

*L3 Mention Informatique  
Parcours Informatique et MIAGE*

# Génie Logiciel Avancé - Advanced Software Engineering

## Part IV : Version and Configuration Management

Burkhart Wolff

([burkhart.wolff@universite-paris-saclay.fr](mailto:burkhart.wolff@universite-paris-saclay.fr))

<https://usr.lmf.cnrs.fr/~wolff>

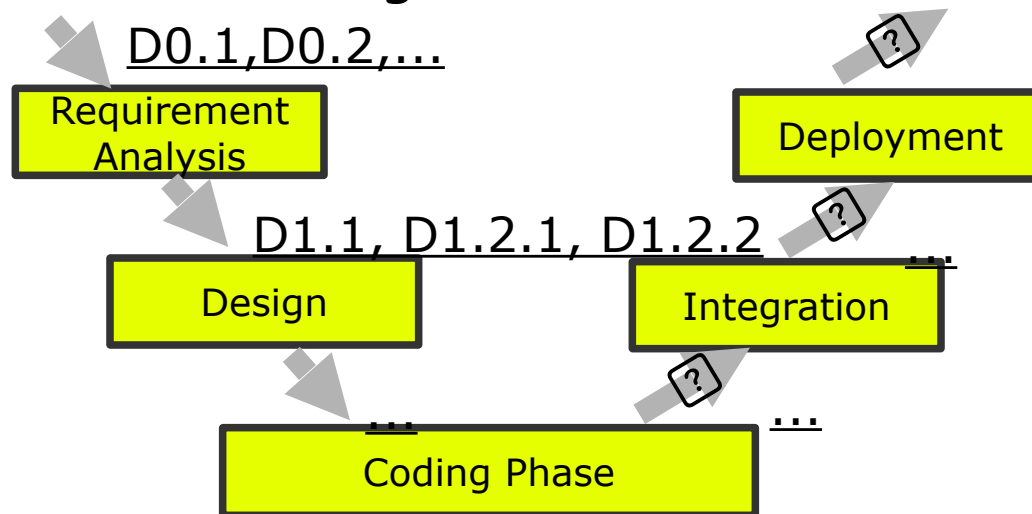
# Plan of the Chapter

---

- ❑ Motivation: Version and Configuration Management as „the“ means for collaborative development
- ❑ Version management
  - Centralized Version Control
  - Distributed Version Control
  - Organizing Merges
- ❑ Beyond Versions: Configurations
  - Build Management
  - Advanced Configuration Management

# Motivation

- ❑ Recall: SE Processes are based
  - on a large flow of documents and code
  - ... that have to be edited collaboratively
  - ... distributed consistently
  - ... while controlling access



# Motivation

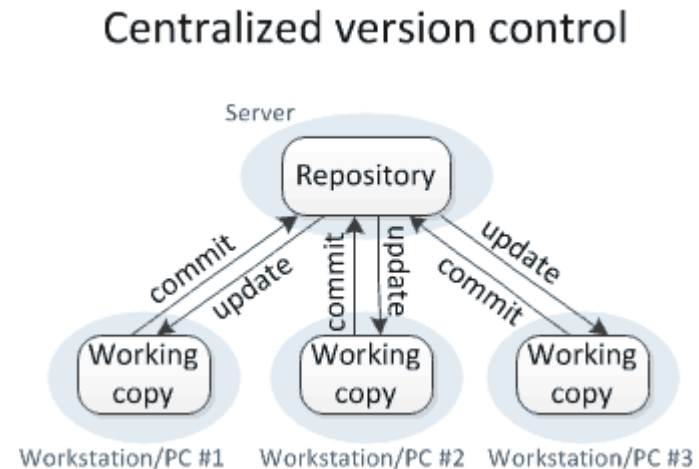
---

- ❑ Important **TECHNICAL** means to support the process
  - Explicit tools for Version Management  
(CVS, svn, git, mercurial, sourcedepot, ...)
  - Create and track **revisions** of files and file-trees
  - Create and track **differences** between files and file-trees
  - Access control to the various parties of process
  - Locking of documents
  - Merging of revisions
  - Quality control
  - ... actually, it is dead-useful for **everything** !!!

