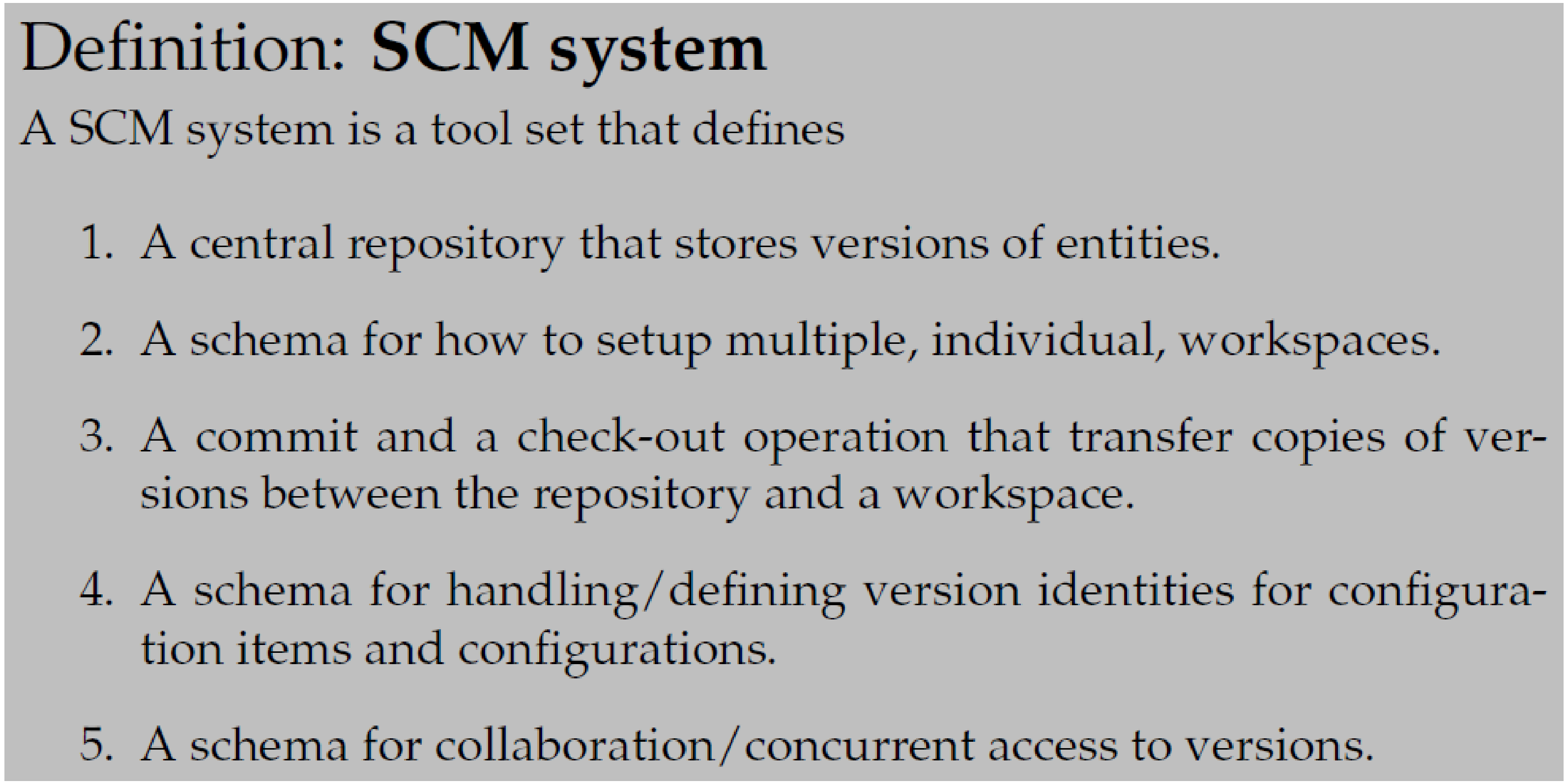
AARHUS UNIVERSITET

Software Engineering and Architecture

SCM Examples
Subversion and Git

AARHUS UNIVERSITET

The SCM Template



Definition: **SCM system**

A SCM system is a tool set that defines

1. A central repository that stores versions of entities.
2. A schema for how to setup multiple, individual, workspaces.
3. A commit and a check-out operation that transfer copies of versions between the repository and a workspace.
4. A schema for handling/defining version identities for configuration items and configurations.
5. A schema for collaboration/concurrent access to versions.

