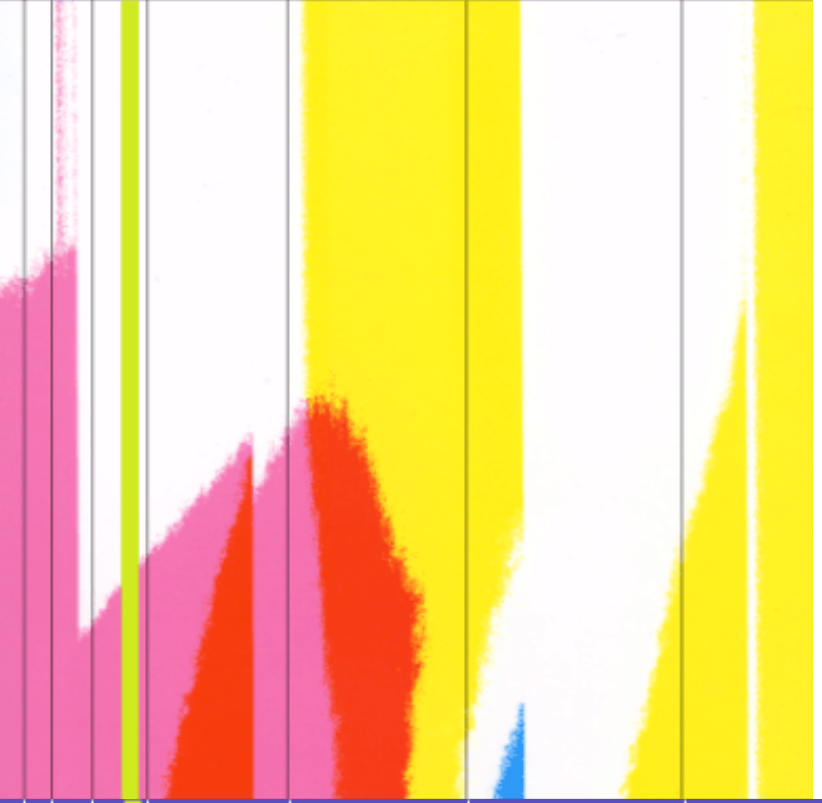


Version Control and Management

Nicolas Chleq - DREAM

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA
SOPHIA ANTIPOLIS

Version Control

Outline

- Concepts
  - Repository
  - Client
  - Simple and everyday use
- Tools
  - CVS
  - Subversion
  - GUI Clients
- Advanced operations

Version Control

Tools and methods for collaborative work

- Maintain history of all past versions and their contents
- Manage concurrent access

Designed to ease:

- Communication among developers
- Organization

Not a replacement for communication and organization

For all files that are not automatically generated

- Sources files, build, tests, deployment
- Documentation, reports

Model of versioning

Centralised versioning

- Clients use a centralized repository

Repository :

- Database of history of all files and directories
- Efficient : keep only the changes and the full content of one version
- Repository contains the **reference** version of the software:
  - Clean code that compiles
  - Passes all tests

Simple versioning operations

Checkout

- Extracts from the repository one version of all files
- Creates on the local disk a “working copy”
- Extracted version defaults to the most recent one

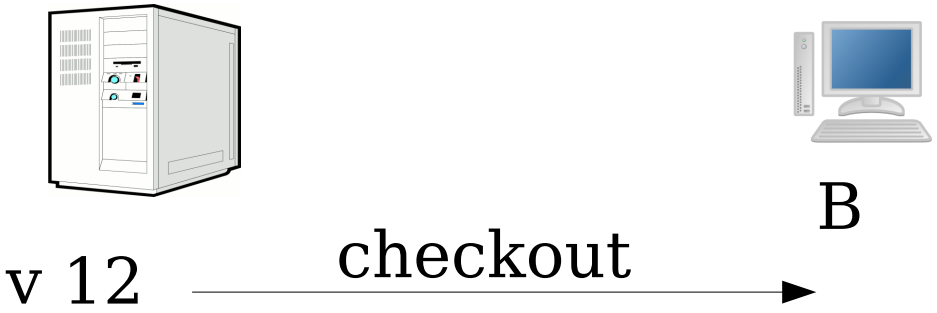
Checkin (commit)

- Creates new versions **in the repository** based on the contents of the “working copy”