# Concepts of Centralized Version Control

---

- ❑ Working copies  
(in user space)
- ❑ Repository  
(on the server-side)
- ❑ update:  
syncing with the  
repository
- ❑ commit: creating a new revision of a document  
(involves new registration, inclusion in documents,  
consistency checks)
- ❑ operations lock, checkout, import, ...



# Concepts of Centralized Version Control

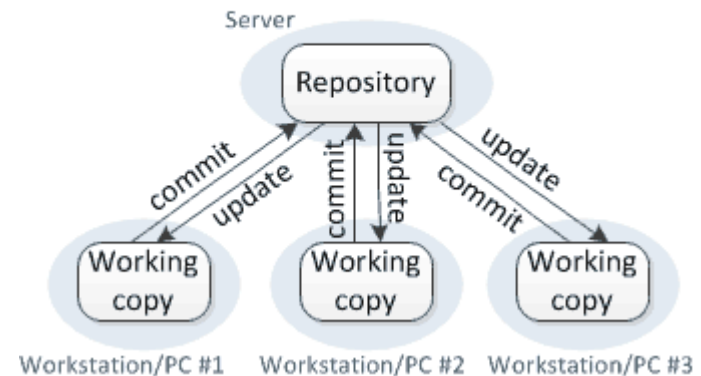
- ❑ First widely used system: *CVS*
- ❑ Nowadays in use : *svn*
- ❑ In connection with a gui-client:  
useable for end-users ...  
... for **everything** ...

❑

<https://subversion.apache.org/>



Centralized version control



# ... for everything ...

- my working copy for this course material

...

- Version - management is not just for code

...

The screenshot displays the TortoiseSVN application interface. The main window shows a file explorer view of a working copy for the 'ens' repository. The left sidebar lists various project folders, including 'codebox-publications', 'bu\_web-WWW', 'fortesse-ens', and 'RomainAissat'. The central pane shows the contents of the 'ens' folder, which includes subfolders like 'L3-GLA', 'partiel', 'labs', 'exam', 'cours', and 'adm'. The right pane shows the 'History' view for the selected file, displaying a table of revisions.

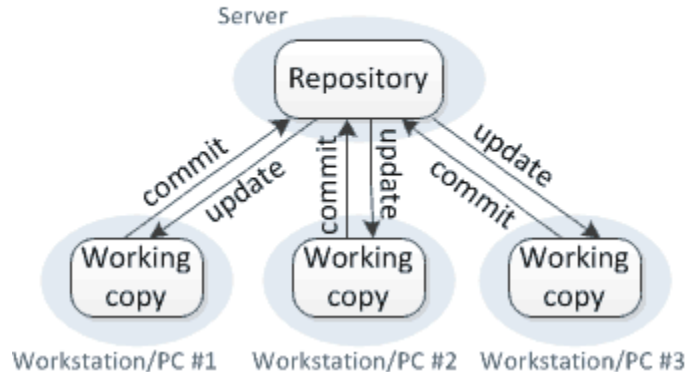
| Rev  | Date                | Author    | Log Message |
|------|---------------------|-----------|-------------|
| 2287 | 2014/09/15 16:42:16 | longuet@  |             |
| 2286 | 2014/09/15 16:41:24 | longuet@  |             |
| 2284 | 2014/09/15 15:33:45 | longuet@  |             |
| 2283 | 2014/09/15 15:29:34 | longuet@  |             |
| 2282 | 2014/09/12 14:07:26 | longuet@  |             |
| 2281 | 2014/09/12 13:40:52 | wolff@iri |             |
| 2280 | 2014/09/10 13:31:42 | wolff@iri |             |
| 2279 | 2014/09/10 13:30:53 | wolff@iri |             |
| 2273 | 2014/09/08 19:12:14 | longuet@  |             |
| 2272 | 2014/09/08 14:07:32 | longuet@  |             |

Below the history table, there is a section titled 'Changed Paths' which lists the files that have been modified in the selected revision range. The list includes paths like '/15/L3-GLA/td/1/ComptesBancaires.odg' and '/15/L3-GLA/td/1/TD1-sol.pdf'.

