Quality Management

Introduction to Systems Engineering
I2ISE

Introduction

- What is *Quality Management*?
- Reviews
- Configuration control
- Subversion/Github

What is a Quality Management?

- Quality Management (QM) is a set of activities performed to ensure that quality is
  - Planned
  - Controlled
  - Assured
  - Improved

- A couple of activities/tools: Reviews , version control and Subversion/Github

Errors

- Test is good in finding errors implemented by mistake
- Review and configuration management is to prevent errors being implemented at all!

QM - reviews

- A *review* is the activity of looking through proposed work prior to its' commitment.

- You can (practically) review anything
  - Code, diagrams
  - Documents
  - Processes (e.g. *Scrum retrospective*)
  - …

- For reviews to be effective, they must be *structured*

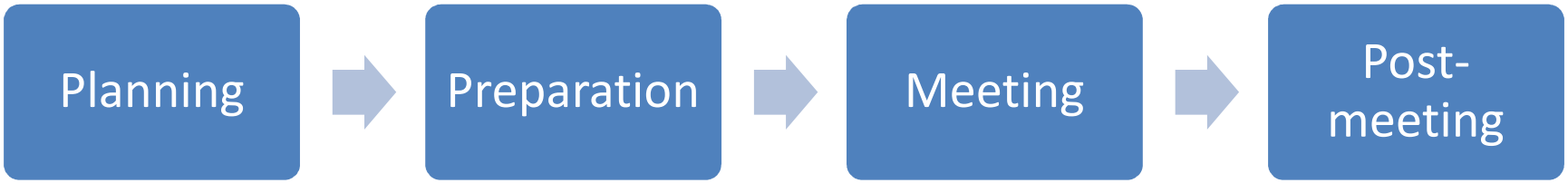
Reviews - goal

- The *goal* of a review: **Release of the item under review**

- How is the goal *supported* by the review?
  - By *constructive critisism* on the review item
  - By finding potential quality problems
  - By ensuring corrective action is taken

- How is the goal *obstructed* by the review?
  - By using it to prove you're smarter than everybody else
  - By establishing yourself as a leader
  - By pressing your preferred solution

Reviews – phases

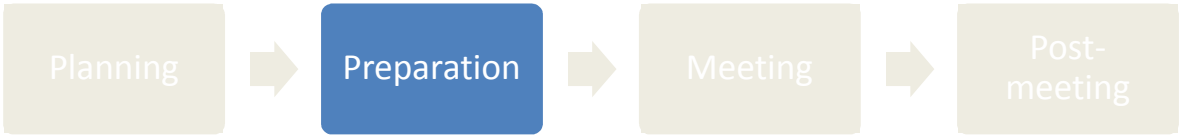


Reviews – planning

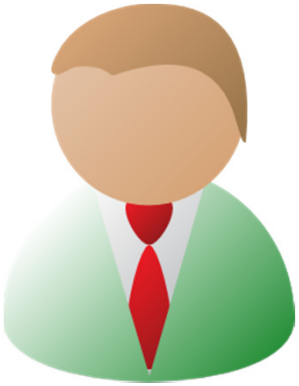
- What should be *planned*?
  - *What* are we going to review?
  - Who are the reviewers?
  - *When, where* and *how* will the review take place?
  - How is the document and supplementary material *distributed*?
  - Who will *chair* the review meeting?

Reviews – preparation (owners)

- How should the *document owners* prepare?
  - Make sure the review item is frozen for review
  - Practicals: Book meeting room, ensure AV equipment is present, …
  - Distribute review item etc. to review opponents along with agenda and venue
  - Define the desired *roles* (review leader, secretary)
  - …
- How should the *reviewer* prepare?
  - Read the review item thoroughly – distribute roles (spelling, diagrams, …)
  - Prepare overall and detailed critique
  - Find relevant sources etc.



The document owners' roles



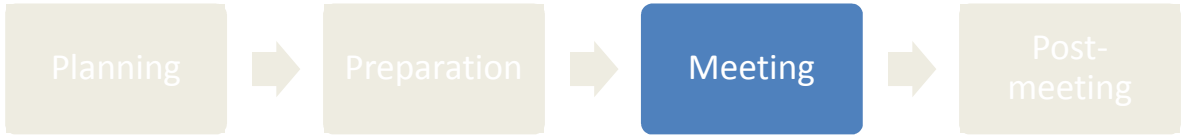
- Review *leader* (chair) ensures…
  - agenda for review is issued and kept
  - everyone is heard during review
  - focus during the meeting
  - delegation of action items

- Review *secretary* ensures
  - distribution of review item, agenda, …
  - creation and distribution of Minutes of Meeting (MoM)
  - changes are captured
  - review item is updated iaw. review (assigns action items)
  - sufficient coffee!

