

*L3 Mention Informatique  
Parcours Informatique et MIAGE*

# Génie Logiciel Avancé - Advanced Software Engineering

## Part IV : Version and Configuration Management

Burkhart Wolff  
wolff@lri.fr

# Plan of the Chapter

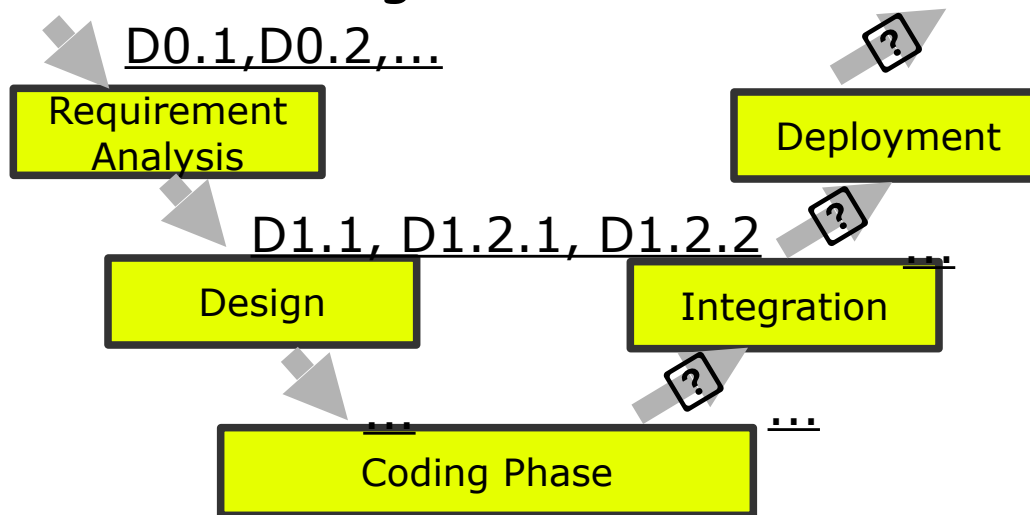
---

- ❑ 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

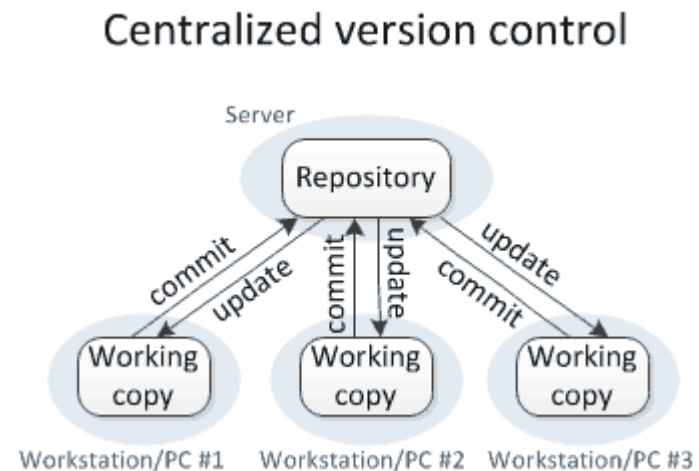
---

- ❑ Important **TECHNICAL** means to support the process
  - Explicit tools for Version Management  
(CVS, svn, git, mercurial, source depot, ...)
  - 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

---

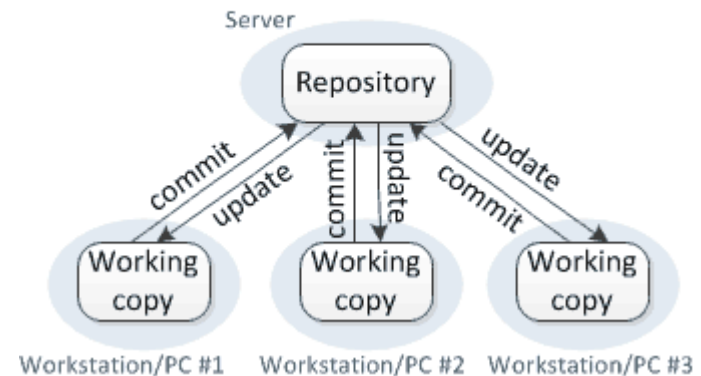
- ❑ Working copies (in user space)
- ❑ Repository (on the server-side)
- ❑ update: syncing with the repository
- ❑ commit: creating a new revision of a document (involves new registration, inclusion in documents, consistency checks)
- ❑ operations lock, checkout, import, ...



