Computer Labs: Version Control with Subversion

2º MIEIC

Pedro F. Souto (pfs@fe.up.pt)

November 21, 2010

The Problem

```
$edit foo.c, make, run, edit, make, run, ...
```

OK! Now that it enters in graphic mode, let's make a backup

```
$copy foo.c foo.v1.c
$edit foo.c, make, run, edit, make, run, ...
```

OK! Now that it maps graphic memory, let's make another backup

```
$copy foo.c foo.v2.c
$edit foo.c, make, run, edit, make, run, ...
```

OK! Now that it draws a pixel, let's make another backup

```
$copy foo.c foo.v3.c
$edit foo.c, make, run, edit, make, run, ...
```

Oops! Does not leave graphics mode, let's retrieve the backup

```
$copy foo.v7.c fooc.
```

Hmm! This is not the version I want. Should it be v3? ... Oops, deleted the last version !@£#%/#*&$@

The Solution? Subversion! (SVN)

- Subversion is a version control system that is able to:
  - Keep several versions of an entire (development) directory tree
  - Restore any of the versions it keeps in a consistent way
- Furthermore, it:
  - supports concurrent access to the different files or directories in the tree by several users;
  - keeps a log of the changes performed to each file/directory that can be used to document/keep track of the main changes between versions
  - allows to create new **branches**, i.e. to keep track of the evolution of multiple directory trees that have a common ancestor (i.e. a tree of directory trees)

SVN Key Concepts

Repository This is the central store (and the server program) that keeps the **different versions** of the data

- It usually keeps data for different “projects”
- We’ll refer to the data of a “project” kept in the repository as ... the repository

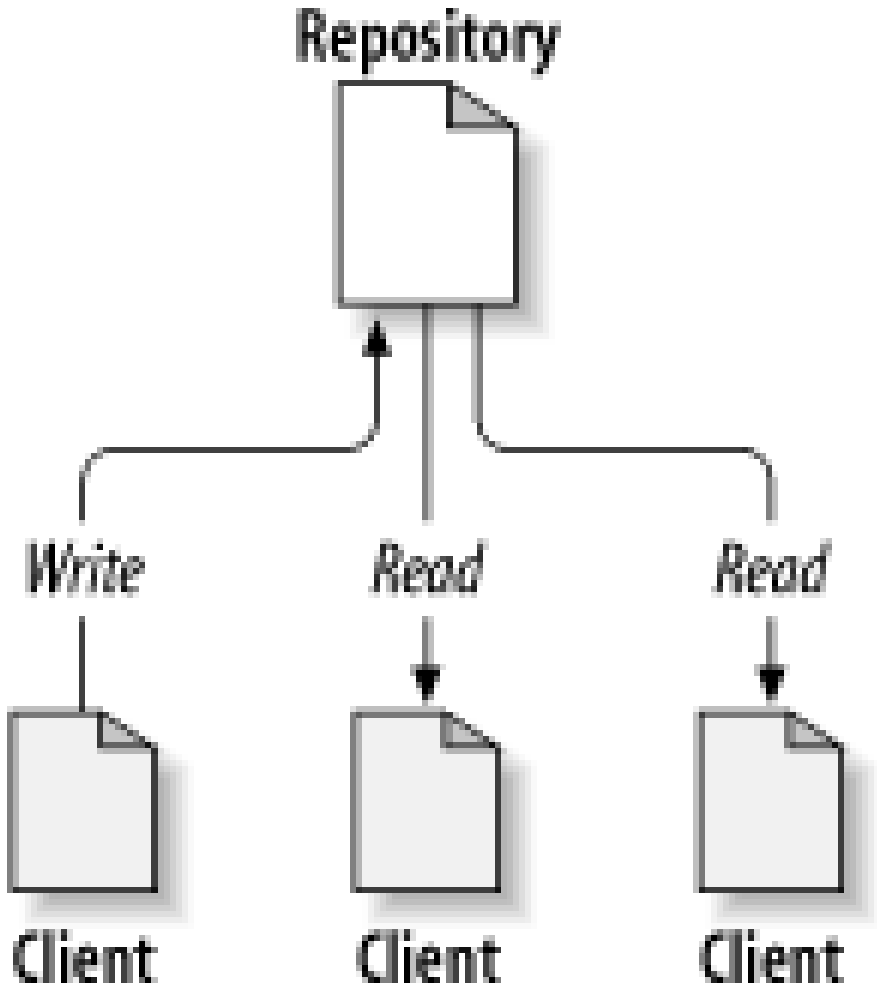
Working Copy This is a copy of **one version** of the data of a “project”. The working copy is kept in a client computer, which is usually different from the computer that keeps the repository

- The working copy is a standard directory tree
- Other programs, like editors and compilers, do not need to be “version-control-aware”

There may be several working copies of a given repository (project), thus supporting the collaboration of multiple programmers in a project

Revision A new revision (version) is created by **committing** to the repository the changes done in a working copy

