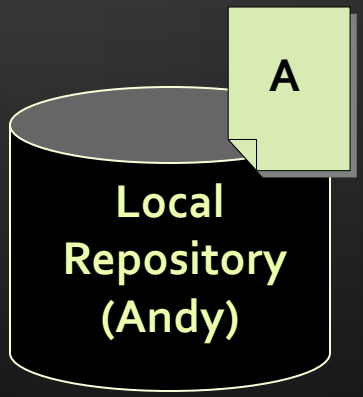
Distributed Version Control (2)

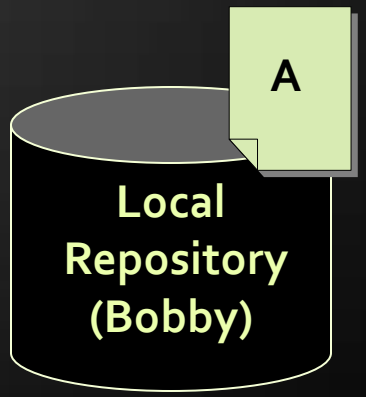
Andy and Bobby work locally on a certain file A.



(Local Edit)



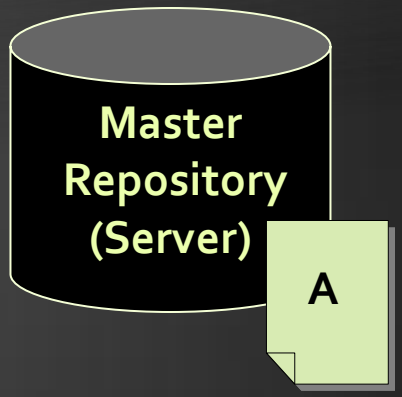


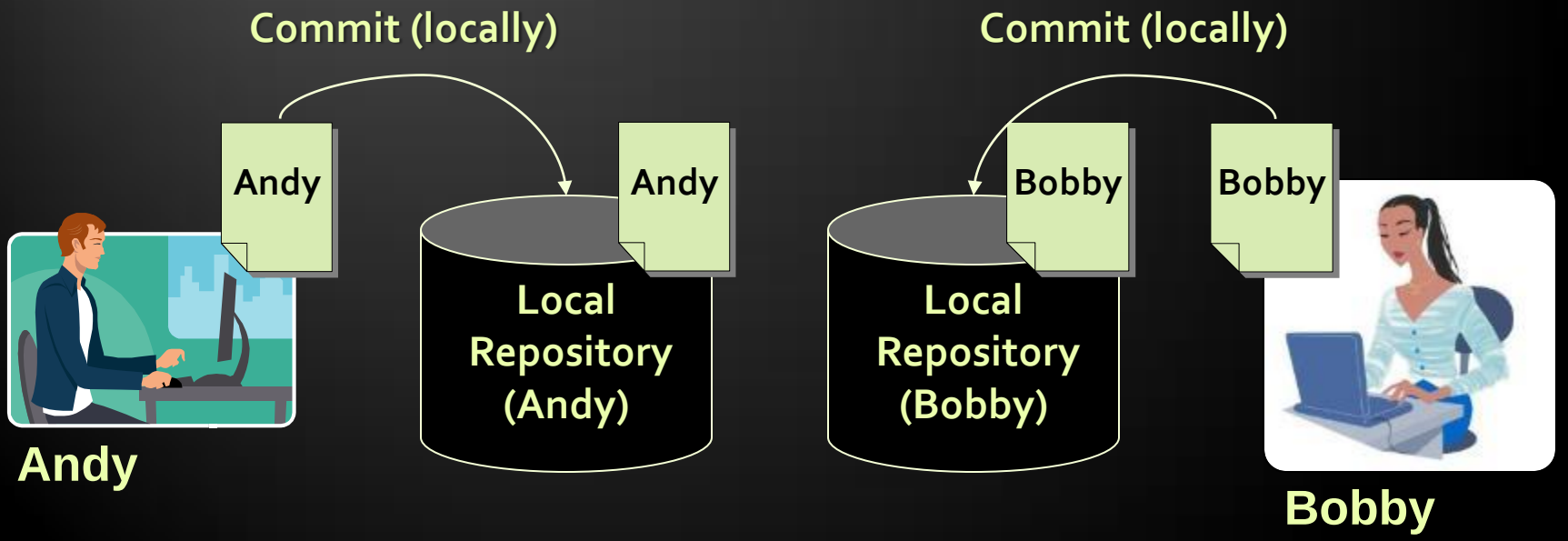


(Local Edit)

