



AARHUS UNIVERSITET

# **Software Engineering and Architecture**

SCM Versioning



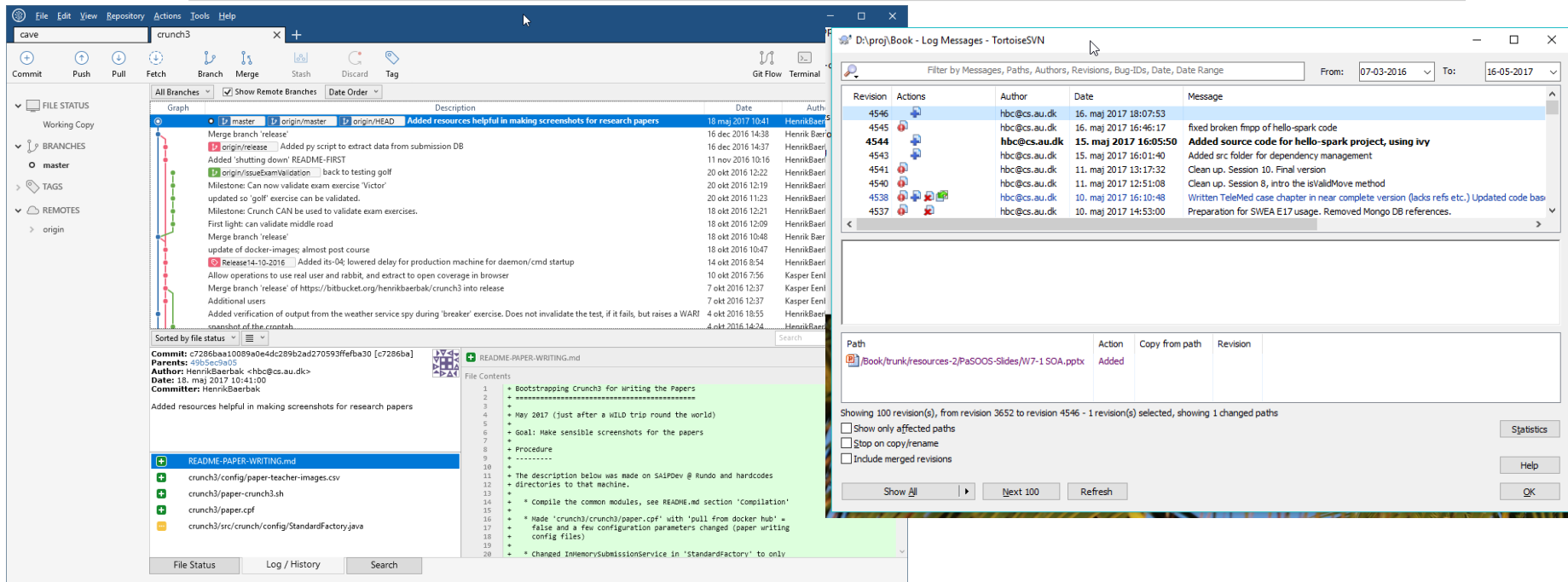
AARHUS UNIVERSITET

# Concepts I: Versioning

Theory of SCM

**Definition: Software configuration management**

Software configuration management (SCM) is the process of controlling the evolution of a software system.



The screenshot displays the TortoiseSVN interface with two windows open.

**TortoiseSVN Main Window:**

- File Status:** Shows the current working copy and branches. The 'master' branch is selected.
- Commit History:** A list of commits with columns for Revision, Date, and Author. The commit history is sorted by file status.
- Commit Details:** The selected commit (Revision 4546) is shown with its message: "Added resources helpful in making screenshots for research papers".
- File Contents:** The contents of the selected file, 'README-PAPER-WRITING.md', are displayed.

**Log Messages - TortoiseSVN Window:**

This window shows a detailed view of the commit history, filtered by messages, paths, authors, revisions, bug-IDs, date, and date range. The selected revision is 4546, dated 16. maj 2017 10:41, by Henrik Bærk Christensen.