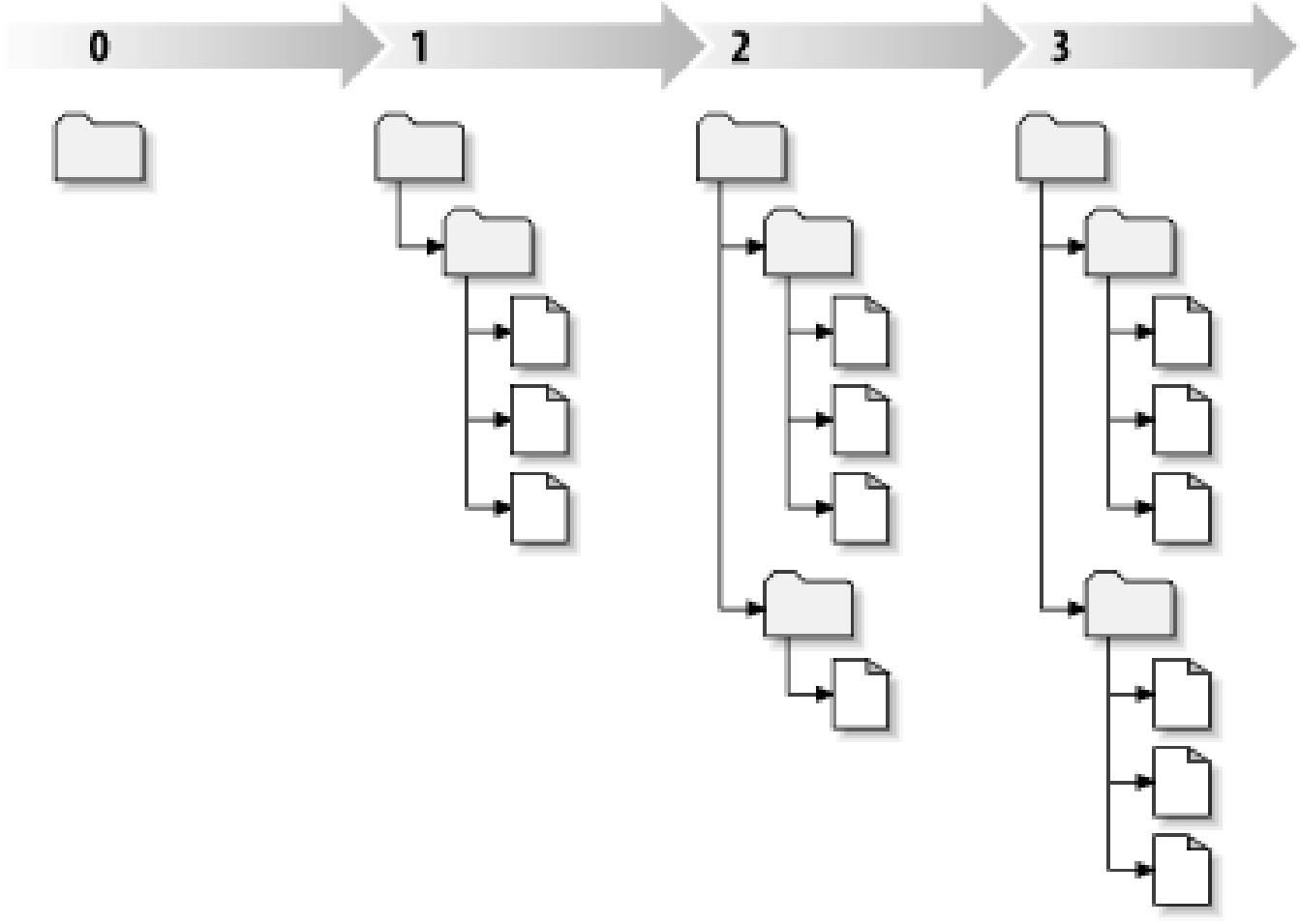
SVN: The Repository and Working Copies



Source: Ben-Collins-Sussman et al. Version Control with Subversion

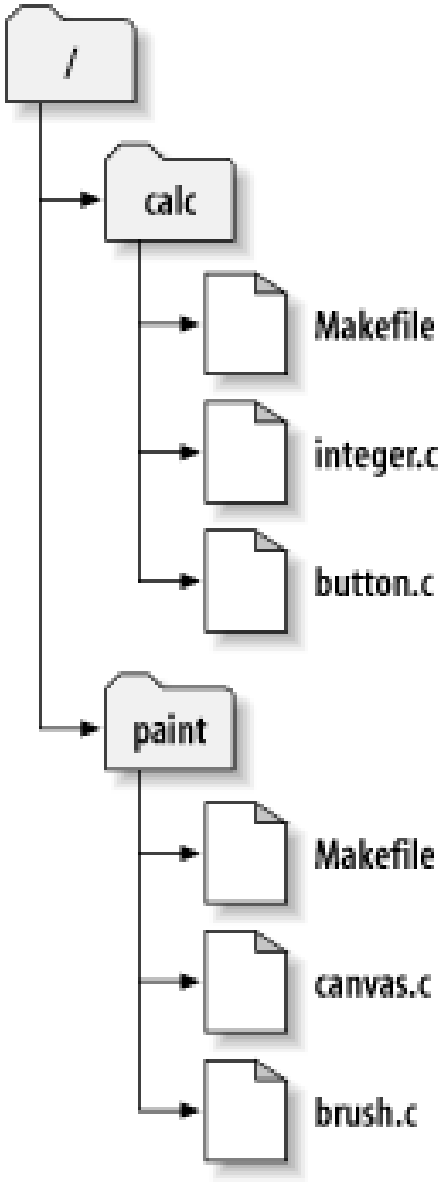
SVN: Revisions

Revision A new revision (version) is created by **committing** to the repository the changes done in a working copy



Source: Ben-Collins-Sussman et al. Version Control with Subversion

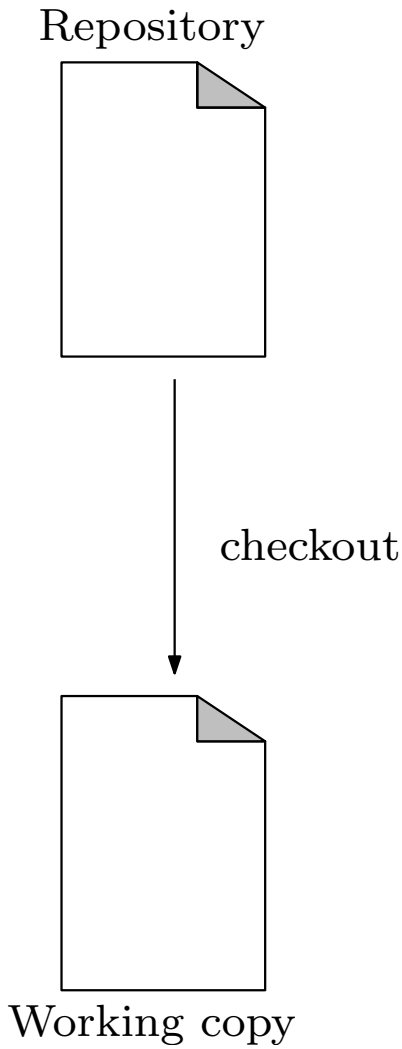
SVN: Multiple Projects



Source: Ben-Collins-Sussman et al. Version Control with Subversion

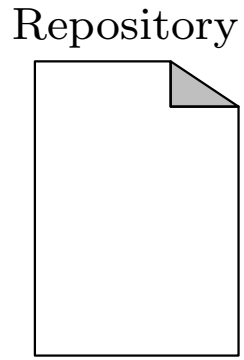
SVN: Basic Usage

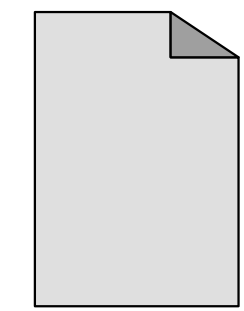
- Generate a working copy using **checkout**



SVN: Basic Usage

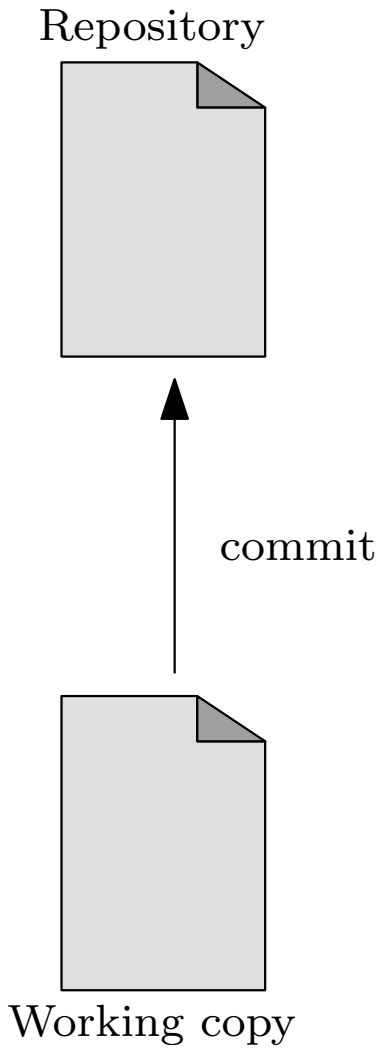
- Change the working copy with your favorite editor