Planning → **Preparation** → Meeting → Post-meeting

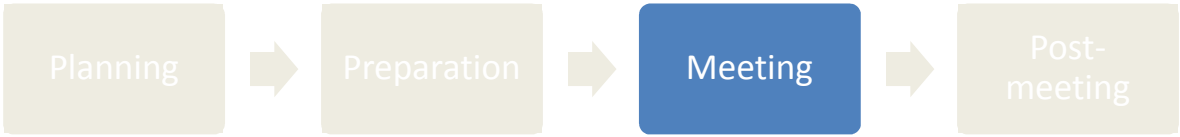
The reviewers' roles

- Some ground rules for the reviewers
  1. Be prepared
  2. Be friendly and empathic
  3. Behave
  4. Point to issues, not solutions
  5. Avoid discussions on style (taste)
  6. Stick to the subject matter



Review: The agenda

- The review meeting is best conducted along an agenda, e.g.
  1. Welcome, opening remarks (*chair*)
  2. General remarks (*opponents*)
  3. Detailed run-through of review item(*opponents*)
  4. Conclusion (*chair, secretary*)
  5. Actions to be taken - rework, corrections (*all*)
  6. Closing remarks (*all*)



Reviews – post-meeting

- Make sure Minutes of Meetings are distributed ASAP
- Make necessary corrections to the review item
- Decide on the action to take now: Call new review or release the review item

Configuration Management

- Another important activity is *Configuration Management*

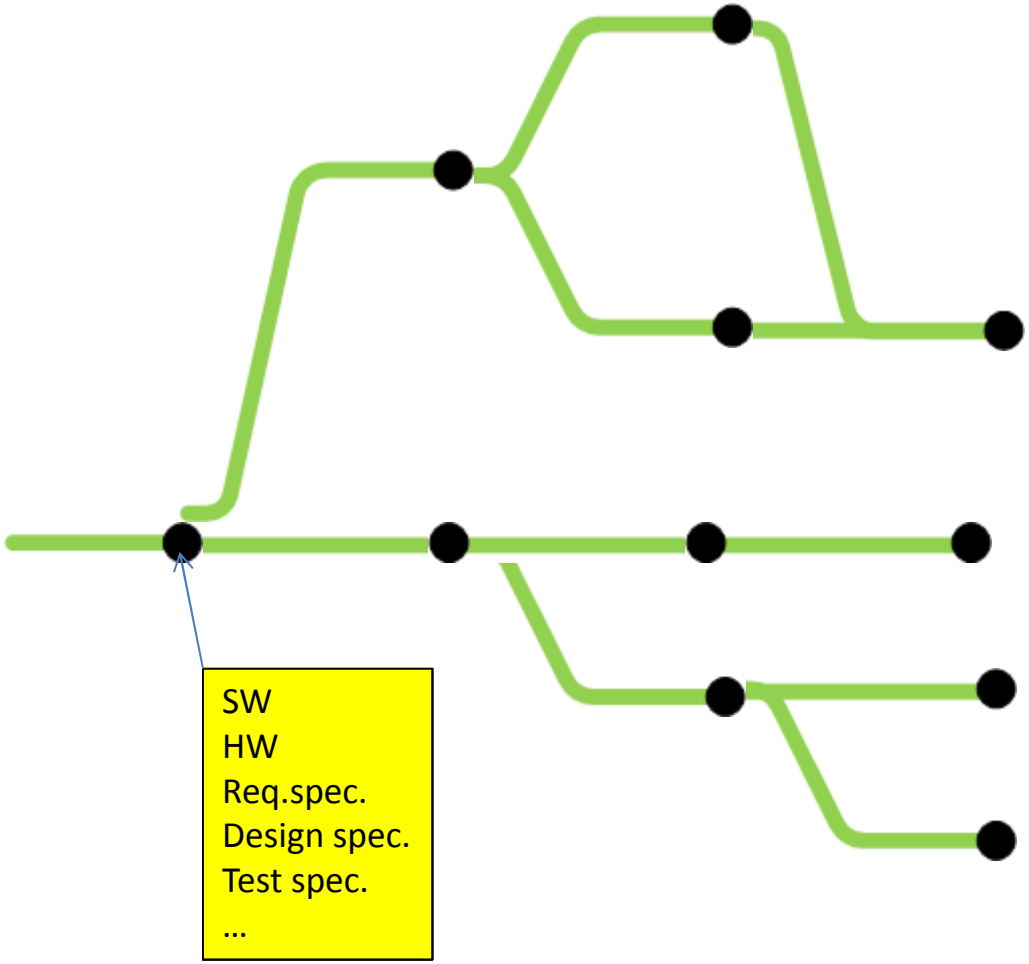
- The purpose of configuration management is to
  - Capture the *baseline* of a given (version) of a product
  - Ensure that a given product can be re-created from scratch