AARHUS UNIVERSITET

Subversion

1. The repository is a real database system with a database server. Several protocols can be used to communicate with the server: HTTP, HTTPS, SVN, and file. The first two are supported by the WebDAV protocol on Apache servers; the SVN is a special protocol supported by a supplied server application, and finally the 'file' protocol allows the clients to use a local folder as repository. The repository is a true database where versions are stored in binary format. Subversion also handles binary files more elegantly and efficiently than CVS.

2. A workspace is a standard folder structure.

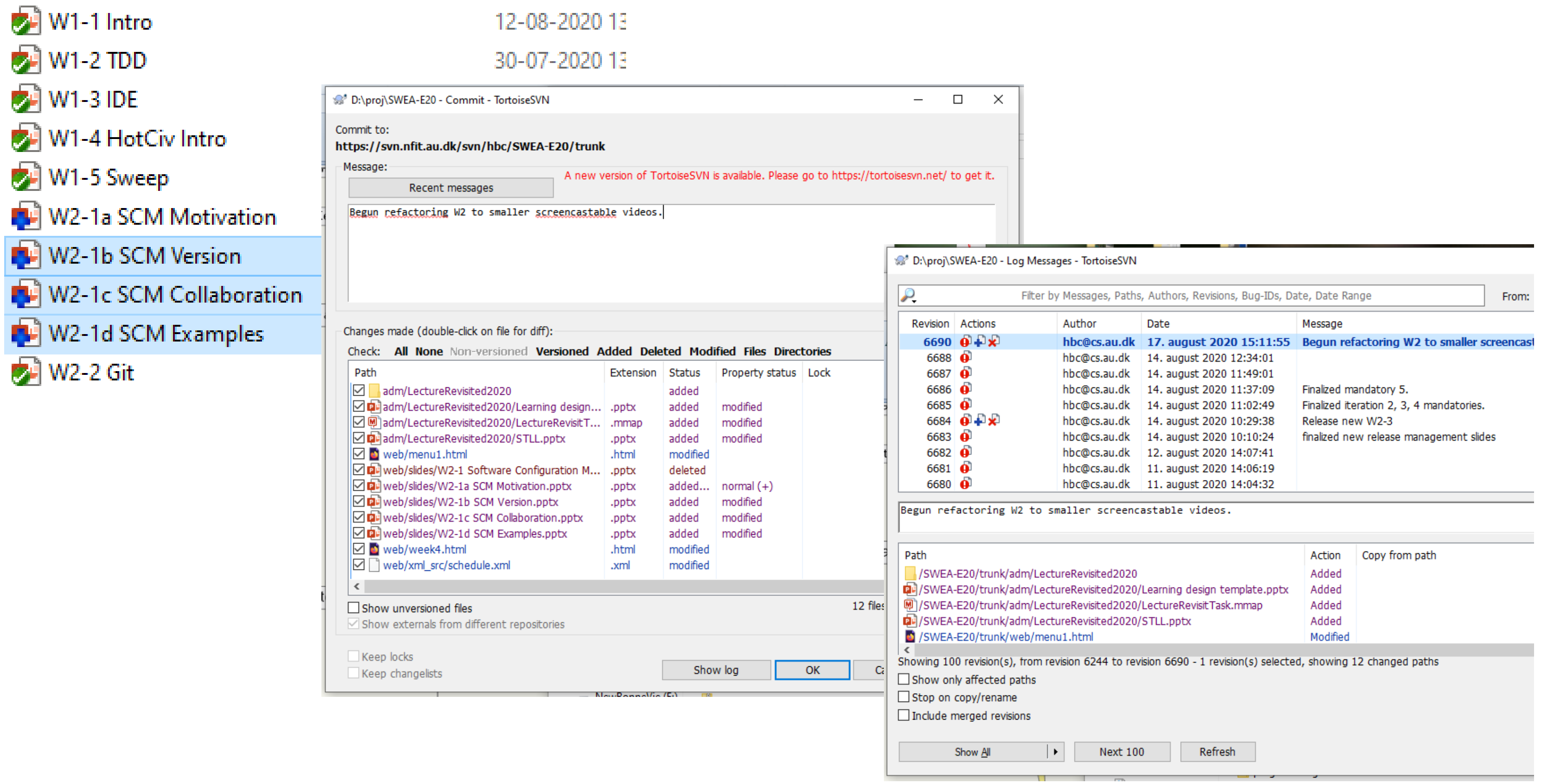
3. All svn commands are handled by the `svn` tool (or a file browser plug-in). The second parameter of `svn` is the operation to perform and follows more or less the CVS syntax.