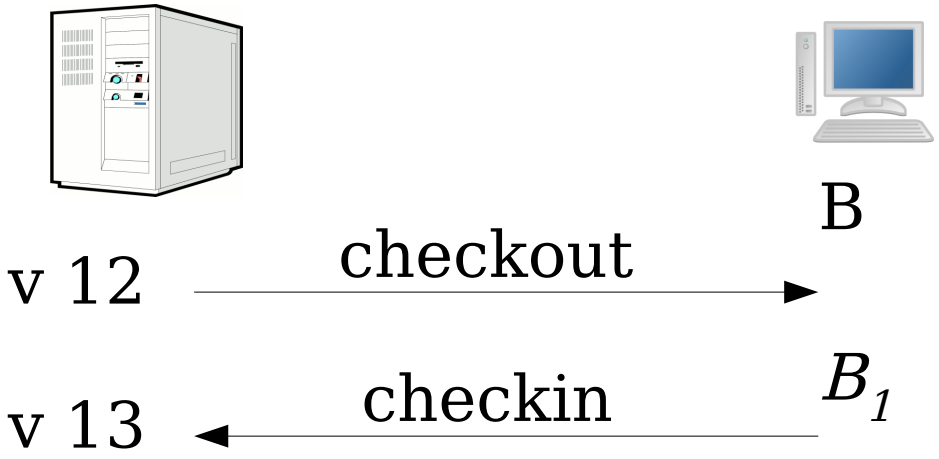
Update

- Synchronize the “working copy” with the repository contents
- Merge in the “working copy” the local changes with the changes recorded in the repository since the last checkout or update