# 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** ...
- ❑

Centralized version control

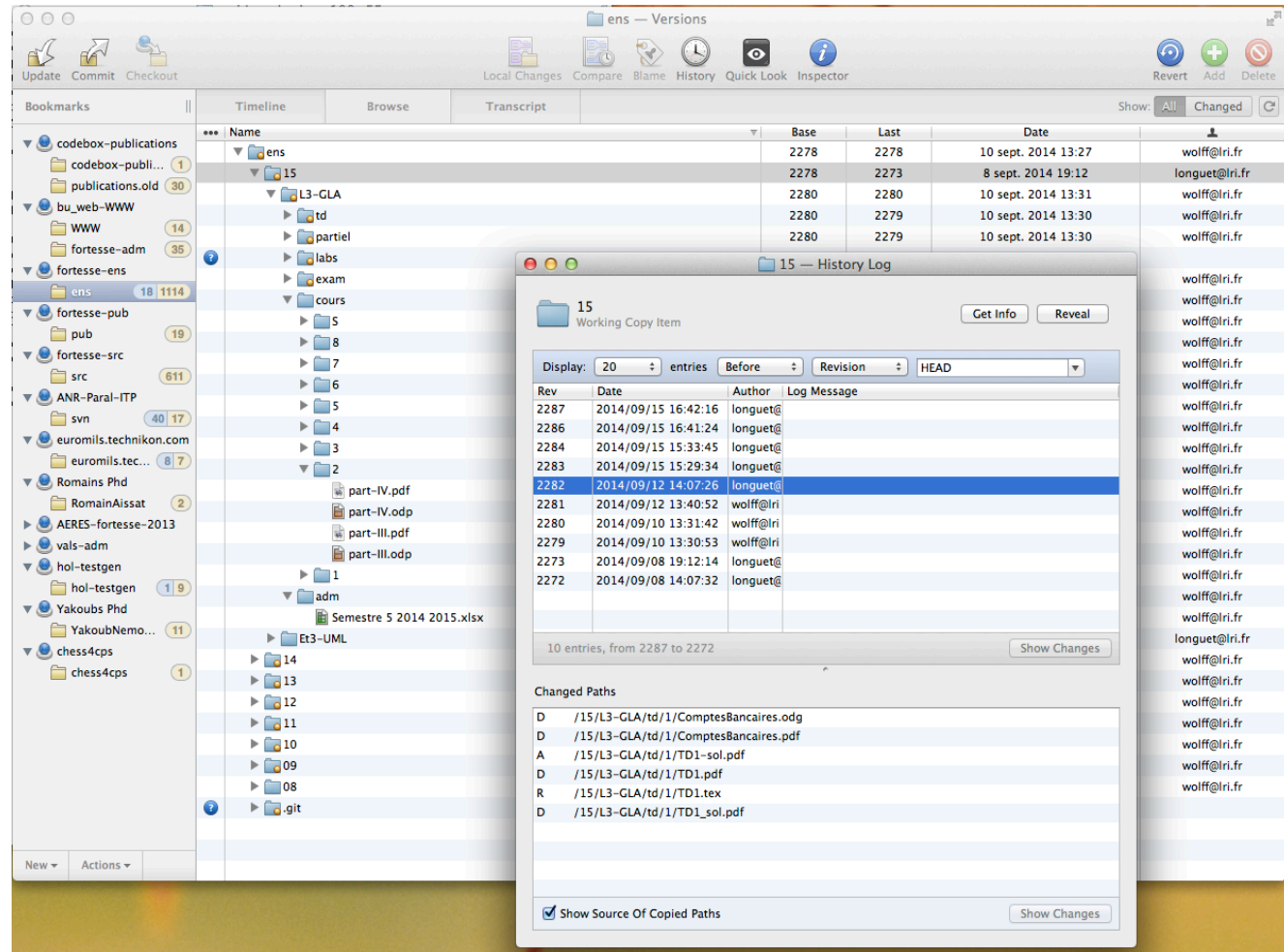


<https://subversion.apache.org/>



# ... for everything ...

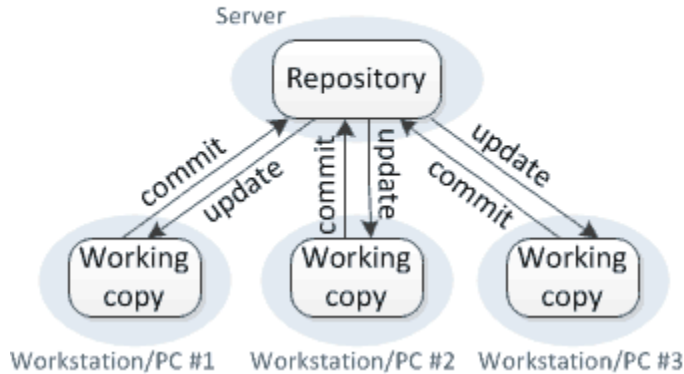
- my working copy for this course material
- ...
- Version - management is not just for code
- ...