Revision	Actions	Author	Date	Message
4546		hbc@cs.au.dk	16. maj 2017 10:41	Added resources helpful in making screenshots for research papers
4545		hbc@cs.au.dk	16. maj 2017 16:46:17	fixed broken frmp of hello-spark code
4544		hbc@cs.au.dk	15. maj 2017 16:05:50	Added source code for hello-spark project, using ivy
4543		hbc@cs.au.dk	15. maj 2017 16:01:40	Added src folder for dependency management
4541		hbc@cs.au.dk	11. maj 2017 13:17:32	Clean up. Session 10. Final version
4540		hbc@cs.au.dk	11. maj 2017 12:51:08	Clean up. Session 8, into the isValidMove method
4538		hbc@cs.au.dk	10. maj 2017 16:10:48	Written TeleMed case chapter in near complete version (lacks refs etc.) Updated code base
4537		hbc@cs.au.dk	10. maj 2017 14:53:00	Preparation for SWEA E17 usage. Removed Mongo DB references.

# Configuration

## Definition: Configuration item

A configuration item is the atomic building block in a SCM system. That is, the SCM system views a configuration item as a whole without any further substructure. A configuration item is identified by a name.

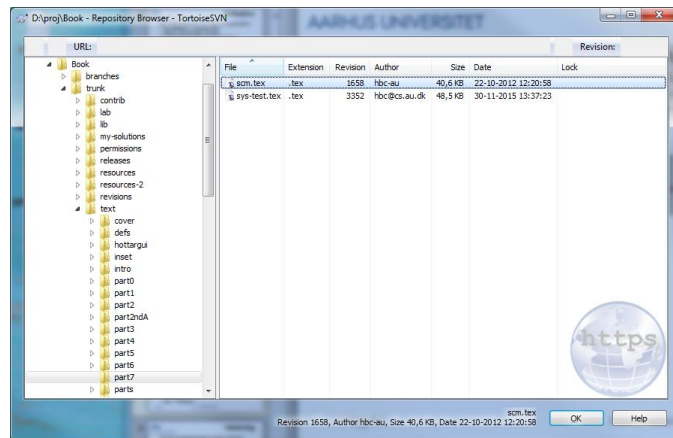
Read: File

## Definition: Configuration

A configuration is a named hierarchical structure that aggregates configuration items and configurations.

Read: Folder

- Think *Composite Pattern*
- *Git and SubVersion uses*
  - *File = Configuration Item*
  - *Folder = Configuration*



# Versioning

- The main purpose is to track evolution: *time focus*
- Another term for SCM is *Version Control*

## Definition: **Version**

A version,  $v_i$ , represents the immutable state of a configuration item or configuration at time  $t_i$ .

- To give us a ‘handle’/‘name’ of a version we need

## Definition: **Version identity**

A version is identified by a version identity,  $v_i$ , that must be unique in the SCM system.



# Version Examples

The image displays two overlapping windows illustrating version control systems. The background window is TortoiseSVN, showing a log of revisions for a project named 'Book'. The foreground window is a Git GUI, showing a commit history and a file diff.

**TortoiseSVN Log:**

Revision	Author	Date	Message
4546	hbc@cs.au.dk	16. maj 2017 18:07:53	fixed broken fmp of hello-spark code
4545	hbc@cs.au.dk	16. maj 2017 16:46:17	Added source code for hello-spark project, using ivy
4544	hbc@cs.au.dk	15. maj 2017 16:05:50	Added src folder for dependency management
4543	hbc@cs.au.dk	15. maj 2017 16:01:40	
4541	hbc@cs.au.dk	11. maj 2017 13:17	
4540	hbc@cs.au.dk	11. maj 2017 12:51	
4538	hbc@cs.au.dk	10. maj 2017 16:10	
4537	hbc@cs.au.dk	10. maj 2017 14:53	

**Git GUI:**

The Git GUI window shows a commit history with columns for Date, Author, and Description. A commit hash is highlighted: c7286ba. The file diff view shows changes to 'README-PAPER-WRITING.md'.