Distributed Version Control (3)

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

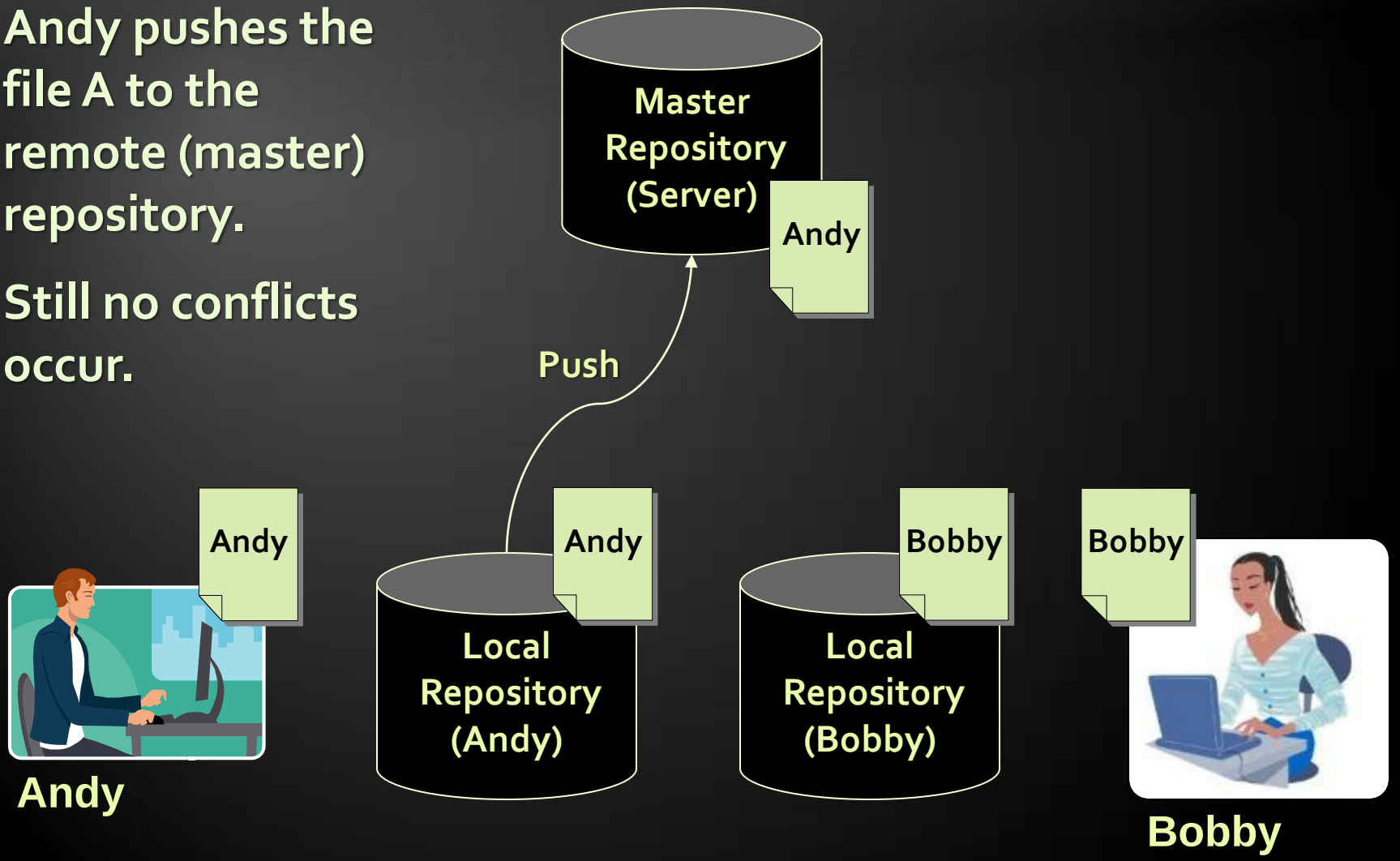




Distributed Version Control (4)

Andy pushes the file A to the remote (master) repository.