## One step further: Distributed Version Control

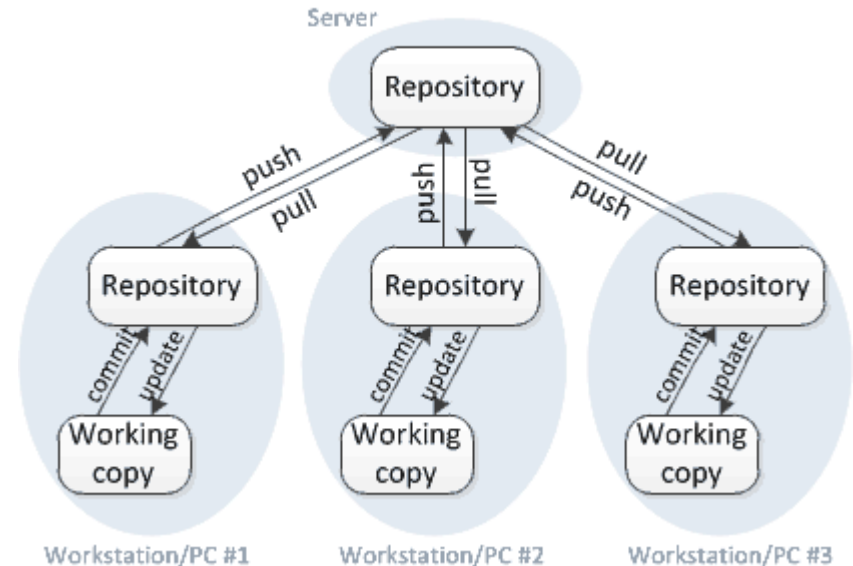
### ■ Hierarchy of repositories:

#### Centralized version control



- one more sync-level, but more precise history in practice... since everybody can check in locally
- hierarchy strictly speaking not necessary

#### Distributed version control



# Distributed Version Control Systems

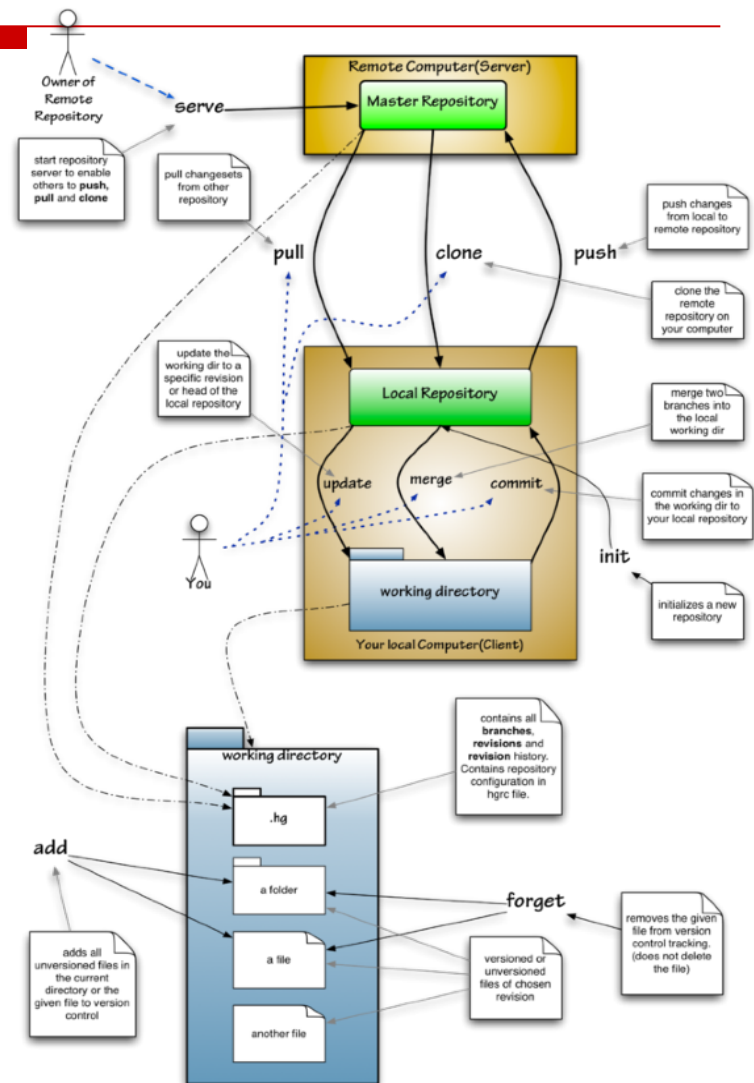
## ❑ Opensource:

➤ git (Linux)



➤ mercurial (Isabelle)

<http://mercurial.selenic.com/>



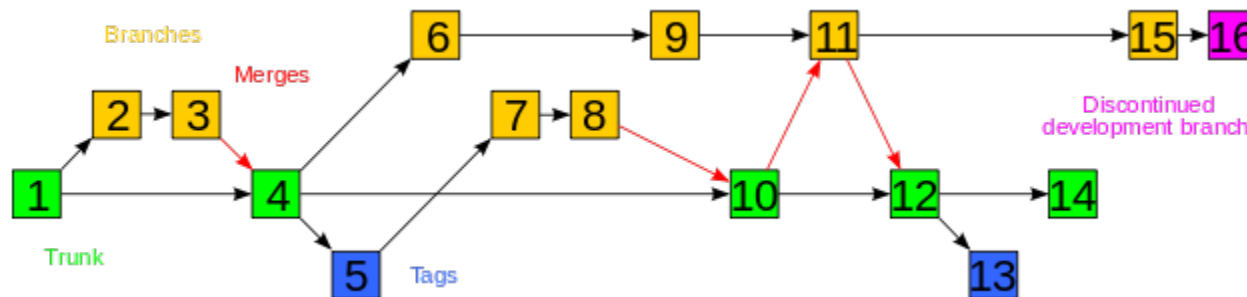
\_\_\_\_\_

- © 2011 Blackwell Publishing Ltd *Journal of Internal Medicine* 270: 103–110 109

Copyright © 2008 Elsevier B.V. All rights reserved.

# Problems of a distributed development

- ❑ If not synced via explicit locks, a development looks like this can be organised via “merges”

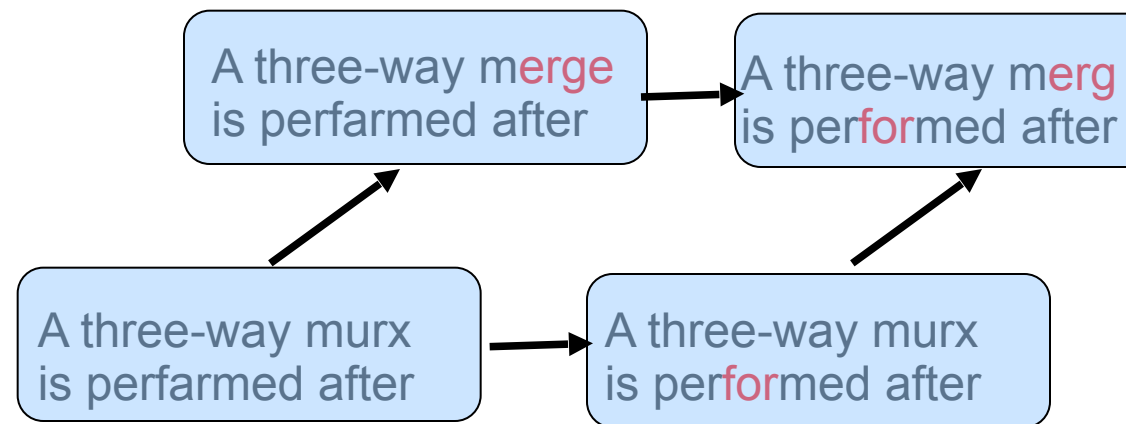
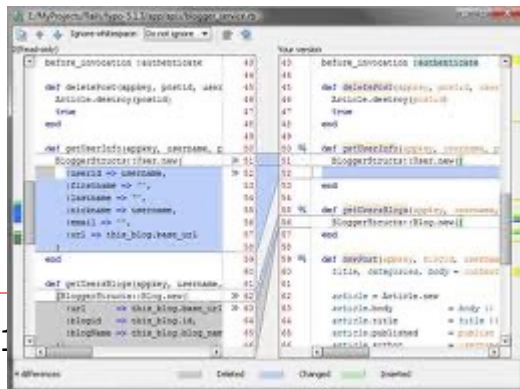
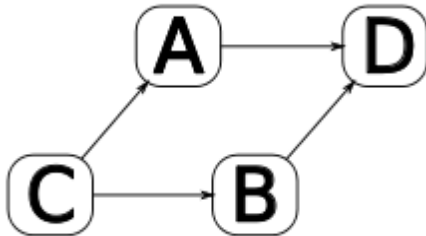