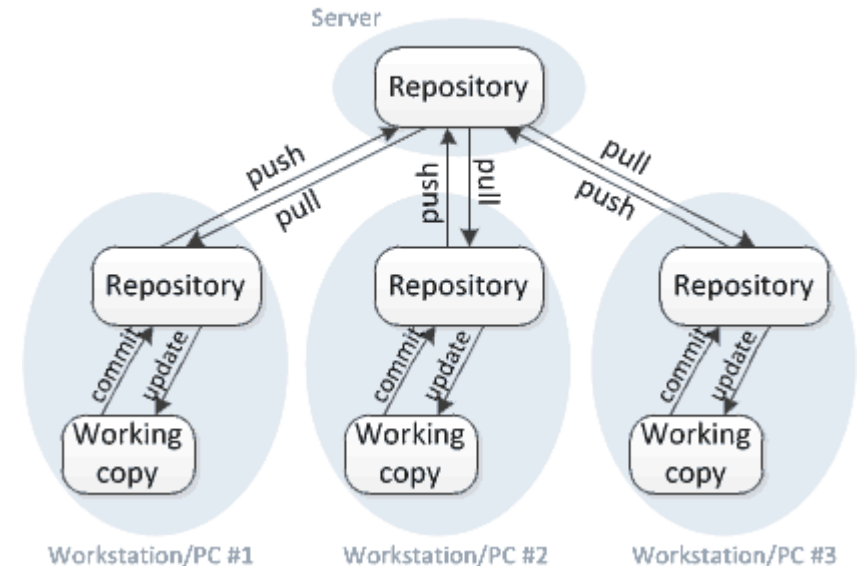
One step further:  
Distributed Version Control

□ Hierarchy of repositories:

Centralized version control



Distributed 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 Systems

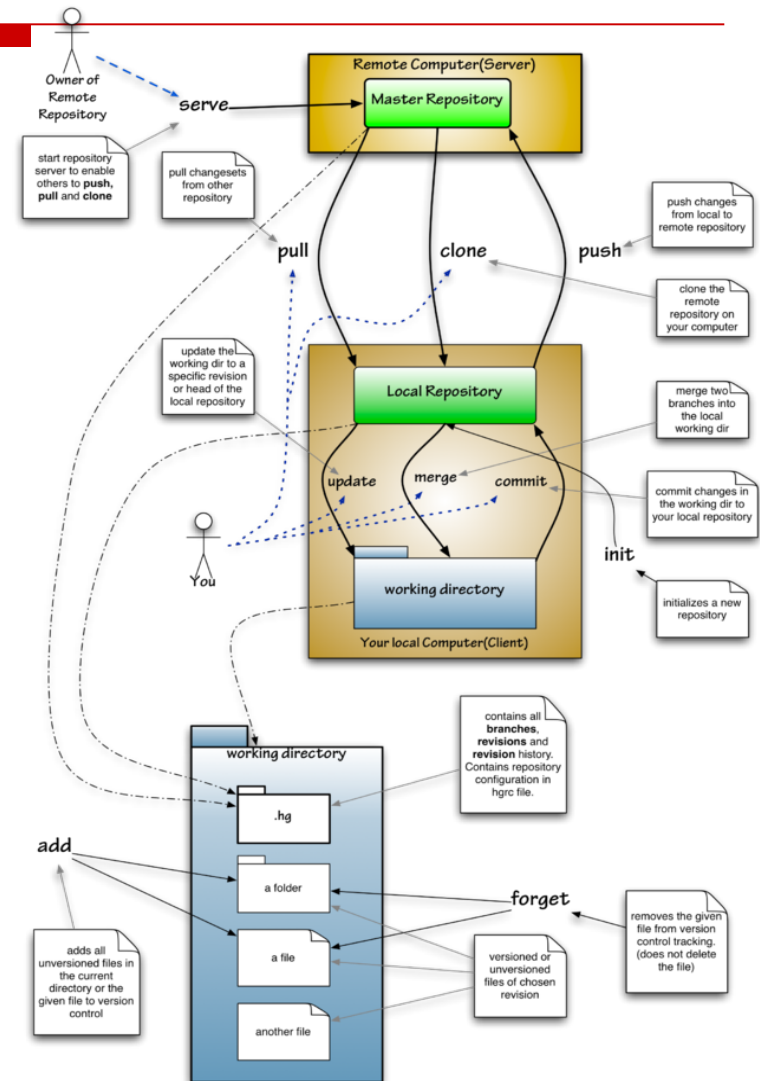
## Open source:

➤ git (Linux)



➤ mercurial (Isabelle)

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



# Consequences for a development process

- ❑ The entire development becomes visible, trackable, reproducible, and can be an object in its own right (if formal): apply changeset XYZ ...

The screenshot shows a table of changesets with the following columns: 'When', 'Who', and 'Why'. Red circles highlight specific entries in each column. The 'When' column highlights a date and time. The 'Who' column highlights a user name. The 'Why' column highlights a description of the change. Below the table, there is a 'Details' section with a 'What' label and a red circle highlighting a specific detail.

When	Who	Why
2007-08-08 10:00:00	John Doe	Applied when new database schema...
2007-08-08 10:00:00	John Doe	Applied when new database schema...
2007-08-08 10:00:00	John Doe	Applied when new database schema...
2007-08-08 10:00:00	John Doe	Applied when new database schema...
2007-08-08 10:00:00	John Doe	Applied when new database schema...
2007-08-08 10:00:00	John Doe	Applied when new database schema...
2007-08-08 10:00:00	John Doe	Applied when new database schema...
2007-08-08 10:00:00	John Doe	Applied when new database schema...
2007-08-08 10:00:00	John Doe	Applied when new database schema...
2007-08-08 10:00:00	John Doe	Applied when new database schema...

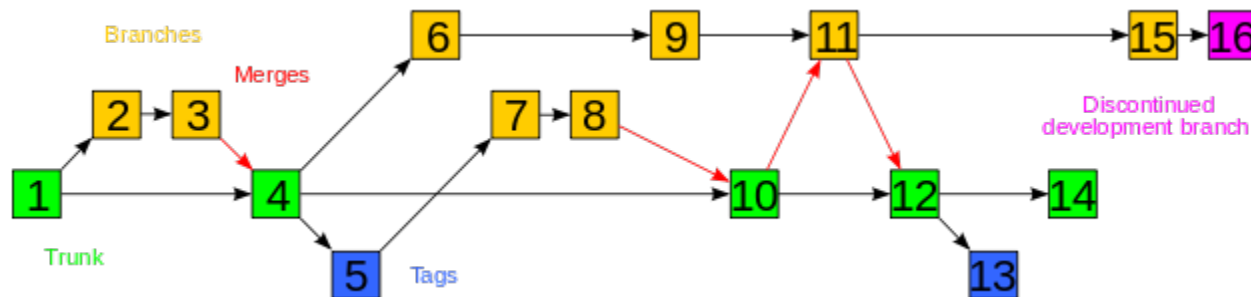
Details

What

Copyright © 2007 Redhat Developer, Inc. All rights reserved.

# Problems of a distributed development

- If not synced via explicit locks, a development looks like this:

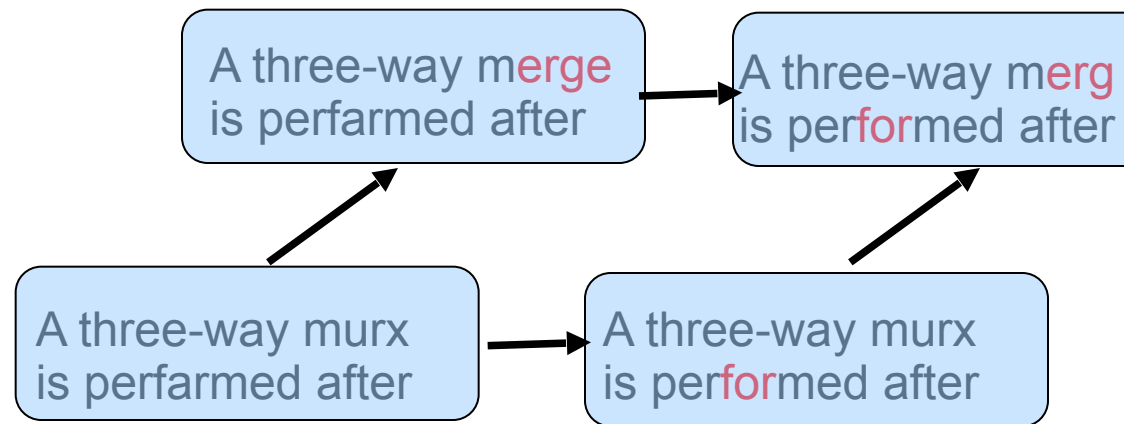
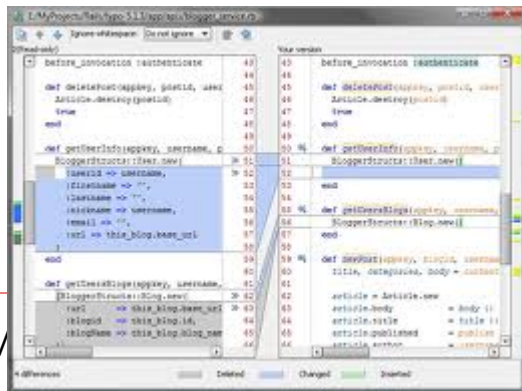
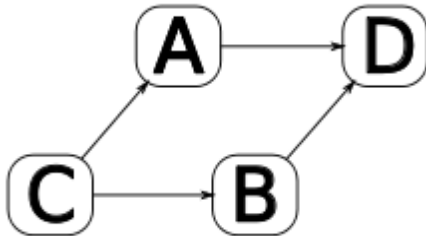


- Merging: For binary formats, only special purpose merges work (as in MS Word, for example ...)
- For textual formats :