4. Subversion can operate both on configurations as well as on individual configuration items; but conceptually it simply copies the complete configuration and assigns a new version identity. See below.

5. Subversion by default uses an optimistic concurrency model, but can also use a locking based model.

AARHUS UNIVERSITET

Examples



W1-1 Intro	12-08-2020 13
W1-2 TDD	30-07-2020 13
W1-3 IDE	
W1-4 HotCiv Intro	
W1-5 Sweep	
W2-1a SCM Motivation	
W2-1b SCM Version	
W2-1c SCM Collaboration	
W2-1d SCM Examples	
W2-2 Git	

D:\proj\SWEA-E20 - Commit - TortoiseSVN

Commit to:
https://svn.nfit.au.dk/svn/hbc/SWEA-E20/trunk

Message:

A new version of TortoiseSVN is available. Please go to https://tortoisesvn.net/ to get it.

```
Begun refactoring W2 to smaller screencastable videos.
```

Changes made (double-click on file for diff):

Check: All None Non-versioned Versioned Added Deleted Modified Files Directories

Path	Extension	Status	Property status	Lock
☑ adm/LectureRevisited2020		added		
☑ adm/LectureRevisited2020/Learning design...	.pptx	added	modified	
☑ adm/LectureRevisited2020/LectureRevisitT...	.mmap	added	modified	
☑ adm/LectureRevisited2020/STLL.pptx	.pptx	added	modified	
☑ web/menu1.html	.html	modified		
☑ web/slides/W2-1 Software Configuration M...	.pptx	deleted		
☑ web/slides/W2-1a SCM Motivation.pptx	.pptx	added...	normal (+)	
☑ web/slides/W2-1b SCM Version.pptx	.pptx	added	modified	
☑ web/slides/W2-1c SCM Collaboration.pptx	.pptx	added	modified	
☑ web/slides/W2-1d SCM Examples.pptx	.pptx	added	modified	
☑ web/week4.html	.html	modified		
☑ web/xml_src/schedule.xml	.xml	modified		

☐ Show unversioned files   12 files
☐ Show externals from different repositories

☐ Keep locks
☐ Keep changelists

D:\proj\SWEA-E20 - Log Messages - TortoiseSVN

Filter by Messages, Paths, Authors, Revisions, Bug-IDs, Date, Date Range   From:

Revision	Actions	Author	Date	Message
6690		**hbc@cs.au.dk**	**17. august 2020 15:11:55**	**Begun refactoring W2 to smaller screencast**
6688		hbc@cs.au.dk	14. august 2020 12:34:01	
6687		hbc@cs.au.dk	14. august 2020 11:49:01	
6686		hbc@cs.au.dk	14. august 2020 11:37:09	Finalized mandatory 5.
6685		hbc@cs.au.dk	14. august 2020 11:02:49	Finalized iteration 2, 3, 4 mandatories.
6684		hbc@cs.au.dk	14. august 2020 10:29:38	Release new W2-3
6683		hbc@cs.au.dk	14. august 2020 10:10:24	finalized new release management slides
6682		hbc@cs.au.dk	12. august 2020 14:07:41	
6681		hbc@cs.au.dk	11. august 2020 14:06:19	
6680		hbc@cs.au.dk	11. august 2020 14:04:32	

```
Begun refactoring W2 to smaller screencastable videos.
```

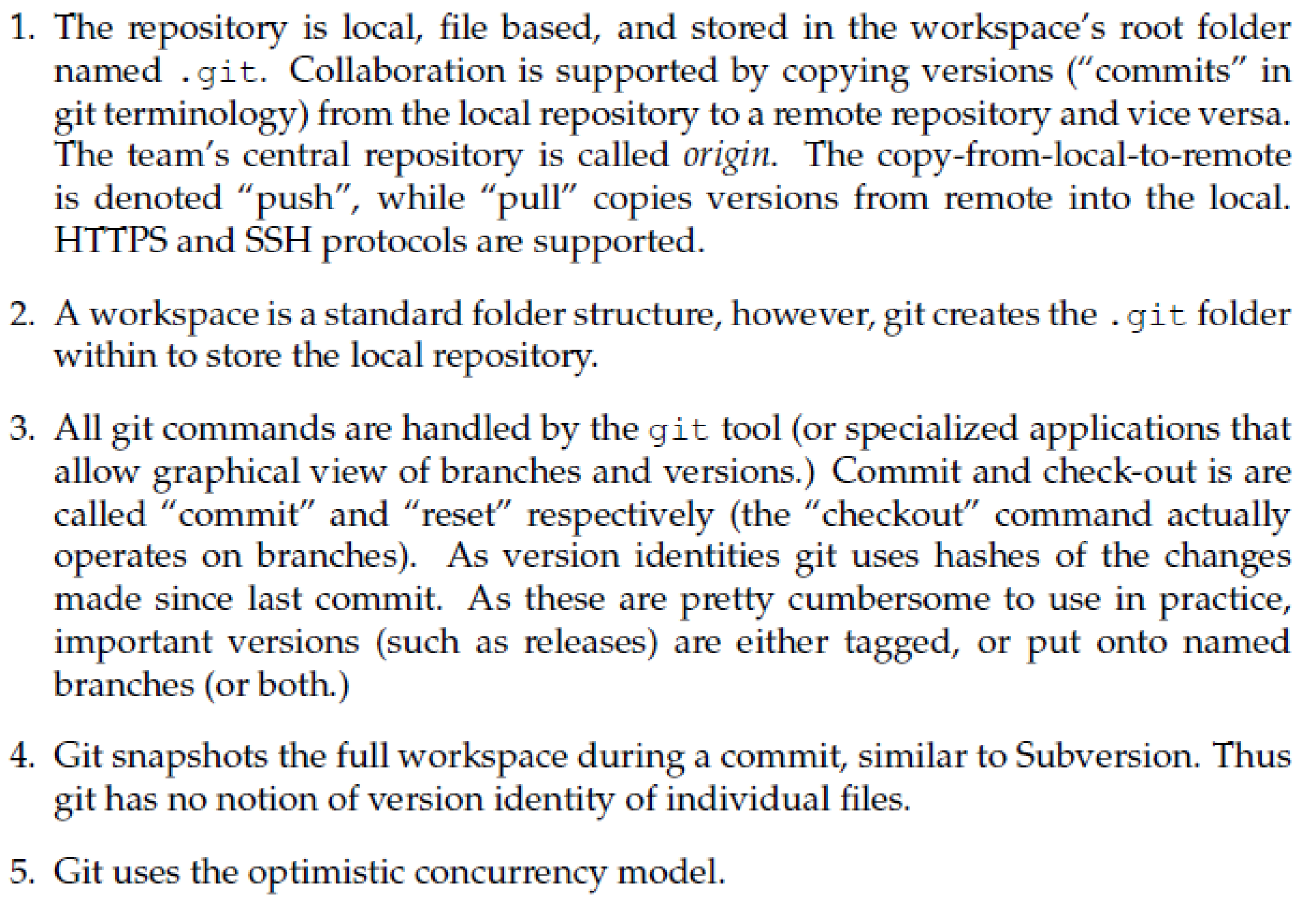
Path	Action	Copy from path
/SWEA-E20/trunk/adm/LectureRevisited2020	Added	
/SWEA-E20/trunk/adm/LectureRevisited2020/Learning design template.pptx	Added	
/SWEA-E20/trunk/adm/LectureRevisited2020/LectureRevisitTask.mmap	Added	
/SWEA-E20/trunk/adm/LectureRevisited2020/STLL.pptx	Added	
/SWEA-E20/trunk/web/menu1.html	Modified	