Baseline!

- Agreed-to information that defines and establishes the attributes of a product at a point in time and serves as the basis for defining change

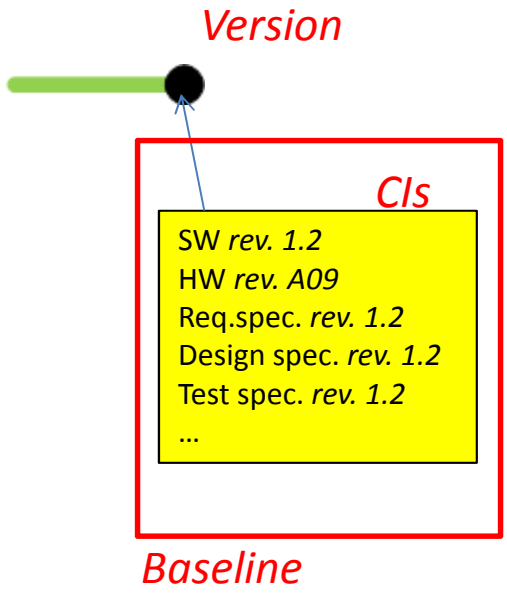
- Goals
  - to handle changes related to baseline
  - to ensure documentation and product artifacts fits together
    - Where do I find documentation that fits with product release x.y.z?