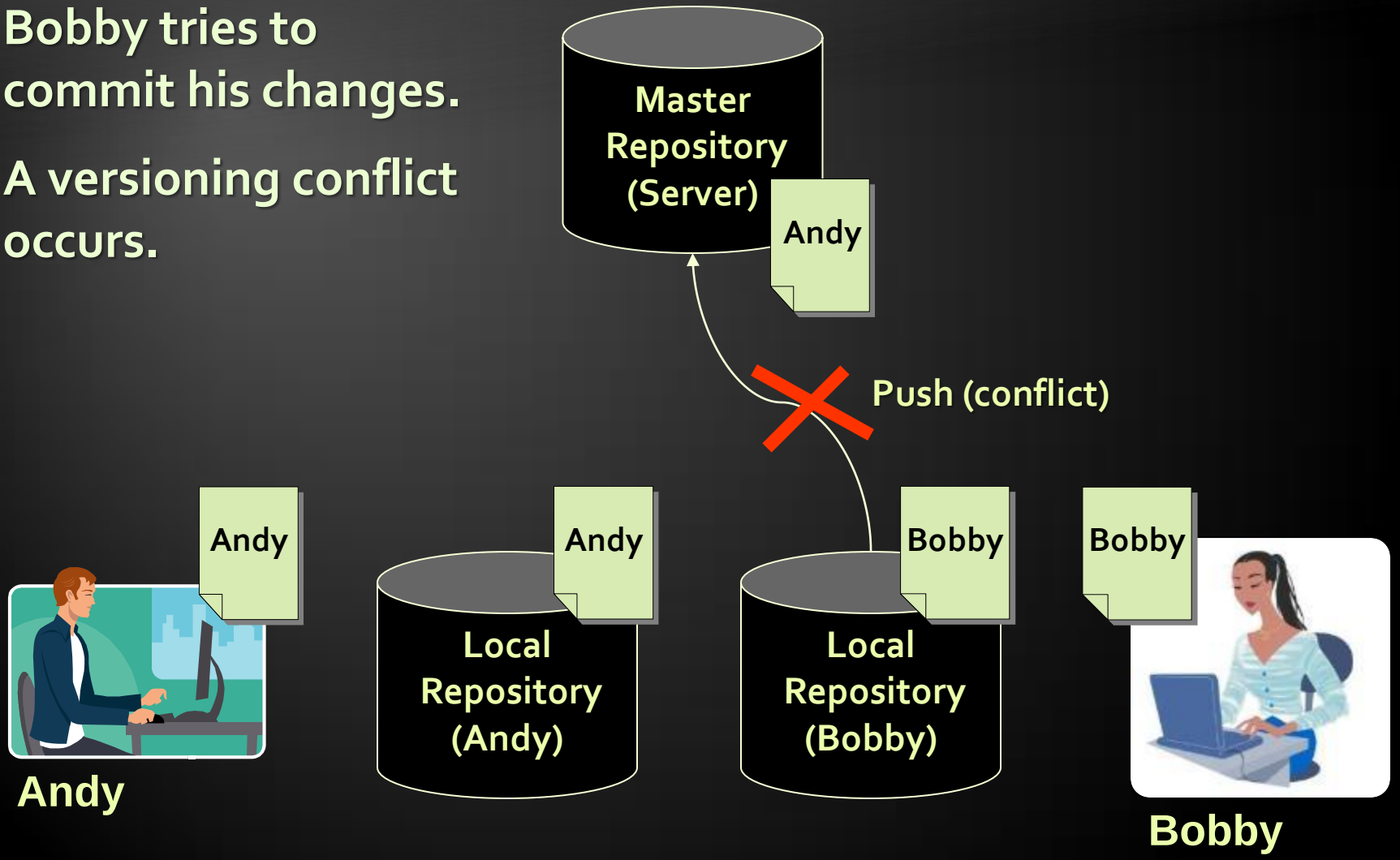
Still no conflicts occur.



Distributed Version Control (5)

Bobby tries to commit his changes.