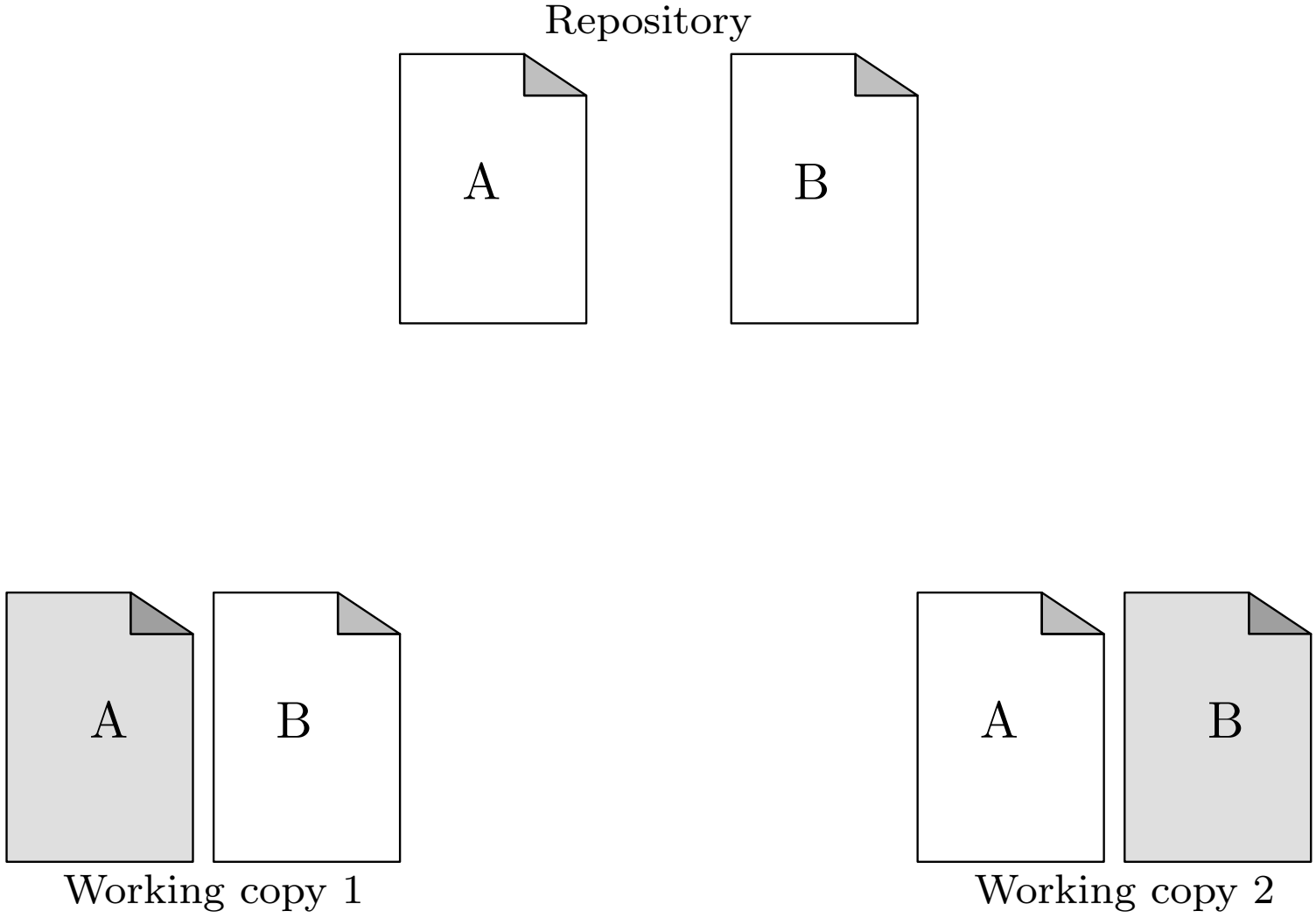
SVN: Basic Usage

- Publish your changes on the repository with **commit**



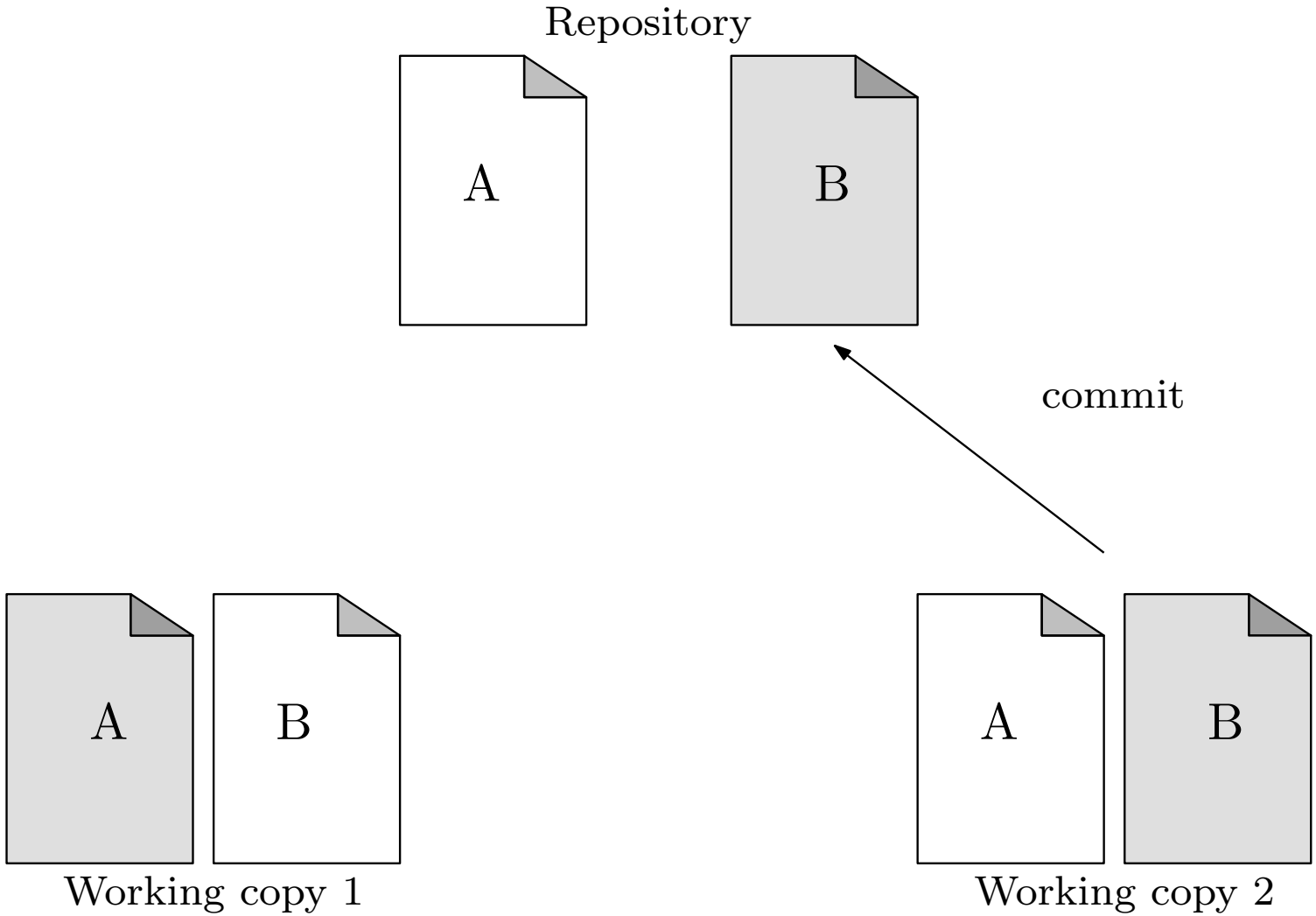
SVN: Multiple Users

- Often several users work concurrently