Configuration Management



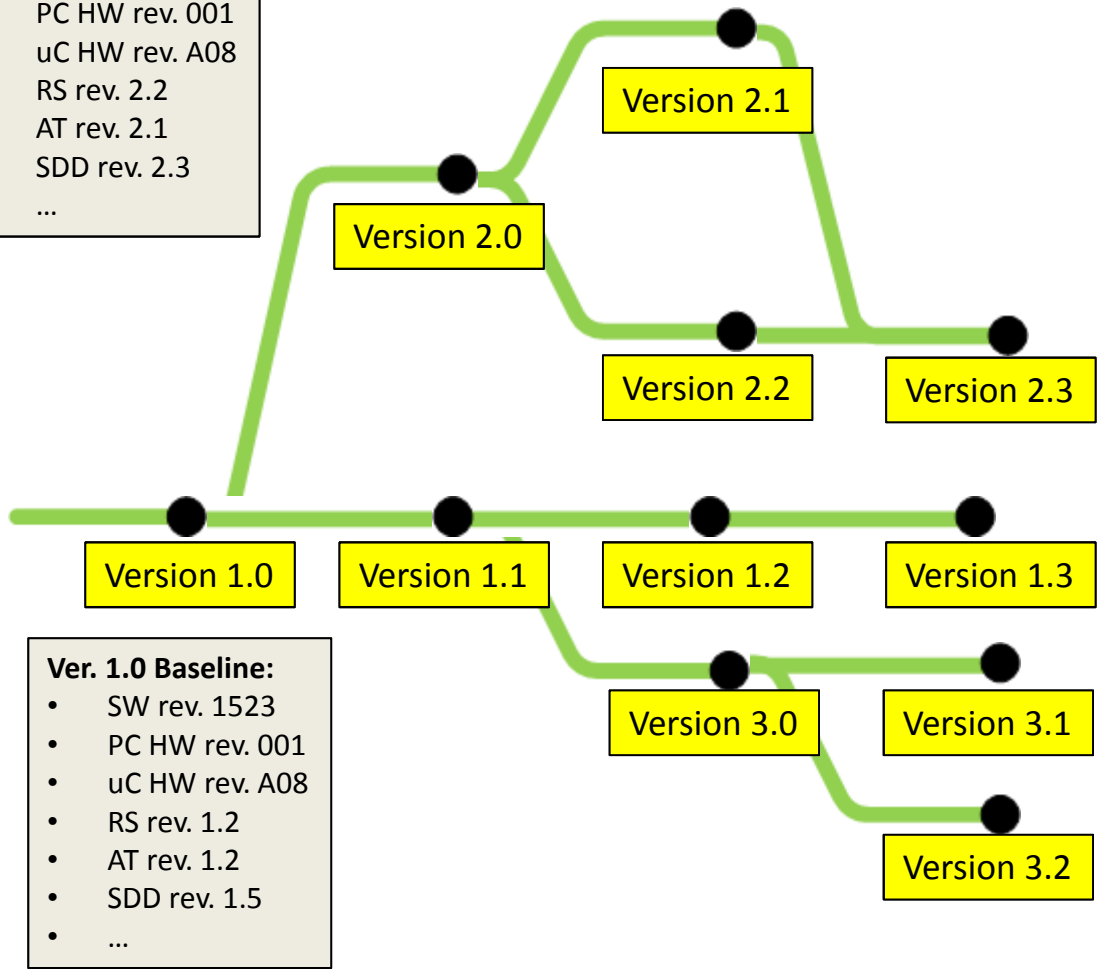
Configuration Management

- What you need is *configuration management*
  - A way to control what *configuration items (CIs)* (documents, HW, SW, tools, …) in what *revision* that apply to a given *version* of the product
- Collectively, this is known as a *baseline*
  - Typically defined by a top-level document that calls out the different versions of CIs
- The baseline must contain everything necessary to rebuild the version from scratch



Configuration Management

Ver. 2.0 Baseline:
- SW rev. 2301
- PC HW rev. 001
- uC HW rev. A08
- RS rev. 2.2
- AT rev. 2.1
- SDD rev. 2.3
- …

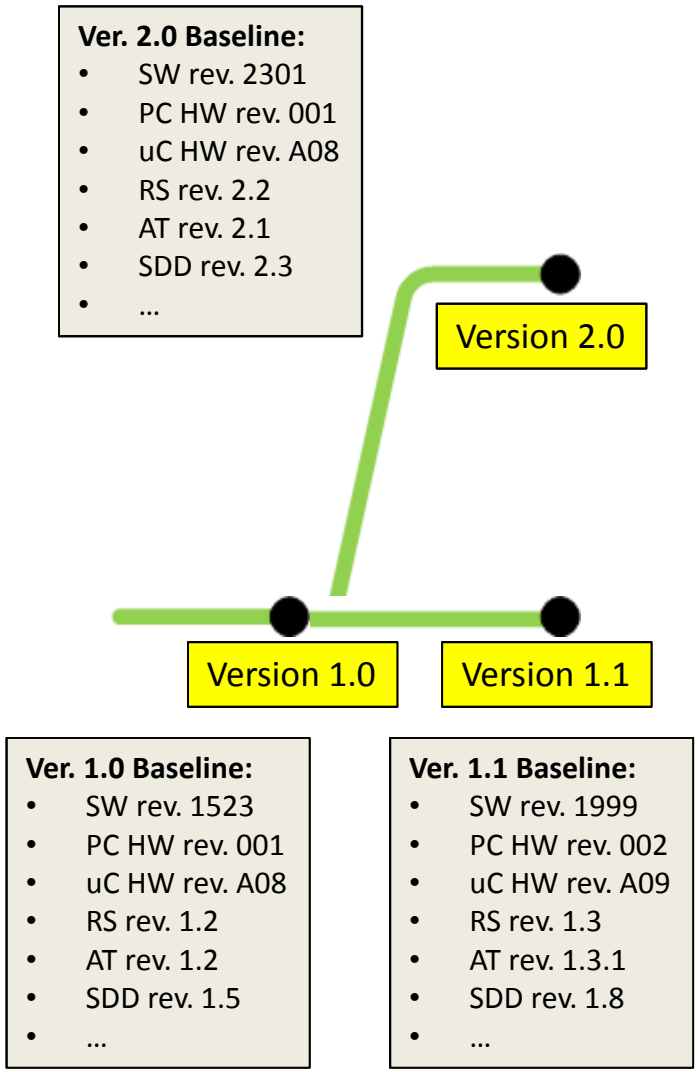


Ver. 1.0 Baseline:
- SW rev. 1523
- PC HW rev. 001
- uC HW rev. A08
- RS rev. 1.2
- AT rev. 1.2
- SDD rev. 1.5
- …

- Versions are defined
- Each *version* defines its own *baseline*

Configuration Management

- Note that…
  - *Versions* are defined at planning-time
  - *Baselines* are defined when a version is completed (released)