- ❑ Preference for re-generation binary formats
- ❑ Merging: For binary formats, only special purpose merges work (as in MS Word, for example ...)
- ❑ For textual formats :

# A partial solution ...

## ■ For textual formats:

A three-way merge is performed after an automated difference analysis between a file 'A' and a file 'B' while also considering the origin, or common ancestor, of both files. It is a rough merging method, but widely applicable since it only requires one common ancestor to reconstruct the changes that are to be merged.

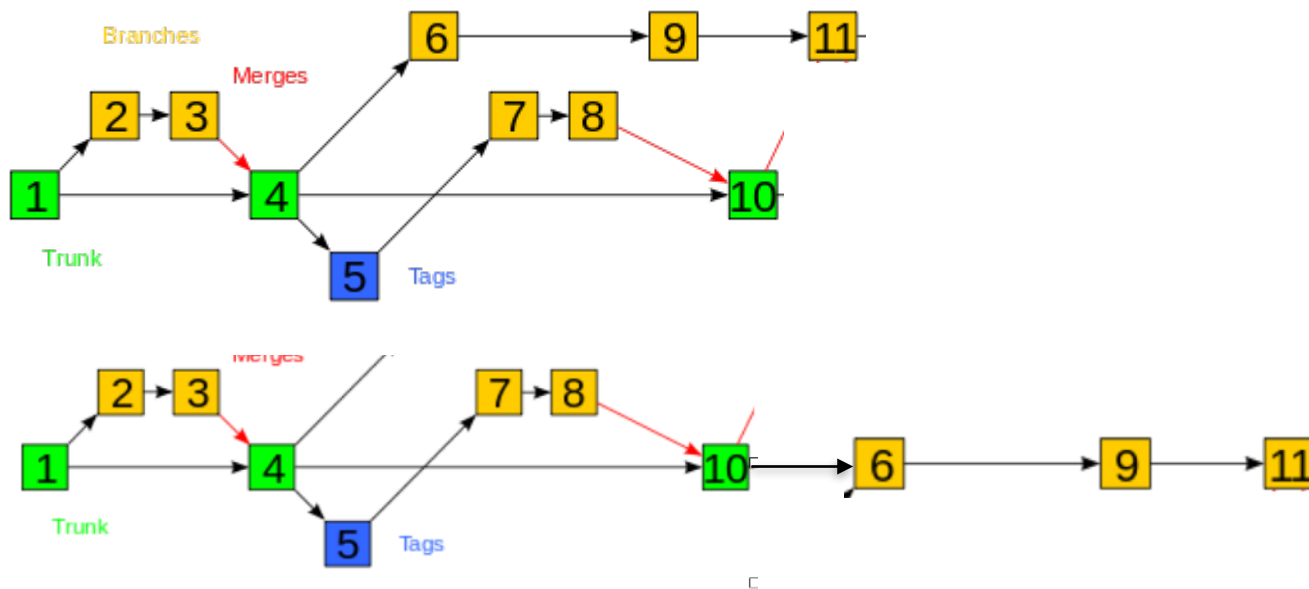


Tools: For example xmerge, which also offer conflict resolution by hand ...