on their own working copies, normally on different files



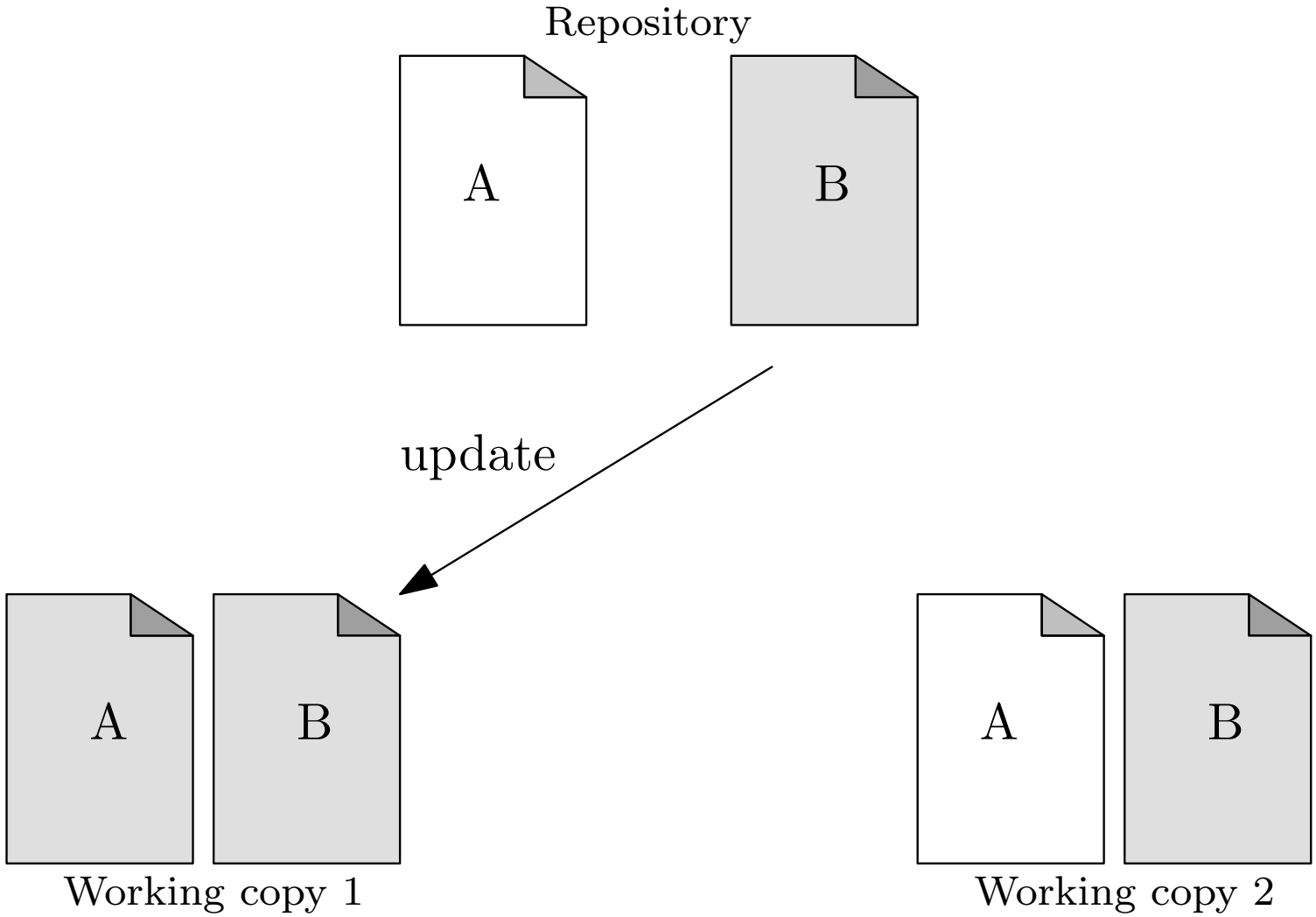
SVN: Multiple Users

- When a user commits its changes, the working copies of the remaining users become outdated