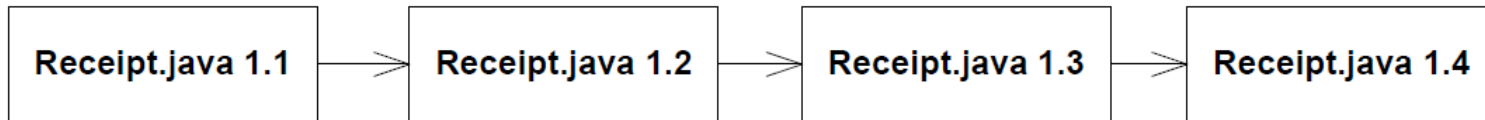
**Text Overlay:**

SubVersion: Global Incrementing Integer  
Git: Hashes/Fingerprints  
Older systems: Dewey numbers (like: 4.3.2.2)

- Evolution is tracked over time

## Definition: Version graph

A version graph is an oriented graph that shows the ancestor-relation between versions of an entity.



- One advantage of integers/dewey over hashes
  - Version 7 is *before* Version 8 and Version 21987
  - Version 8 is *after* Version 3

# Check-in/Check-out

- Operations

- Check-in / Commit: Snapshot in time **to** repository
- Check-out / Update: Snapshot in time **from** repository

## Definition: Commit

A commit is an operation that

1. generates a new, unique, version identity for the given entity.
2. stores a copy/snapshot of the entity under this identity.

Sigh! Git uses term  
'checkout' for something  
completely different!

## Definition: Check-out

A check-out is an operation that, given a unique version identity, is able to retrieve an exact copy of an entity as it looked when the given version identity was formed during a commit.

*Note: Versions cannot  
be deleted!*

## Definition: Repository

The repository is a central database, maintained and controlled by the SCM system that stores all versions of all controlled entities.

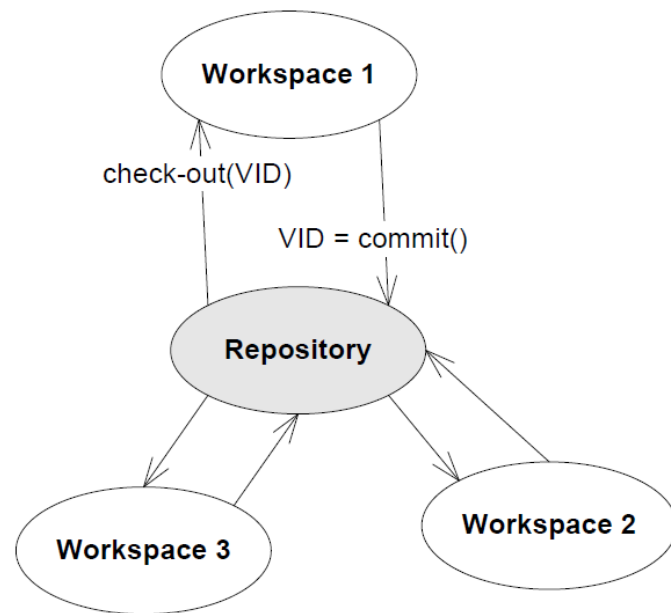


# Workspace

## Definition: Workspace

A workspace is a local file system in which individual versions of entities can be modified and altered. Only one version of a given entity is allowed at the same time in the workspace.

- Each developer has his/her own workspace(s) in which modifications are made
- Repository is shared, collaboration is handled through it...



- OK – what have we got?
  - Versions made by commits into the repository; they form a version graph of identified versions that form a graph. This graph reflects the evolution over time.
  - I can checkout any version into my workspace for review.
- Basically **version/release management**
  - I can reproduce the exact codebase as it looked when I gave it to the customer...
  - if I
    - a) remember to commit it and
    - b) can remember the version identity