# Problems of a distributed development

---

- ❑ Another strategy of organising developments are "rebases":



- ❑ Preference for re-generation binary formats

# A better solution ...

---

- ❑ In a distributed development process, where a very large number of merges occurs routinely, the validation of the intermediate results becomes **crucial**.

(This limits the use of informal documents.)

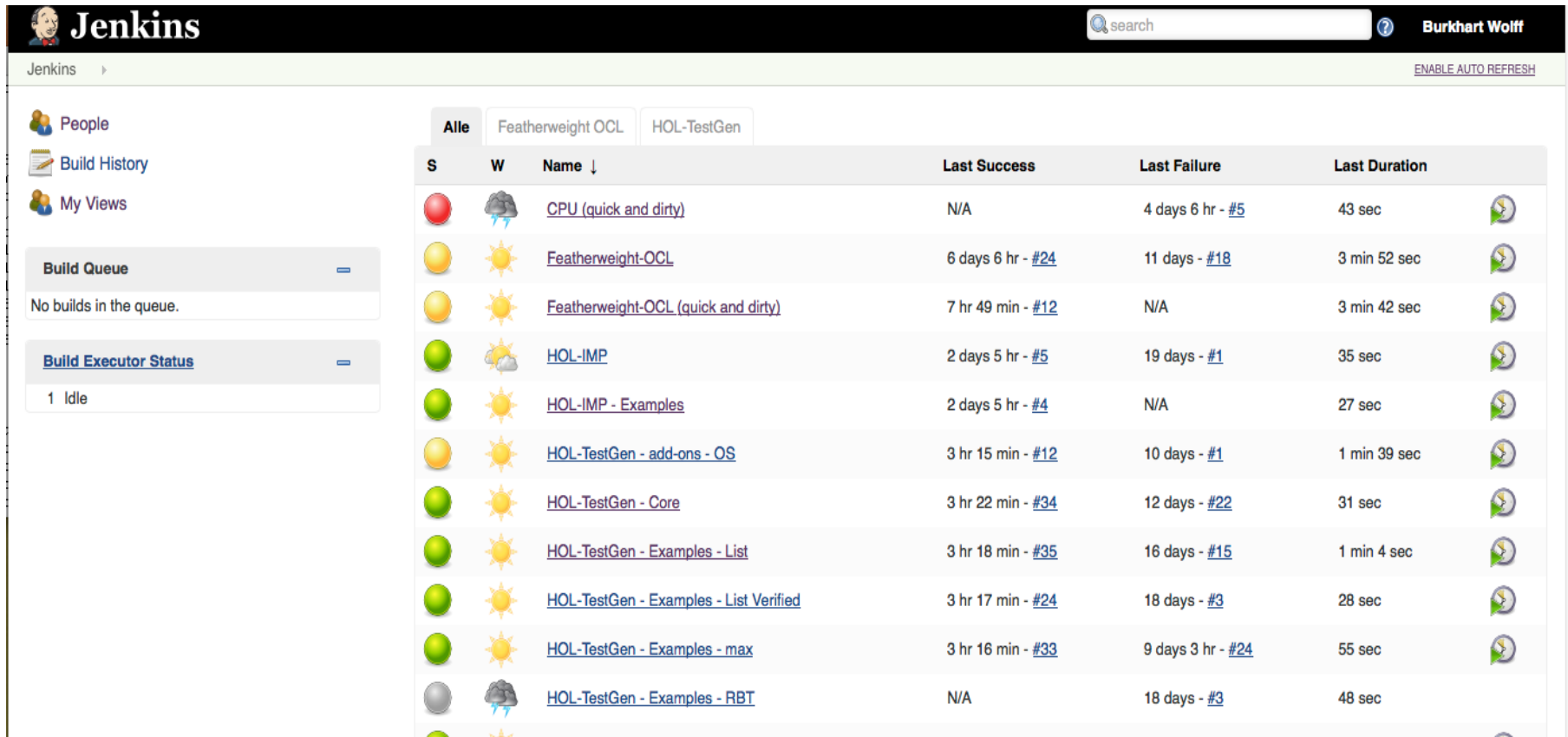
- validation on any check-in  
(for example: automated type-checking)
- validation for UML - documents  
(self-defined consistency checkers for UML models)
- **automated static analysis and tests during systematic and periodic builds ...**
- ... more advanced methods, using proof techniques.