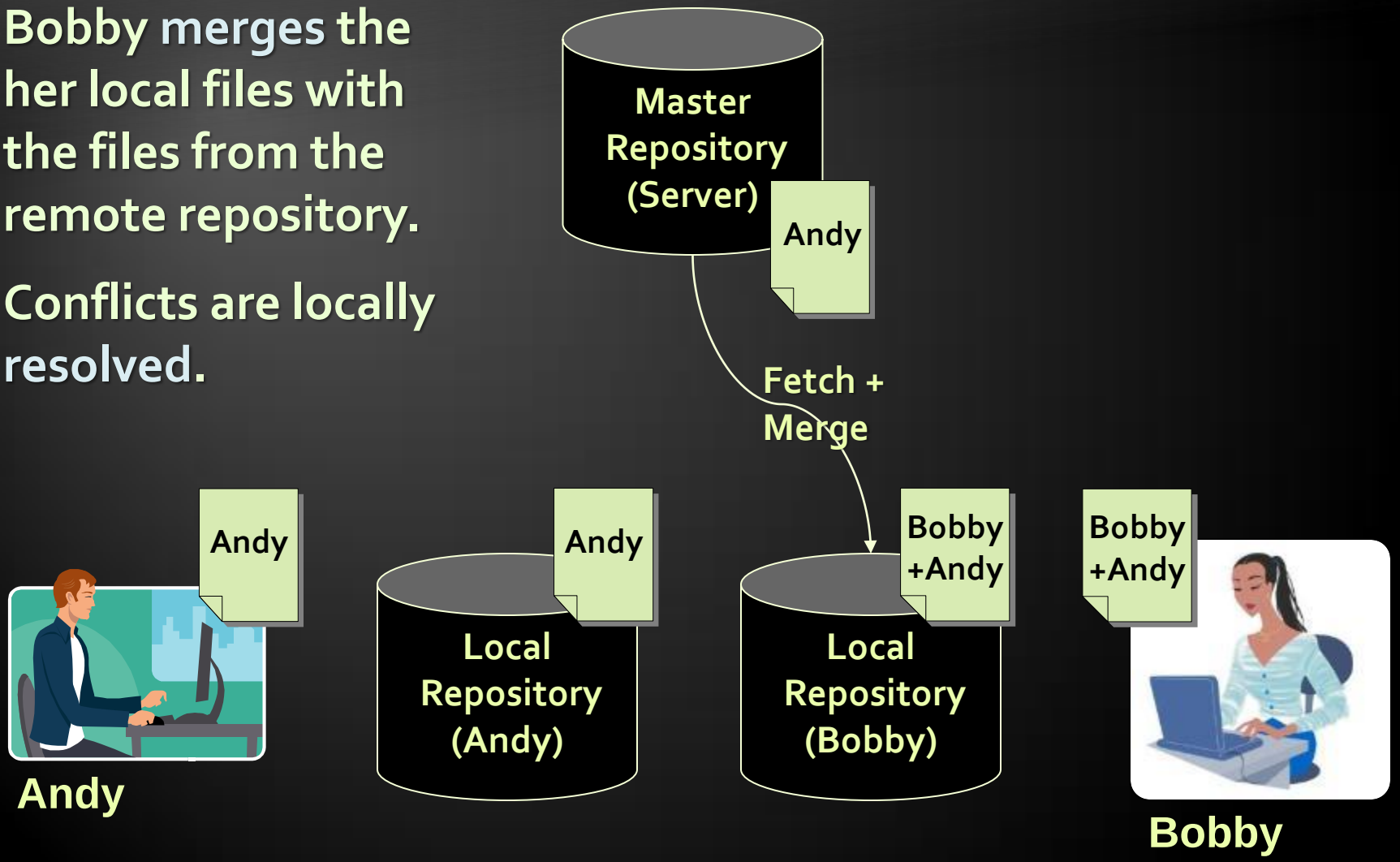
A versioning conflict occurs.



Distributed Version Control (6)