# 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 ...

# A better solution ...

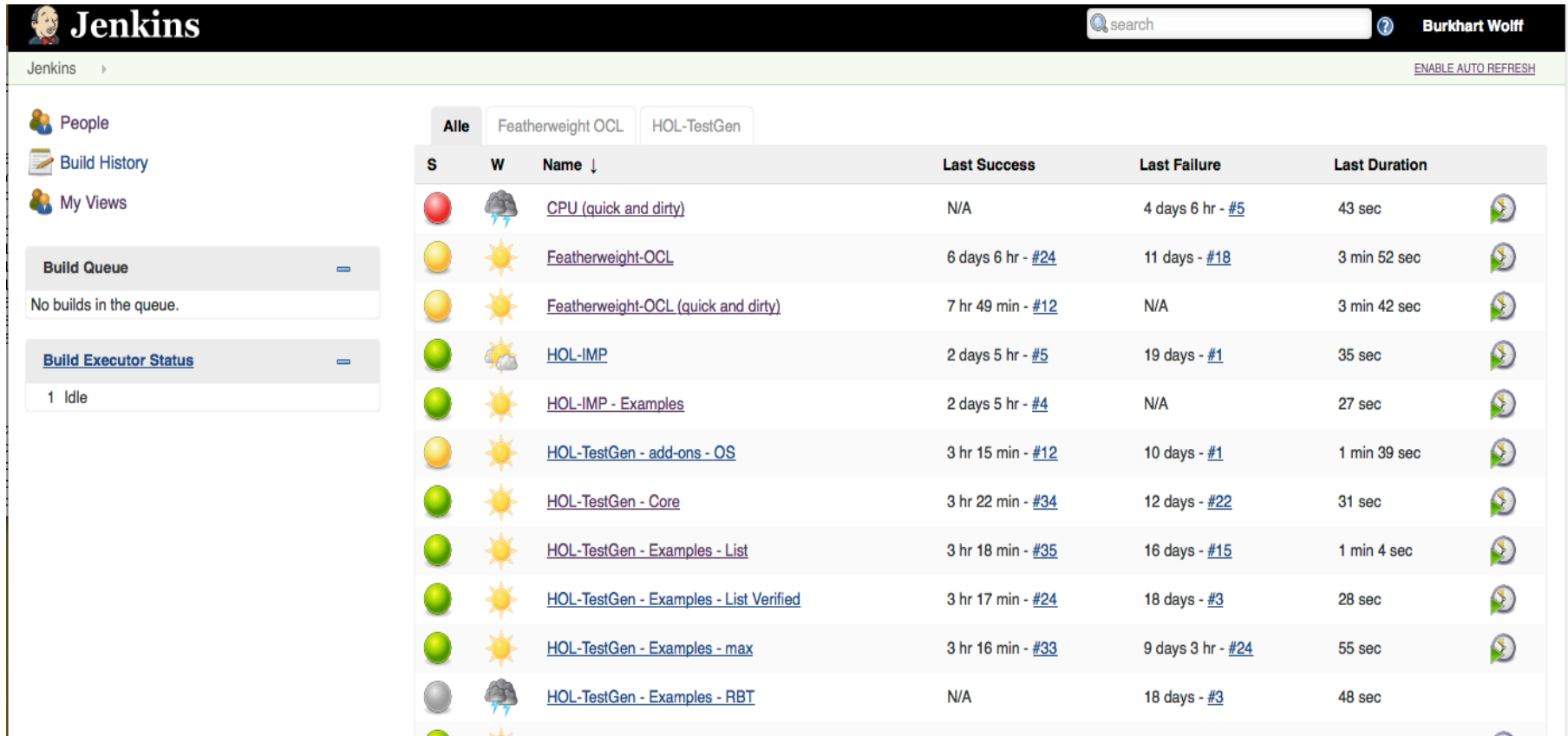
---

- ❑ 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



The screenshot displays the Jenkins web interface. At the top left is the Jenkins logo and the name 'Jenkins'. On the right, there is a search bar and the user name 'Burkhard Wolff'. Below the header, there are navigation links for 'People', 'Build History', and 'My Views'. On the left side, there are two panels: 'Build Queue' showing 'No builds in the queue.' and 'Build Executor Status' showing '1 Idle'. The main area shows a table of build jobs with filters for 'Alle', 'Featherweight OCL', and 'HOL-TestGen'. The table has columns for status (S), weather icon (W), name, last success, last failure, and last duration. Each row also includes a small circular icon on the right.

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

search?Burkhard WolffENABLE AUTO REFRESH

Jenkins > Featherweight-OCL

Back to Dashboard

Status

Changes

Workspace

Build Now

### Build History

trend

#24	<a href="#">Sep 10, 2014 2:36:00 AM</a>
#23	<a href="#">Sep 6, 2014 5:44:26 PM</a>
#22	<a href="#">Sep 5, 2014 6:09:33 PM</a>
#21	<a href="#">Sep 5, 2014 6:29:28 AM</a>
#20	<a href="#">Sep 5, 2014 12:14:47 AM</a>
#19	<a href="#">Sep 4, 2014 11:39:52 PM</a>
#18	<a href="#">Sep 4, 2014 6:21:38 PM</a>
#17	<a href="#">Sep 4, 2014 5:12:55 PM</a>
#16	<a href="#">Sep 4, 2014 1:57:50 PM</a>
#15	<a href="#">Sep 4, 2014 1:54:15 PM</a>
#14	<a href="#">Sep 4, 2014 12:12:01 PM</a>
#13	<a href="#">Sep 4, 2014 10:08:02 AM</a>
#12	<a href="#">Sep 4, 2014 9:58:40 AM</a>
#11	<a href="#">Sep 3, 2014 11:08:58 PM</a>
#10	<a href="#">Sep 3, 2014 10:56:24 PM</a>
#9	<a href="#">Sep 3, 2014 10:37:27 PM</a>

## Project Featherweight-OCL

This Jenkins projects builds nightly the proposed Annex A for the OCL Standard 2.6 and later.

- Workspace
- Recent Changes
- Latest Test Result (8 failures / ±0)

### Document links

- Proof Outline
- Proof Document
- Annex A

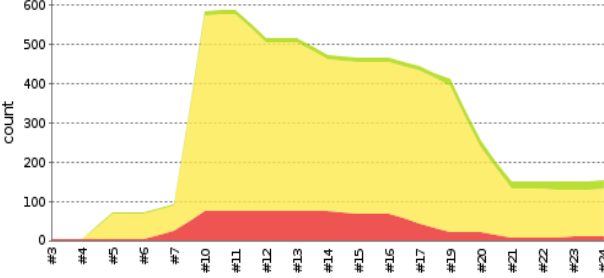
### Upstream Projects

- Featherweight-OCL (quick and dirty)

### Permalinks

- Last build (#24), 6 days 8 hr ago
- Last successful build (#24), 6 days 8 hr ago
- Last failed build (#18), 11 days ago
- Last unstable build (#24), 6 days 8 hr ago
- Last unsuccessful build (#24), 6 days 8 hr ago

### Test Result Trend



(just show failures) enlarge

# 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)