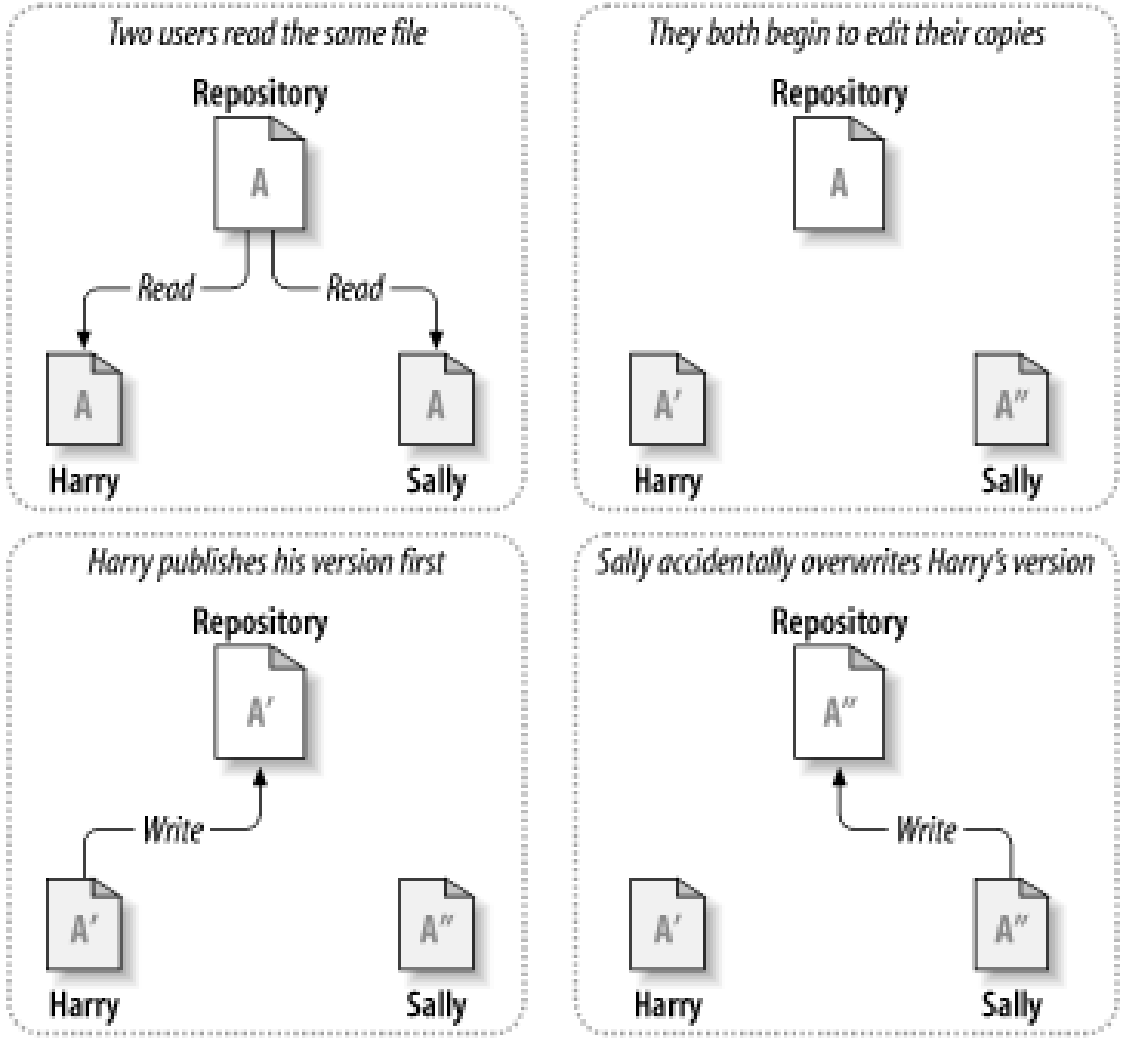


SVN: Multiple Users

- To bring its working copy in sync with the latest version of the repository a user must **update** it