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

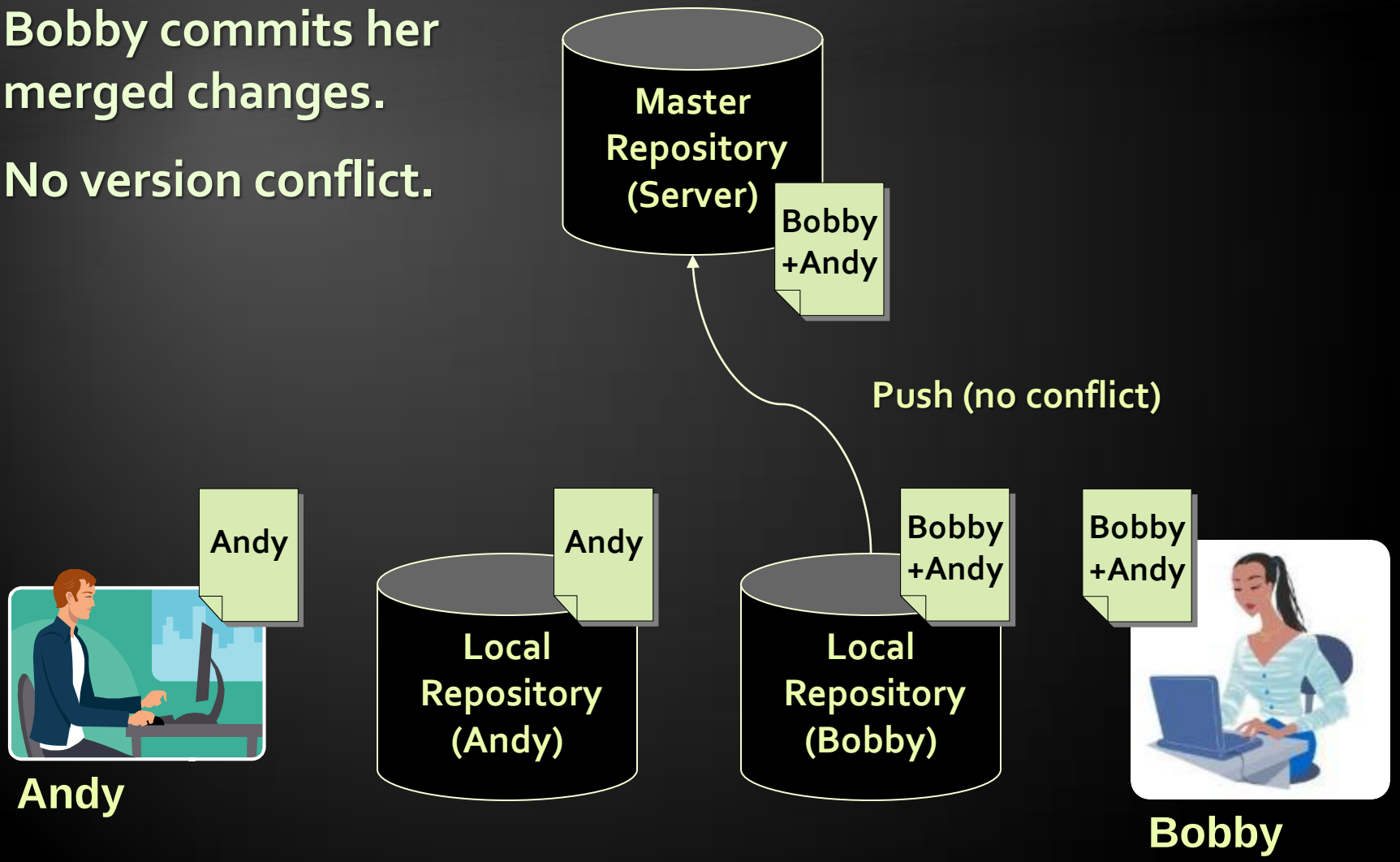
Conflicts are locally resolved.