CM – version control systems

- There exists a variety of version control systems
  - Systems that allow you to get, update, submit, track and revert revisions of a document/source file
  - Examples: Git, Subversion, PVCS, Dropbox, …
- Indispensable for a number of reasons:
  - Concurrent work
  - Versioning and revision control
- Example: Subversion (http://subversion.tigris.org/)

CM – Subversion

- *History*: All earlier revisions of a document are maintaied
- *Availability*: Documents are securely accessible in a single place
- *Sharing*: Several people can contribute to a document

CM – Subversion basics

- Somewhere, someone (maybe you?) have created a *repository*
  - Ask Google how if you're interested

- First (and only once), you ***check out*** the repository
  1. Perform an **`SVN checkout`** operation – this will give you a *working copy* of the repository

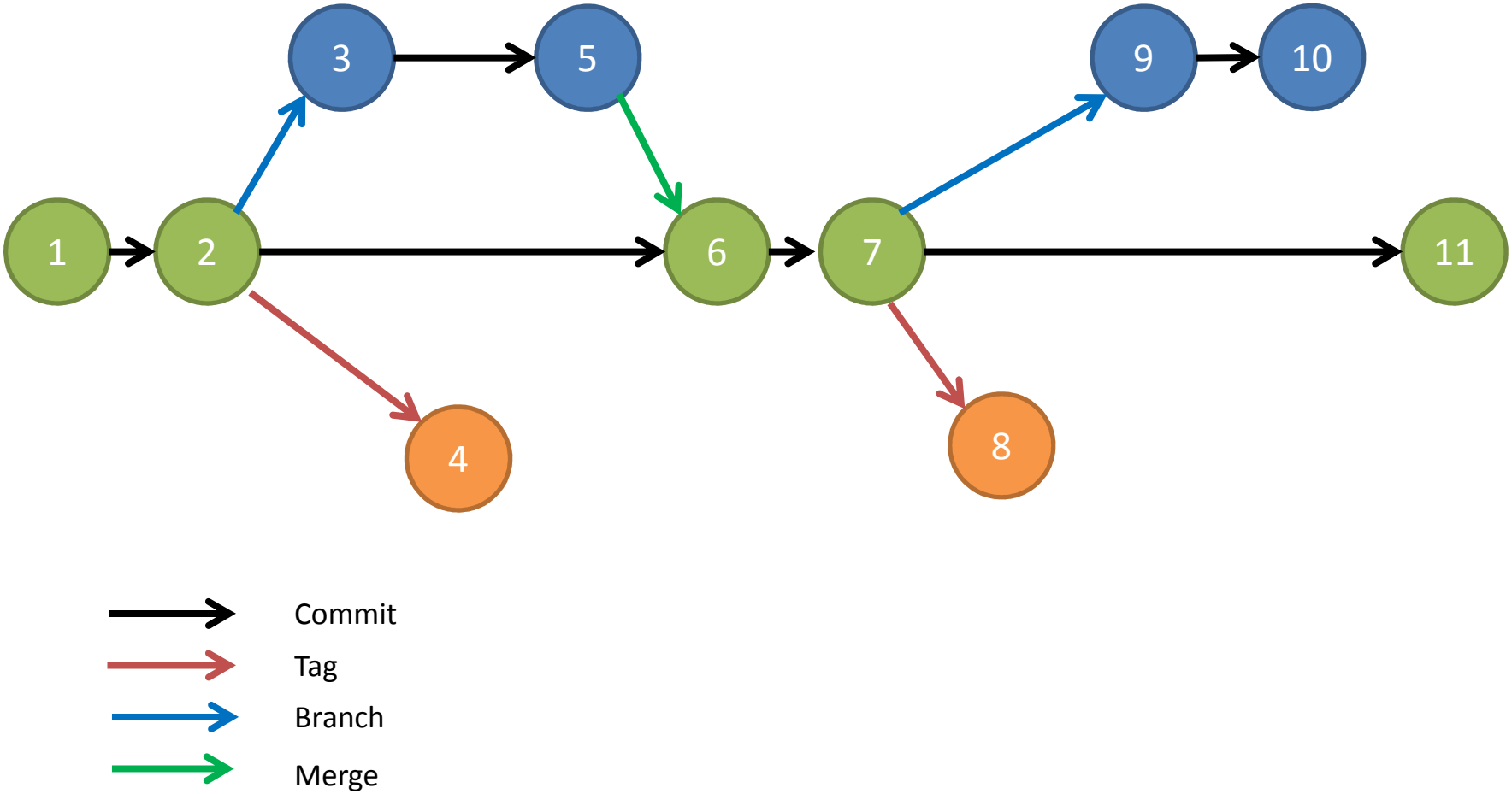
- Before you start work, you ***update*** your working copy
  1. Perform an **`svn update`** operation