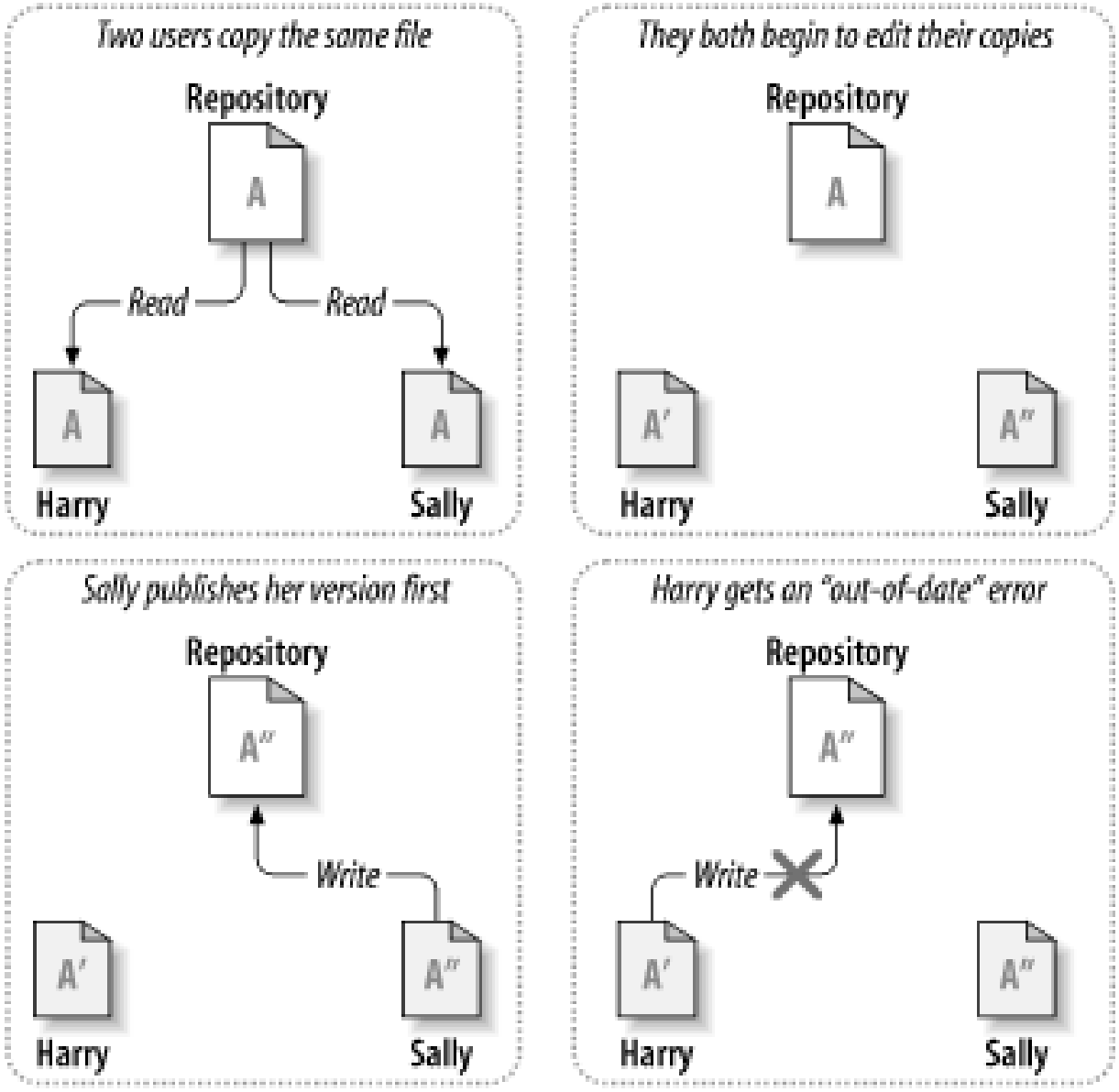
SVN: Conflicts with Multiple Users



Source: Ben-Collins-Sussman et al. Version Control with Subversion

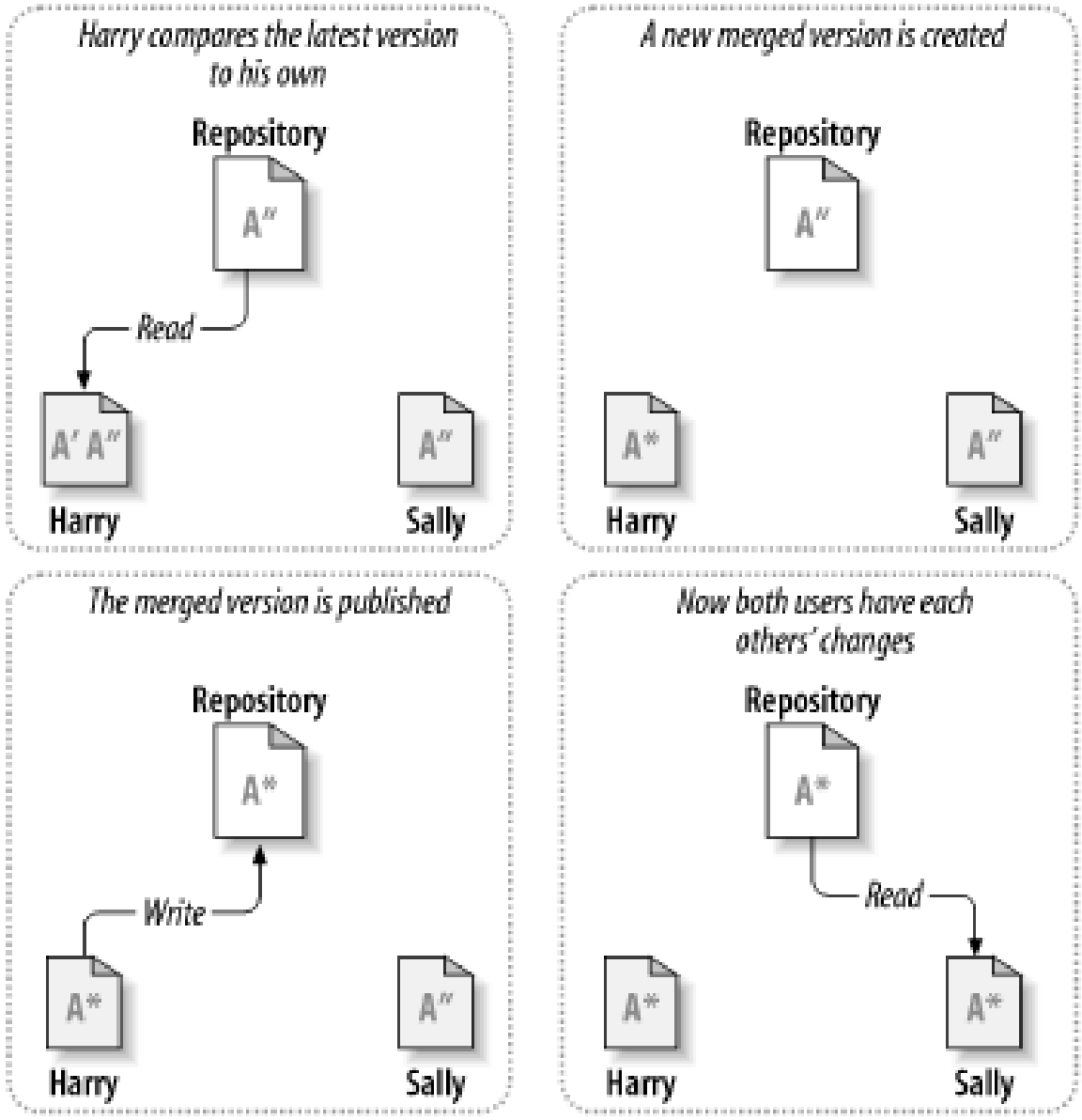
To address this problem SVN uses a versioning approach known as **Copy-Modify-Merge**

SVN: Automatic Conflict Detection



Source: Ben-Collins-Sussman et al. Version Control with Subversion

SVN: Manual Conflict Resolution



Source: Ben-Collins-Sussman et al. Version Control with Subversion

SVN: Main Commands

checkout create a working copy

add add a file/directory to the working copy

delete remove a file/directory to the working copy

move rename a file in the repository

status list changes made to the working copy

diff show differences between the working copy and the repository, or between revisions in the repository

update update the working copy

commit update the repository

log list messages with date and author information attached to revisions and which paths changed in each revision

Other commands include **mkdir** and **copy**, but they do little more than **add**

SVN Session: Creating a Repository and Adding a File

First, let's check out empty repository (which was previously created)

```
$ svn checkout file:///home/pedro/svn/at/
A    at
Checked out revision 1.
$ cd at

$edit foo.c, make, run, edit, make, run, ...
```

OK! Now that it enters in graphic mode, let's make a backup

```
$svn add foo.c
 A             foo.c
$svn commit -m "Enters graphic mode" foo.c
Adding          foo.c
Transmitting file data .
Committed revision 2.
$edit foo.c, make, run, edit, make, run, ...
```

SVN Session: Committing Chantes to a File

OK! Now that it maps graphics' memory, let's make another backup

```
$ svn commit -m "Memory maps graphics' memory" foo.c
Sending        foo.c
Transmitting file data .
Committed revision 3.
$edit foo.c, make, run, edit, make, run, ...
```

Hmmm, let's see version 2

```
$ svn update -r 2 foo.c
U    foo.c
Updated to revision 2.
$edit foo.c
```

OK! Let's retrieve the latest version

```
$ svn update foo.c
U    foo.c
Updated to revision 3.
$edit foo.c
```

SVN Session: Modification History

Hmm, in which version was memory mapping added?

```
$ svn log foo.c
------------------------------------------------------------------------
r4 | pedro | 2010-11-16 17:04:14 +0000 (Tue, 16 Nov 2010) | 1 line

Draws pixel
------------------------------------------------------------------------
r3 | pedro | 2010-11-16 16:48:41 +0000 (Tue, 16 Nov 2010) | 1 line

Memory maps graphics memory
------------------------------------------------------------------------
r2 | pedro | 2010-11-16 16:40:41 +0000 (Tue, 16 Nov 2010) | 1 line

Enters in graphics mode
------------------------------------------------------------------------
```

SVN: Checkout and Log

Weekend and I have forgotten LCOM's project! (#%@&#$)
Wait ... "Sem problemas":

```
$svn checkout <URL>/svn/at/ foo.c
A    foo.c/foo.c
Checked out revision 4.
```

Did Zé add something new?

```
$ svn log foo.c
------------------------------------------------------------------------
r5 | zemanel | 2010-11-16 17:15:41 +0000 (Tue, 16 Nov 2010) | 1 line

Draws line
------------------------------------------------------------------------
r4 | pedro | 2010-11-16 17:04:14 +0000 (Tue, 16 Nov 2010) | 1 line

Draws pixel
[...]
```