Distributed Version Control (7)

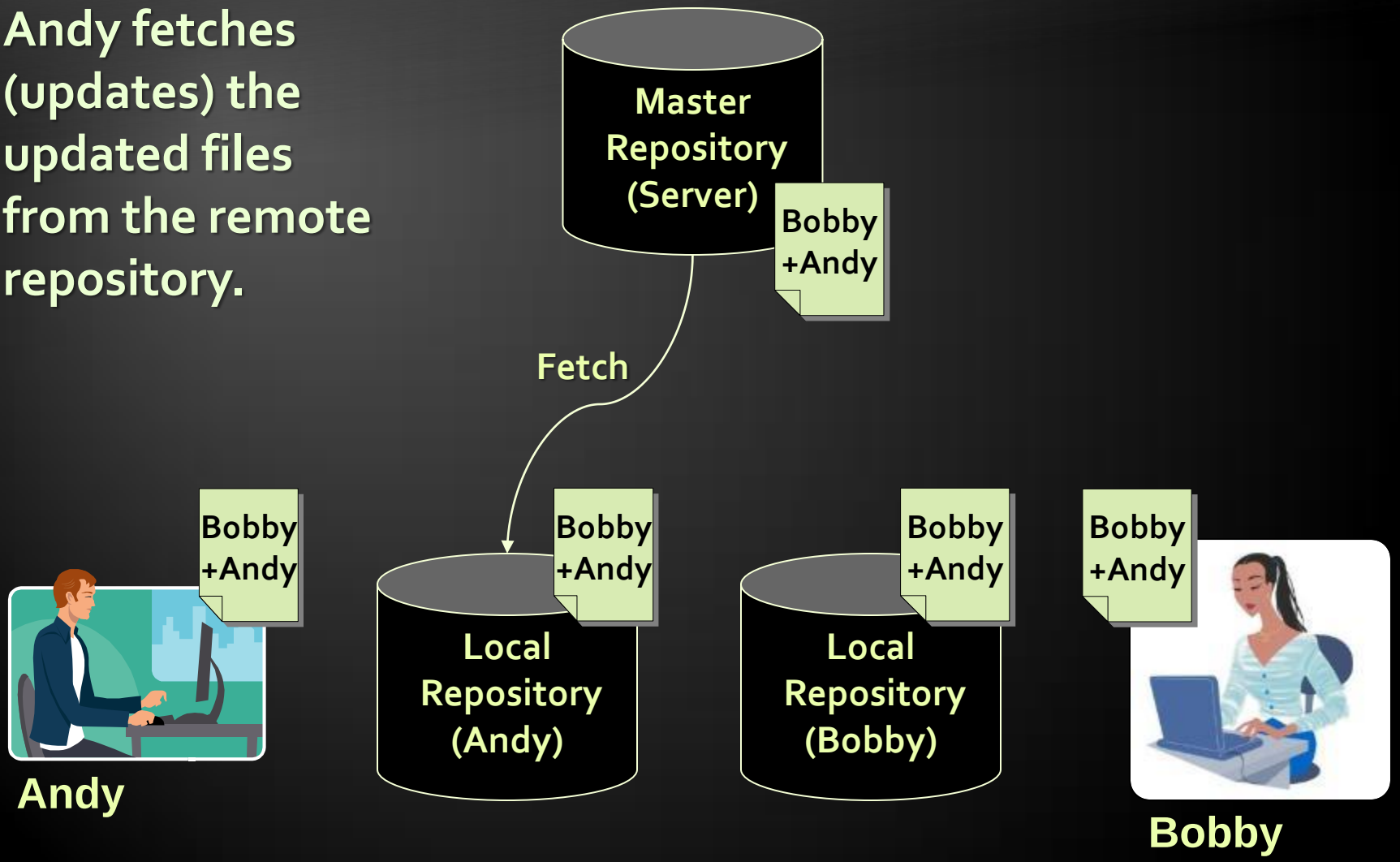
Bobby commits her merged changes.