- You work on your working copy. When happy, you ***commit*** your changes
  1. Perform an **`svn commit`** operation on a file or folder

CM – Subversion basics

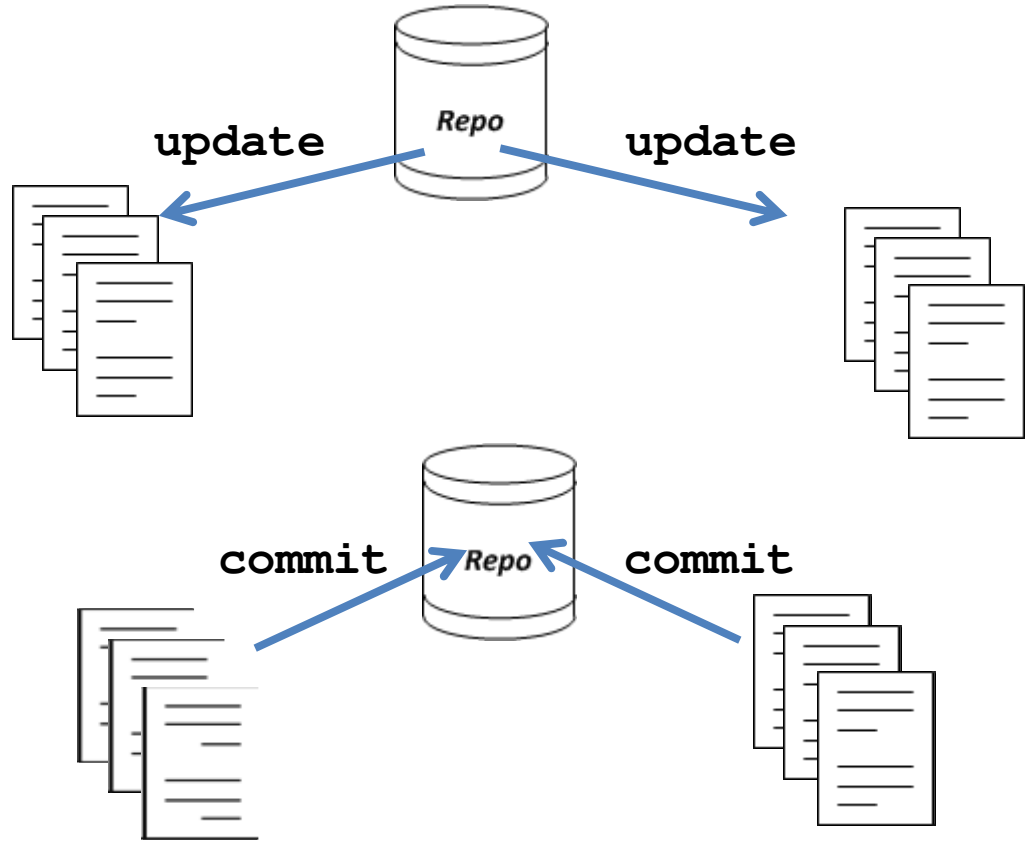
- If you make any new files, you can ***add*** them to the repository
  - Perform an **`SVN add`** operation
  - Note: Nothing changes until you *commit* your working copy
- If you regret your current changes you can ***revert*** them
  - Perform an **`SVN revert`** operation
- Other stuff:
  - Version history
  - Diff
  - Branching
  - Tagging
  - Merging

CM – Subversion



CM – Subversion: Multiple users

- Multiple users checkout/update ➔ no problem!
- Multiple users commit ➔ *usually* no problem!



Git - source code management system

- git clone (**checkout** a repository)
- git add (**add** new file to repository)
- git commit (**commit** changes to repository)
- git push (send changes to remote repository)
- git pull (**update** local repository to newest commit)

Github

- Web-based Git repository hosting
- Web-based graphical interface (PC and mobile)
- Incorporates bug tracking, feature request and task management
  https://en.wikipedia.org/wiki/GitHub
- Server: https://github.com
- Desktop Client: https://desktop.github.com/
- Repository:
  - https://github.com/kimbjerge/education-ise