Showing 100 revision(s), from revision 6244 to revision 6690 - 1 revision(s) selected, showing 12 changed paths

☐ Show only affected paths
☐ Stop on copy/rename
☐ Include merged revisions

AARHUS UNIVERSITET

Git



1. The repository is local, file based, and stored in the workspace's root folder named `.git`. Collaboration is supported by copying versions ("commits" in git terminology) from the local repository to a remote repository and vice versa. The team's central repository is called *origin*. The copy-from-local-to-remote is denoted "push", while "pull" copies versions from remote into the local. HTTPS and SSH protocols are supported.

2. A workspace is a standard folder structure, however, git creates the `.git` folder within to store the local repository.

3. All git commands are handled by the `git` tool (or specialized applications that allow graphical view of branches and versions.) Commit and check-out is are called "commit" and "reset" respectively (the "checkout" command actually operates on branches). As version identities git uses hashes of the changes made since last commit. As these are pretty cumbersome to use in practice, important versions (such as releases) are either tagged, or put onto named branches (or both.)

4. Git snapshots the full workspace during a commit, similar to Subversion. Thus git has no notion of version identity of individual files.

5. Git uses the optimistic concurrency model.

AARHUS UNIVERSITET

Git Dist. Repository

- Git is a ***Distributed SCM system***
  - *Every workspace holds a complete copy of the repo**
- ‘git commit’ makes a check-in/commit
  - But to the *local repository (gobbling up your disk space!)*

- Thus we cannot collaborate? Yes we can because we can
- **Push**: Copy all changes from local repo to remote repo
- **Pull:** Copy all changes from remote repo to local repo

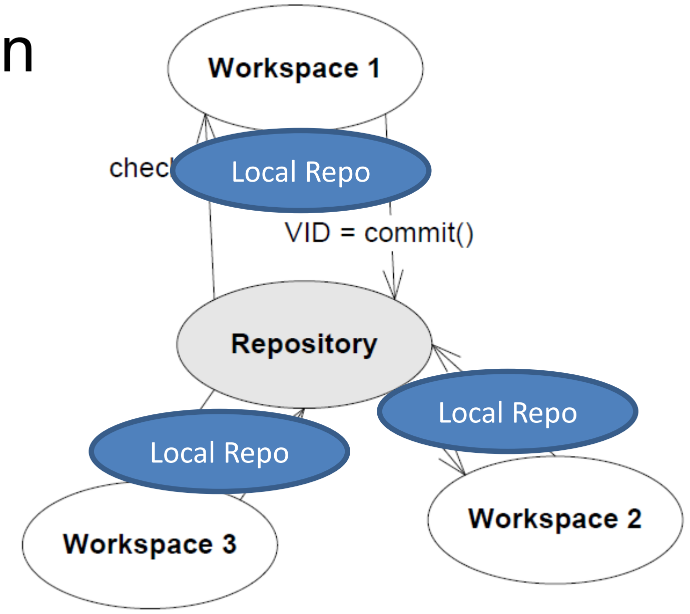
*) Not quite true: complete copy of subset of branches

Multiple Repositories

- The ‘common’ one is called *origin*
  - *Typically hosted at AU GitLab, BitBucket, GitHub, …*

- They can naturally form a chain
  - Local – Team – Company

- Exercise:
  - Pros?
  - Cons?