SVN Session: Diff and Update

What did Zé do?

```
$ svn diff -r 5:4 foo.c
Index: foo.c
===================================================================
--- foo.c (revision 5)
+++ foo.c (revision 4)
@@ -16,7 +16,3 @@
 void draw_pixel() {

 }
-
-void draw_line() {
-
-}
```

Great! Let's update my copy:

```
$svn update
Conflict discovered in 'foo.c'.
Select: (p) postpone, (df) diff-full, (e) edit,
        (mc) mine-conflict, (tc) theirs-conflict,
        (s) show all options: r
G    foo.c
Updated to revision 5.
```

SVN Session: Status

Not exactly what we were expecting:

```
 ...
 void draw_pixel() {
 }
+<<<<<<< .mine
+
+void draw_rectangle() {
+}
+=======
+
+void draw_line() {
+}
+>>>>>>> .r5
```

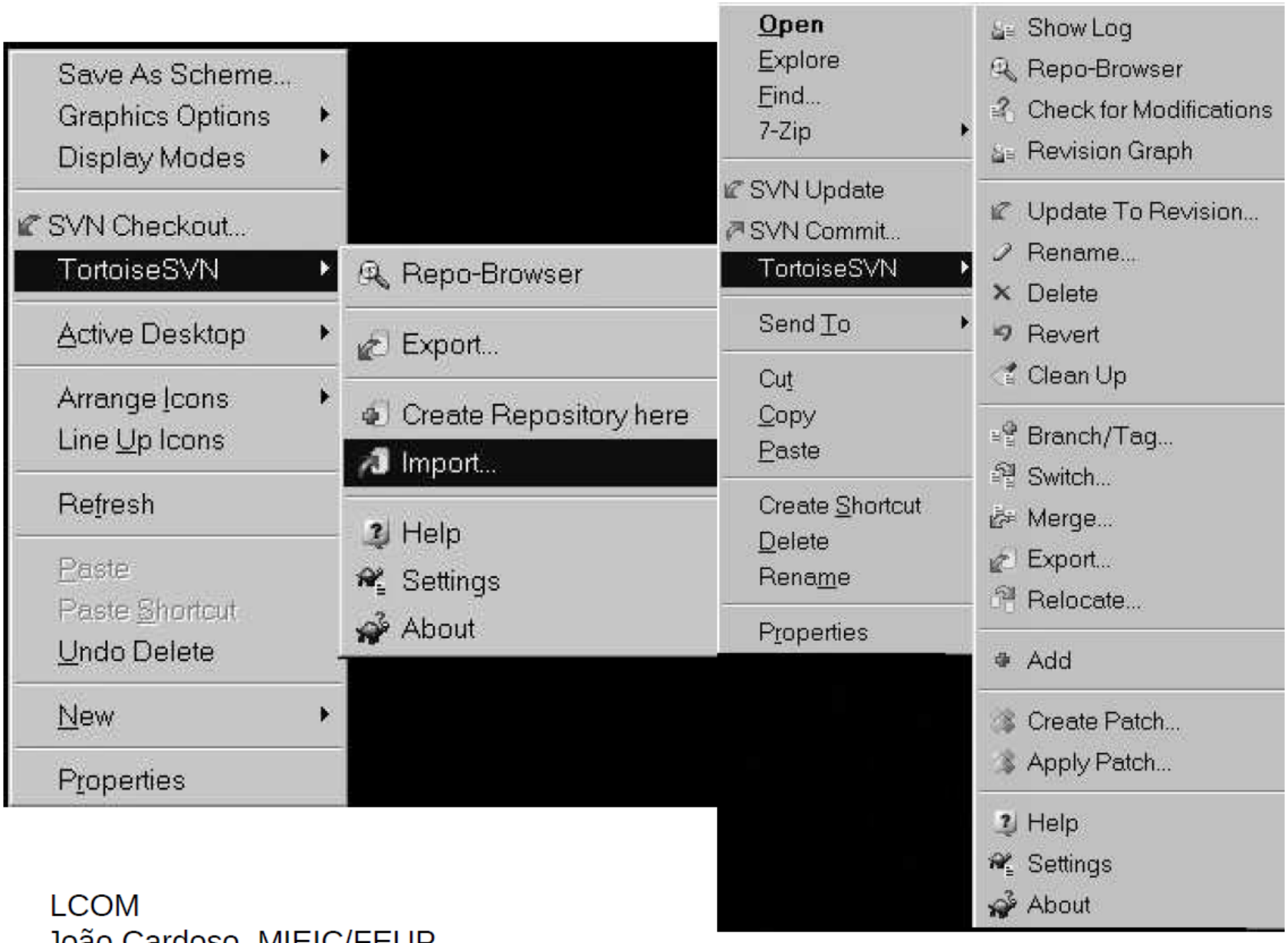
but it can be fixed ...
OK! What's the current status of our working copy?

```
$ svn status -u -v foo.c
M                5        5 pedro        foo.c
Status against revision:      5
```