No version conflict.



Distributed Version Control (8)

Andy fetches (updates) the updated files from the remote repository.

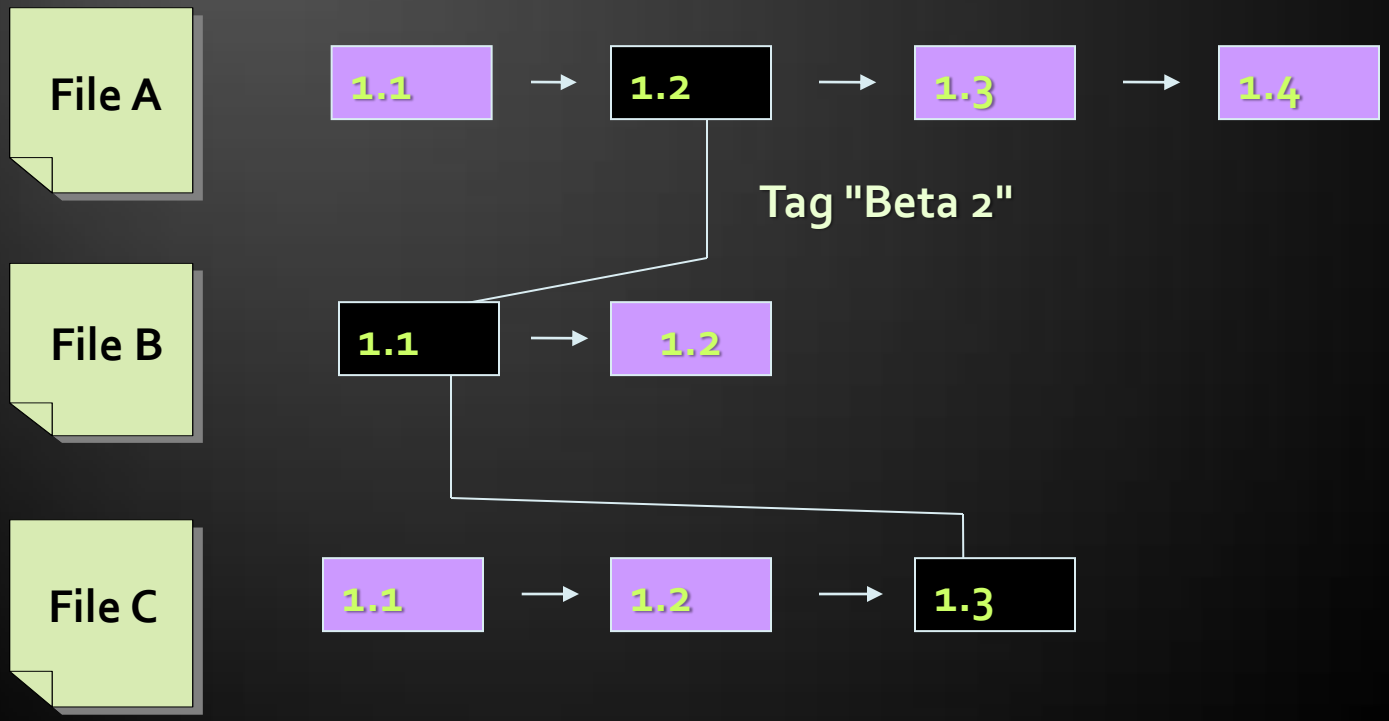


Telerik Academy

Tags and Branches

Tags

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



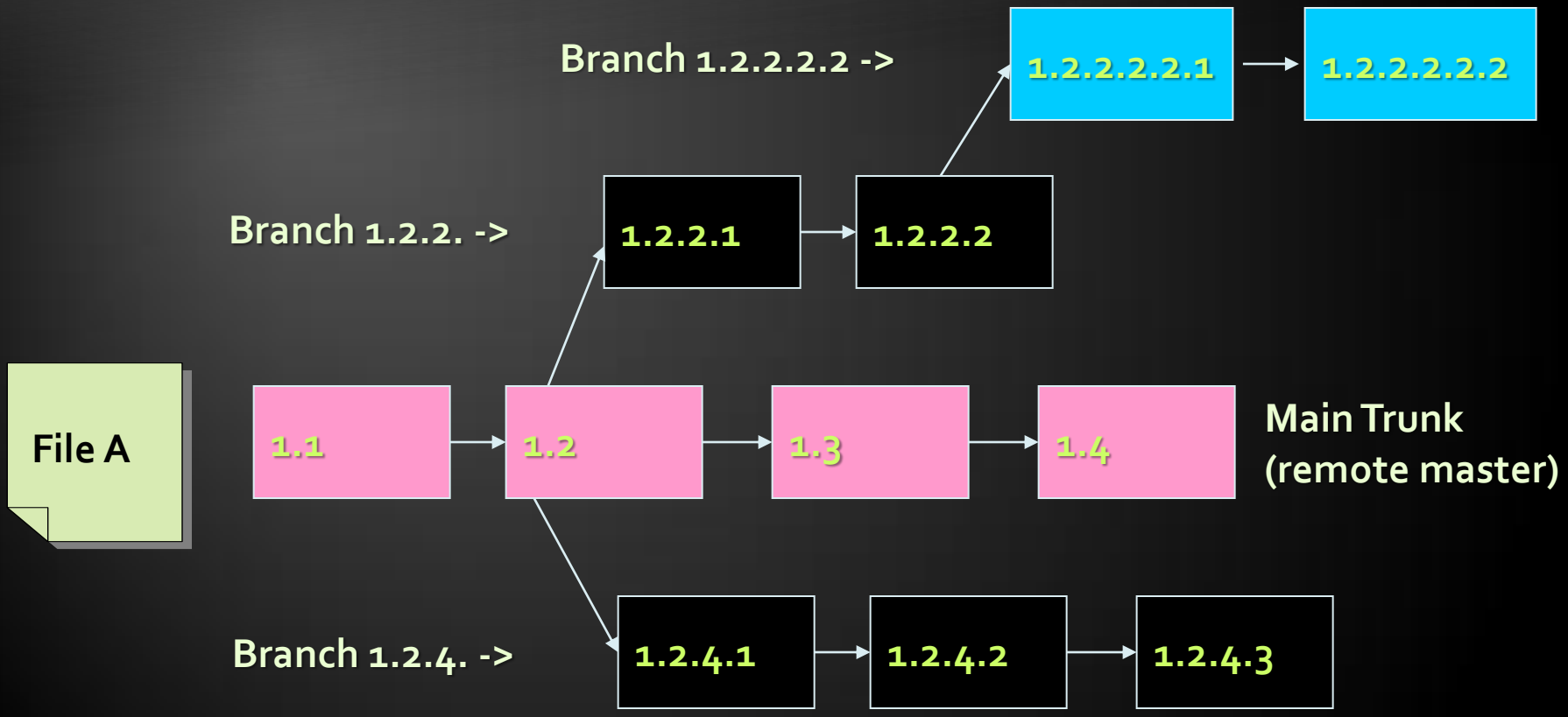
Branching

- Branching allows a group of changes to be separated in a development line
  - 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