Git Staging Area

- One further complication
  - A modified file in workspace is **not considered modified** until it is added to the Git ***staging area***
- Thus the procedure is as this
  - Arne modifies hans.txt
  - Arne adds it to the staging area by *‘git add hans.txt’*
  - Arne commits to local repo: *‘git commit –m ”modified hans”’*
  - Arne pushes to remote repo: *‘git push’*
- Subversion equivalent
  - Arne modifies hans.txt
  - Arne commits: *‘svn commit –m ”modified hans”’*

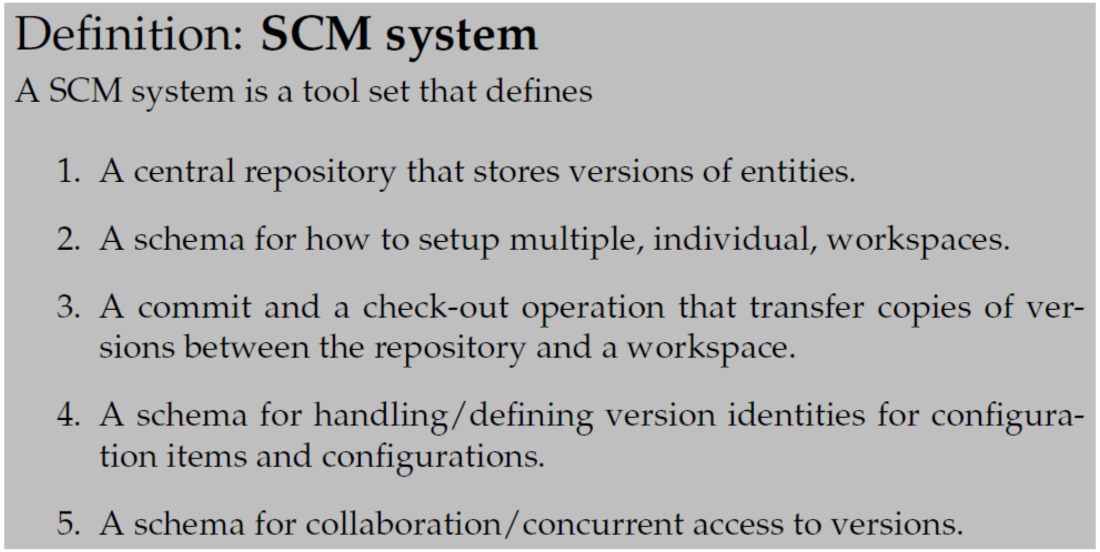
Why?

- The staging area (aka index) adds one extra layer of complication to the use of git.
- Why?
  - Fine-grained version control
- Scenario:
  - In iteration 755 I
    - Add feature x to my fabulous program
    - … which uses the 'doSuperStuff' method that I spotted a bug in!
  - Git
    - Add just the files related to bug fix to index; commit bugfix
    - Add rest of files to index; commit feature

AARHUS UNIVERSITET

Summary

- Why?
  - To enable
    - Release management and historical tracking
    - Collaboration in the team
- How?



Definition: SCM system

A SCM system is a tool set that defines

1. A central repository that stores versions of entities.
2. A schema for how to setup multiple, individual, workspaces.
3. A commit and a check-out operation that transfer copies of versions between the repository and a workspace.
4. A schema for handling/defining version identities for configuration items and configurations.
5. A schema for collaboration/concurrent access to versions.