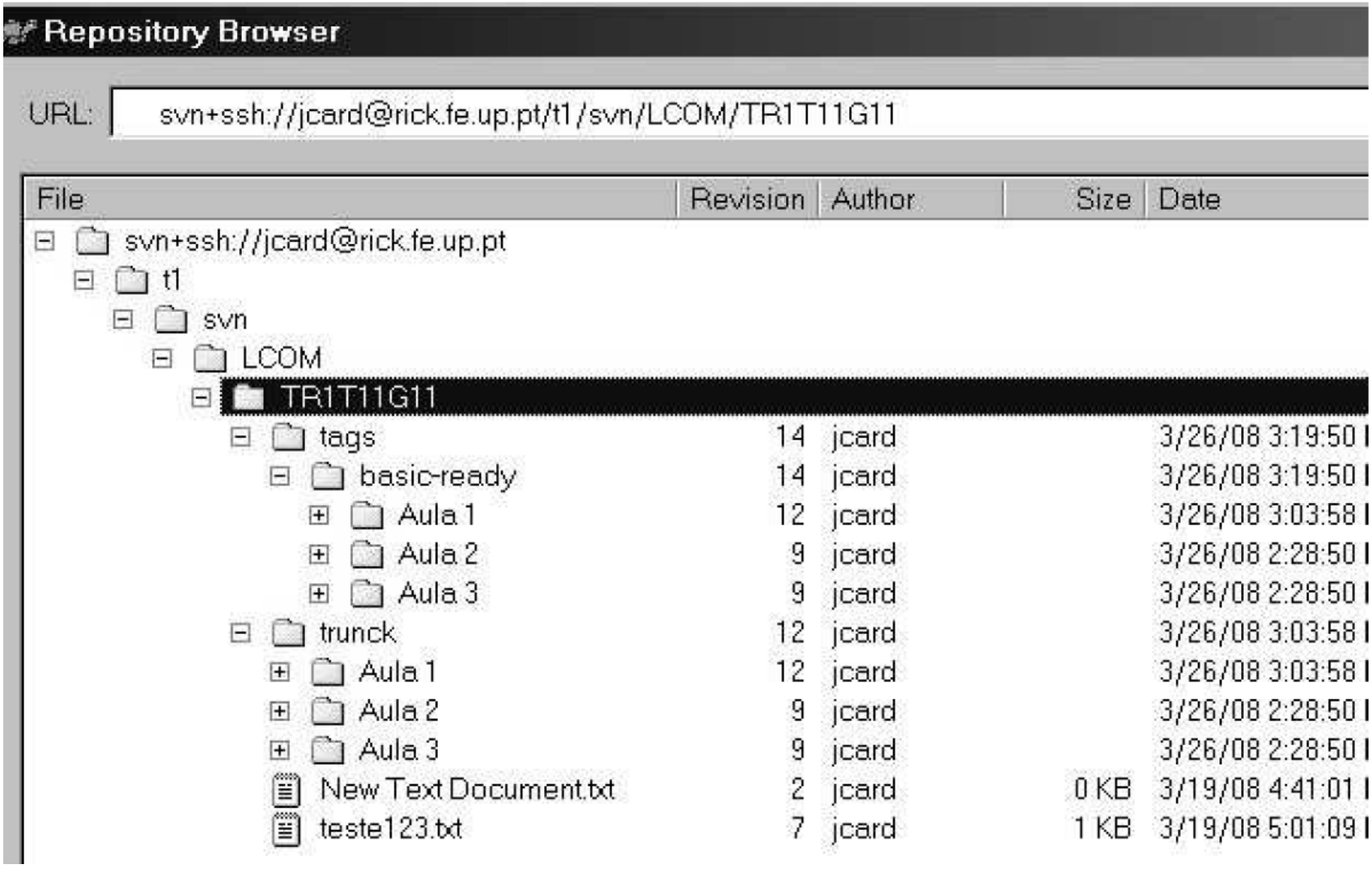
SVN: Advantages

- It provides automatic backup
- It makes it easy to restore a previous version
- It is supported by most IDEs
- Users can work on any computer
- Members of a team can work simultaneously and independently on the same project
- It logs **who** did/committed **what** and **when**
  - By using appropriate messages or comments, it is also possible to know **why**
- It is possible to try a new approach, and continue development on the older one

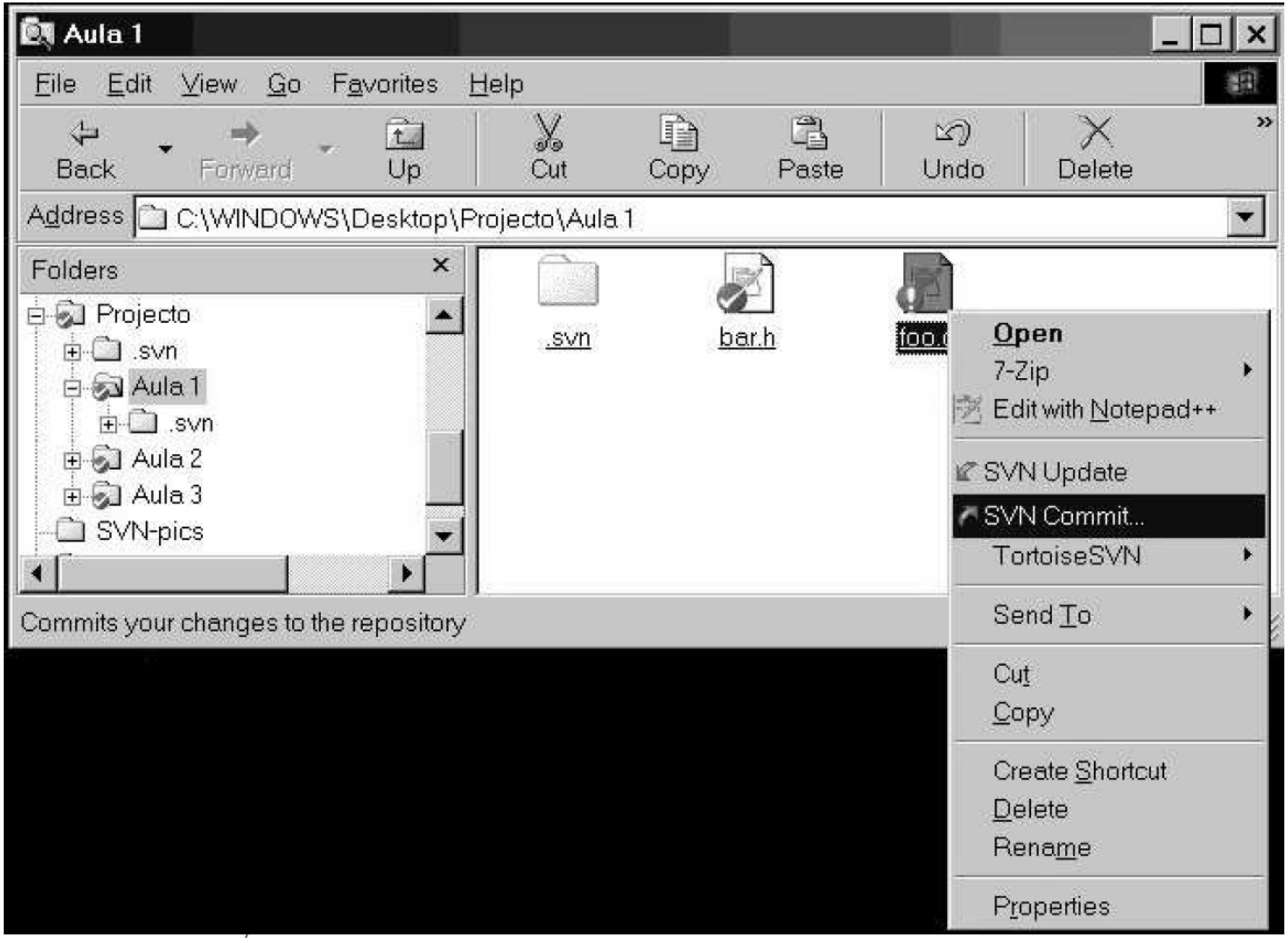
TortoiseSVN-1



TortoiseSVN-2



TortoiseSVN-3



Thanks to:

I.e. shamelessly translated material by:

- João Cardoso (jcard@fe.up.pt)

Further Reading

- Serch the web for the right tutorial for you on
  - SVN
  - TortoiseSVN
- *Ben Collins-Sussman et. al* Version Control with Subversion [DRAFT] For Subversion 1.6