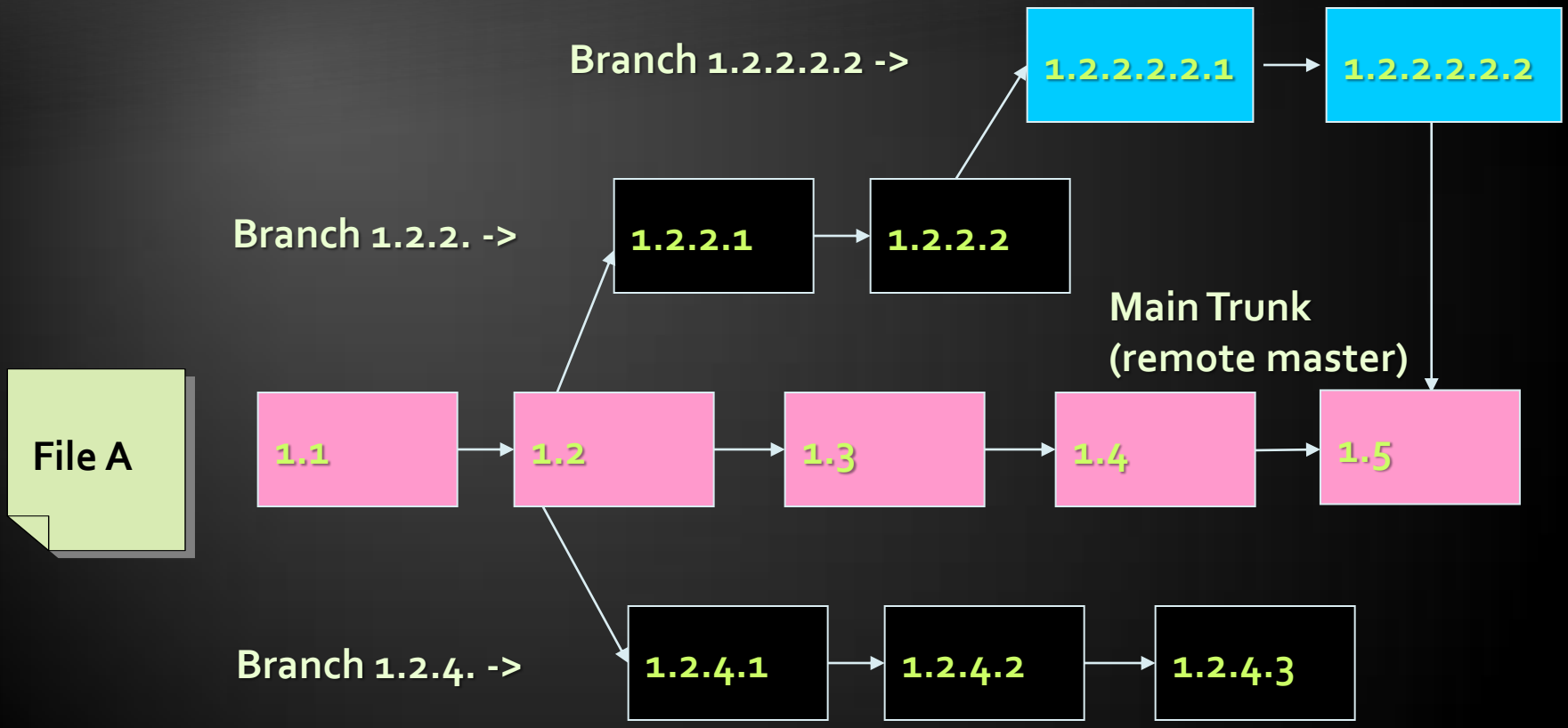
- Some companies work in separate branches
  - For each new feature / fix / task
- Once a feature / fix / task is completed
  - It is tested locally and committed in its branch
- Finally it should be merged into the main development line
  - Merging is done locally
  - Conflicts are resolved locally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

Branching – Example



Telerik Academy

Merging Branches – Example



Telerik Academy

GitHub

Live Demo

Software Configuration Management (SCM)

Questions?

Exercises

1. Play with GitHub. Work in teams of 3-10 people.
   - Register a Git repository in GitHub (one per team). Add your teammates to the project.
   - Upload a few of your projects (C# / HTML code / etc.).
   - Each team member: change something locally. Commit and push your changes into GitHub.
   - Intentionally make a conflict: each team member simultaneously edits one of the files and tries to commit. In case of conflict merge locally and commit.
   - Review the project history (change log) at GitHub.
   - Revert to a previous version and commit.