Gioele Barabucci
Cologne Center for eHumanities
2015-11-23

Version control system, in teoria ed in pratica

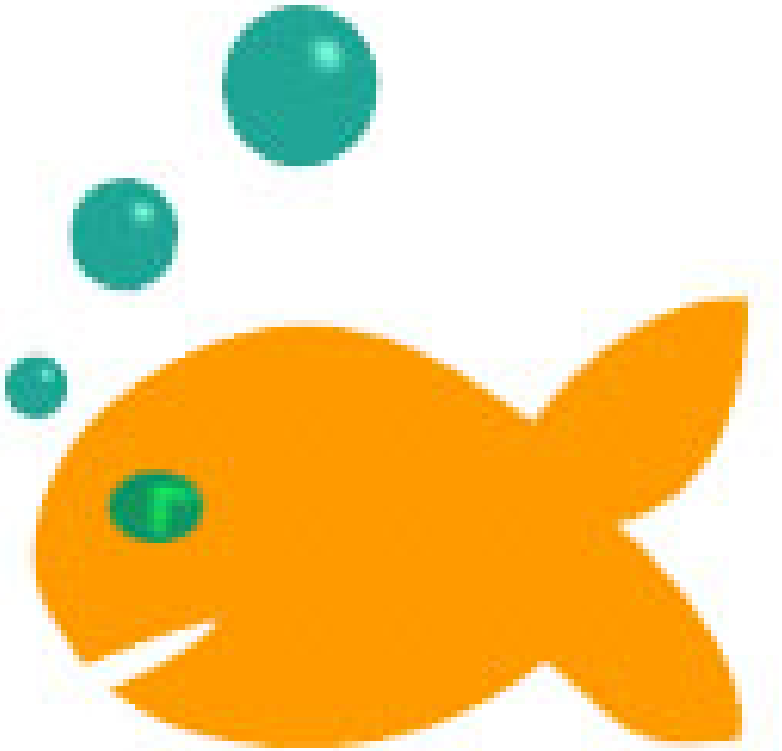
Universität zu Köln

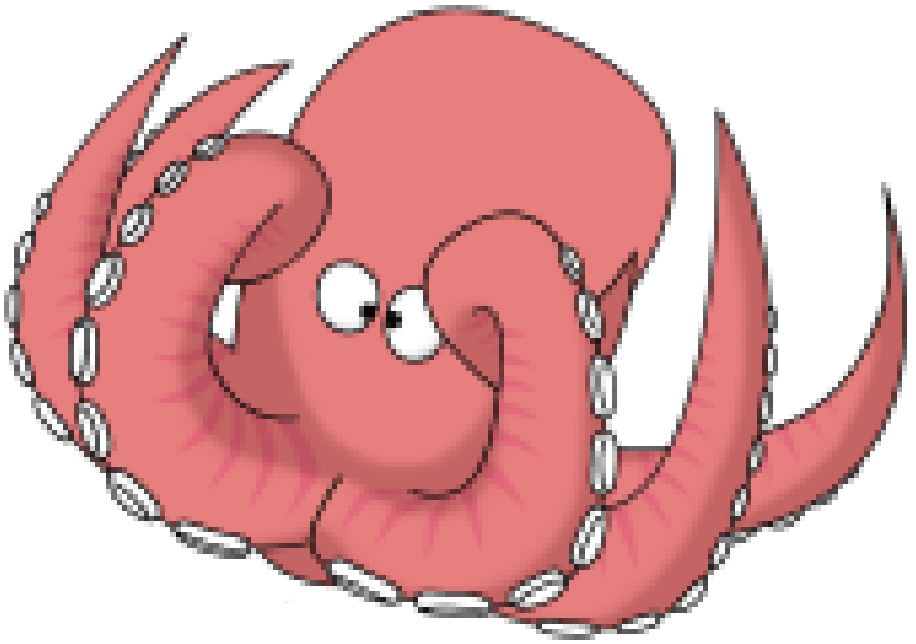
Chi è?