# Build Management: A Build-Server

---

- Pervasive Version-Management allows for a “continuous build / continuous integration” element in the dev-process
  - Update all sources from the repository
  - Re-compile everything
  - Rerun all static checks (lint, code analysis...)
  - Rerun all executable test-suites
- Open Source Solution : Jenkins

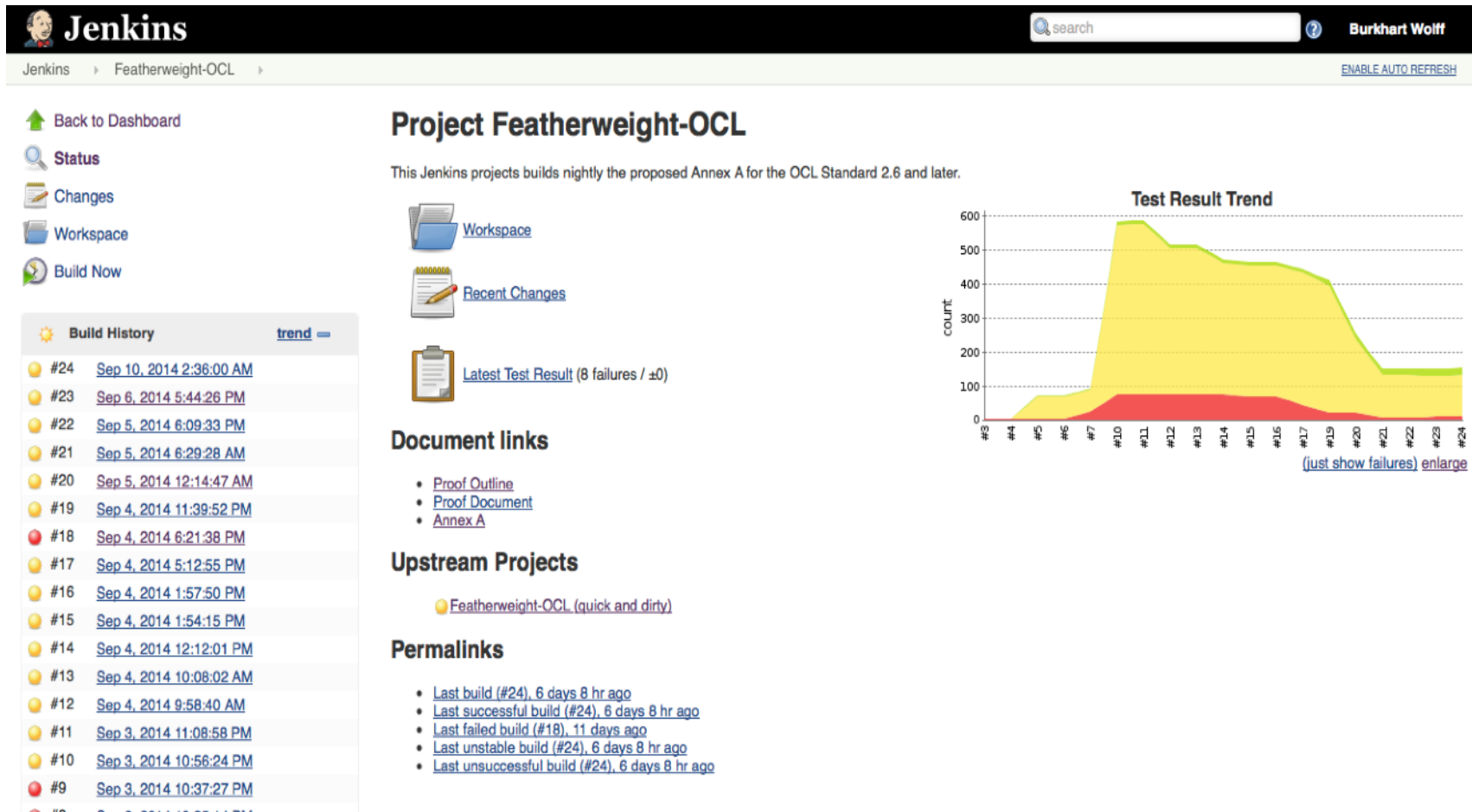
# Build Management: A Build-Server



The screenshot displays the Jenkins web interface. At the top, the Jenkins logo and name are on the left, a search bar and the user name 'Burkhart Wolff' are on the right. Below the header, the left sidebar contains navigation links: 'People', 'Build History', 'My Views', 'Build Queue' (showing 'No builds in the queue.'), and 'Build Executor Status' (showing '1 Idle'). The main content area shows a table of build jobs, with tabs for 'Alle', 'Featherweight OCL', and 'HOL-TestGen'. The table has columns for status (S), icon (W), name, last success, last failure, and last duration. Each row represents a build job with its corresponding status icon and a link to view the build details.

| S | W | Name ↓                                                 | Last Success                      | Last Failure                      | Last Duration |
|---|---|--------------------------------------------------------|-----------------------------------|-----------------------------------|---------------|
|   |   | <a href="#">CPU (quick and dirty)</a>                  | N/A                               | 4 days 6 hr - <a href="#">#5</a>  | 43 sec        |
|   |   | <a href="#">Featherweight-OCL</a>                      | 6 days 6 hr - <a href="#">#24</a> | 11 days - <a href="#">#18</a>     | 3 min 52 sec  |