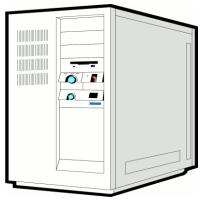
Scenarios



Scenarios



Scenarios



B

A

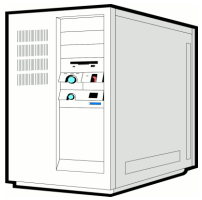
v 12 — checkout →

← checkin — B_1 — v 13

checkout →

A_1

Scenarios



B

A

v 12 — checkout →

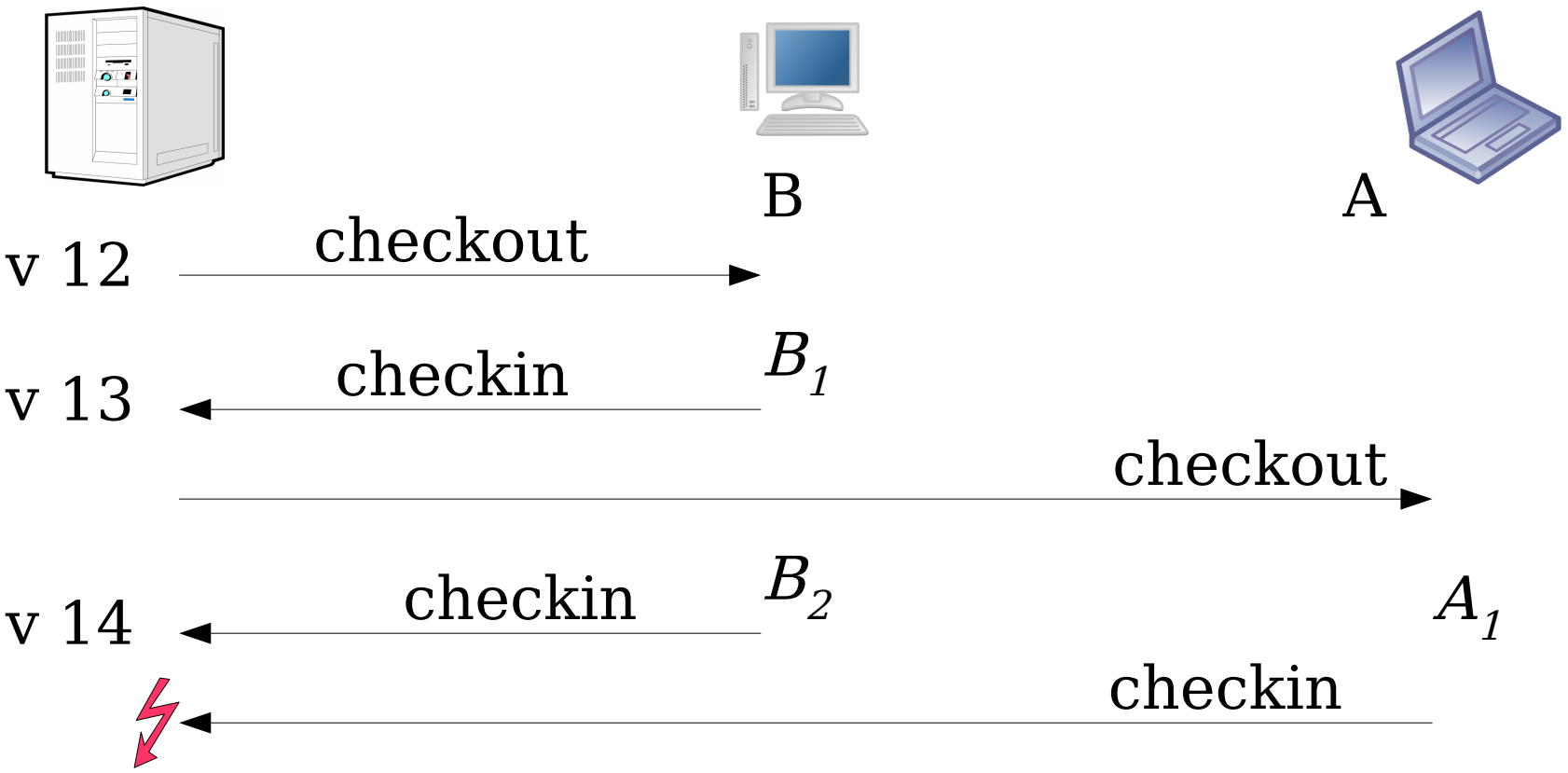
v 13 — ← checkin B_1

checkout →

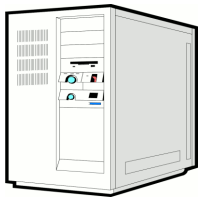
v 14 — ← checkin B_2

A_1

Scenarios


B
A
v 12
checkout
v 13
checkin
B_1
checkout
v 14
checkin
B_2
A_1
checkin

Scenarios



B

A

v 12 — checkout →

v 13 — ← checkin B_1

checkout →

v 14 — ← checkin B_2

A_1

checkin ←

update →

Merge of A_1 and B_2

v 15 — ← checkin

Tools/1

Command line CVS

- General syntax of commands
  - cvs checkout *[files/directory]*
  - cvs checkin *[...]*
  - cvs update *[...]*
- Status querying commands
  - cvs diff *[...]*
  - cvs status *[...]*
  - cvs log *[...]*

Old tool with some deficiencies

New projects shoud use its successor: Subversion

Tools/2

Command line Subversion (SVN)

- General syntax of commands
  - svn co *[files/directory]*
  - svn ci *[...]*
  - svn update *[...]*
- Status querying commands
  - svn diff *[...]*
  - svn status *[...]*
  - svn log *[...]*

One revision per checkin : applies to all files in the checkin

Can version the directories (name, contents)

Tools/3

Graphical interface clients

- Cervisia : CVS client, extension of KDE file explorer
- Plugins SVN for Eclipse
  - subclipse
  - Subversive
- TortoiseCVS et TortoiseSVN : Windows clients
- SmartCVS et SmartSVN : portable Java clients

Subversion usage

Working copy status

```
$ svn status module.c
M       module.c
```

Subversion usage

Working copy status

```
$ svn status module.c
M       module.c

$ svn diff module.c
Index: module.c
===================================================================
--- module.c    (revision 41)
+++ module.c    (working copy)
@@ -97,7 +97,9 @@

 RTIME module_load_time;

+int s1 = 0, s2 = 0, s3 = 0, s4 = 0;
+
 static void cleanup_control_module(void);

$
```

Subversion usage

Checkin

```
$ svn ci -m "Traces pour /proc en mode debug" module.c
   Sending        module.c
   Transmitting file data .
   Committed revision 47.
```

Subversion usage

Checkin

```
$ svn ci -m "Traces pour /proc en mode debug" module.c
   Sending        module.c
   Transmitting file data .
   Committed revision 47.
```

Add a file to the repository

```
$ svn status
?       enable.c
M       Makefile
$
```

Subversion usage

Checkin

```
$ svn ci -m "Traces pour /proc en mode debug" module.c
   Sending        module.c
   Transmitting file data .
   Committed revision 47.
```

Add a file to the repository

```
$ svn status
?       enable.c
M       Makefile
$ svn add enable.c
A         enable.c
$ svn ci -m "Fonctions d'activation des actionneurs" enable.c
Adding         enable.c
Transmitting file data .
Committed revision 49.
```

Advanced Operations

Import

- Imports in the repository a source tree in a single operation
  - Commands : `cvs import` and `svn import`

Branches

- Fork of a history line : creates 2 independent lines of versions
- CVS uses “tags” (character strings) to handle branches
  - Commands : `cvs tag` and `cvs rtag`
- SVN uses copies
  - Command : `svn copy`

Creation of a repository

- `cvs init` *`path`* and `svnadmin create` *`path`*

Links/1

CVS

http://cvshome.org/
http://cvsbook.red-bean.com/

SVN

http://subversion.tigris.org
http://svnbook.red-bean.com/
http://calcul.math.cnrs.fr/IMG/pdf/subversion.pdf

Général

http://www.perforce.com/perforce/bestpractices.html
http://svn.collab.net/repos/svn/trunk/doc/user/svn-best-practices.html

Clients

http://tortoisesvn.tigris.org/
Subversive
SmartCVS et SmartSVN : http://www.syntevo.com/

Links/2

Other appoaches : decentralised/distributed/P2P

- Several repositories (one per developer for example)
- Checkins can be performed into one or several repositories
- Repositories can be selectively synchronized
- Links:
  - http://www.jres.org/slides/2.pdf
  - http://www.selenic.com/mercurial/wiki/

- New approach for versioning control : the way of the future ?

Conclusions

We are **always** part of a team (in space and/or time) :

- We should **always** use versioning

Is only a tool :

- Not a replacement for communication among developers
- Not a replacement for organization of software development

Centralized versioning model:

- The easiest to begin with
- The easiest to work with
- The most frequently used nowadays