|   |   | <a href="#">Featherweight-OCL (quick and dirty)</a>    | 7 hr 49 min - <a href="#">#12</a> | N/A                               | 3 min 42 sec  |
|   |   | <a href="#">HOL-IMP</a>                                | 2 days 5 hr - <a href="#">#5</a>  | 19 days - <a href="#">#1</a>      | 35 sec        |
|   |   | <a href="#">HOL-IMP - Examples</a>                     | 2 days 5 hr - <a href="#">#4</a>  | N/A                               | 27 sec        |
|   |   | <a href="#">HOL-TestGen - add-ons - OS</a>             | 3 hr 15 min - <a href="#">#12</a> | 10 days - <a href="#">#1</a>      | 1 min 39 sec  |
|   |   | <a href="#">HOL-TestGen - Core</a>                     | 3 hr 22 min - <a href="#">#34</a> | 12 days - <a href="#">#22</a>     | 31 sec        |
|   |   | <a href="#">HOL-TestGen - Examples - List</a>          | 3 hr 18 min - <a href="#">#35</a> | 16 days - <a href="#">#15</a>     | 1 min 4 sec   |
|   |   | <a href="#">HOL-TestGen - Examples - List Verified</a> | 3 hr 17 min - <a href="#">#24</a> | 18 days - <a href="#">#3</a>      | 28 sec        |
|   |   | <a href="#">HOL-TestGen - Examples - max</a>           | 3 hr 16 min - <a href="#">#33</a> | 9 days 3 hr - <a href="#">#24</a> | 55 sec        |
|   |   | <a href="#">HOL-TestGen - Examples - RBT</a>           | N/A                               | 18 days - <a href="#">#3</a>      | 48 sec        |

# Build Management: A Build-Server



# Towards Configuration Management

## ❑ ... generalizes Version-Management

- by configuration descriptions  
(including functionality environment, hardware,...)
- attempts to GENERATE a revision on the basis of meta-data over dependencies and change-sets
- works with heavy virtualization techniques nowadays (Google, Microsoft)