Version control system

(sistemi di versionamento)

hg logo, Matt Mackall, GPL





file,v

In principio fu il file

Il file

```
print “ciao”
```

robo.c

Il file

```
print “ciao $USER”
```

robo.c

Il file

```
print “ciao ” +
   $(whoami)
```

robo.c

Il file

```
print “ciao ” +
    $(getent passwd $USER)
```

robo.c

Il file

```
print “ciao ” +
    $(getent passwd $USER)
```

robo.c

Un backup al giorno

```
20 6 * * * tar -c \
    -f ~/backups/robo-$(date).tar.gz \
    ~/robo/
```

crontab

Un backup al giorno?

```
20 6 * * * tar -c \
    -f ~/backups/robo-$(date).tar.gz \
    ~/robo/
```

crontab

Un backup ogni salvataggio

```
:autocmd BufWritePost * !tar -c \
    -f ~/backups/robo-$(date).tar.gz \
    ~/robo/
```

Un backup ogni salvataggio

```
:autocmd BufWritePost * !tar -c \
    -f ~/backups/robo-$(date).tar.gz \
    ~/robo/
```

VMS/Files-11, 1977

Tutte le versioni salvate, tutte!

```
$ ls
robo.c    robo.c.8  robo.c.16 robo.c.24
robo.c.1  robo.c.9  robo.c.17 robo.c.25
robo.c.2  robo.c.10 robo.c.18 robo.c.26
robo.c.3  robo.c.11 robo.c.19 robo.c.27
robo.c.4  robo.c.12 robo.c.20 robo.c.28
robo.c.5  robo.c.13 robo.c.21 robo.c.29
robo.c.6  robo.c.14 robo.c.22 robo.c.30
robo.c.7  robo.c.15 robo.c.23 robo.c.31
```

Manca il concetto di *versione*

Una versione *è*
un certo nome
che associamo ad un
certo stato del software.

Nomi

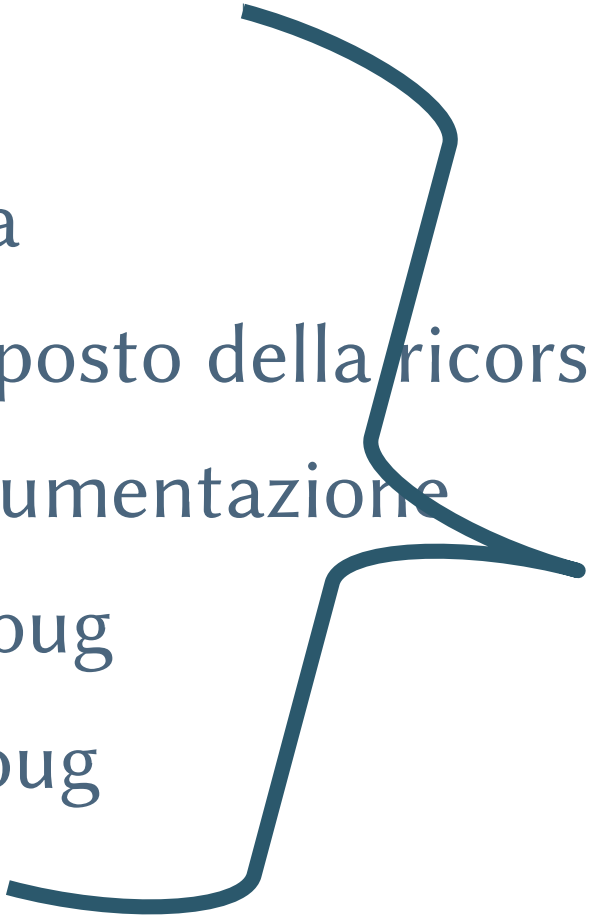
- Numerici
  - `1`, `2`, `3`
- Numerici con gerarchie
  - `1.1`, `1.2`, `1.3`, `2.1`, `2.2`, `3.1`
- Esadecimali
  - `0ffa297d6bb2e`
- Strighe
  - `v12_post_review`
- Strutturati
  - `robo-main-1.2-be0d4`

Versioni

- V0: file vuoto
- V1: prima stesura
- V2: iterazione al posto della ricorsione
- V3: aggiunta documentazione
- V4: risolti tutti i bug
- V5: risolto altro bug

Versioni

- V0: file vuoto
- V1: prima stesura
- V2: iterazione al posto della ricorsione
- V3: aggiunta documentazione
- V4: risolti tutti i bug
- V5: risolto altro bug



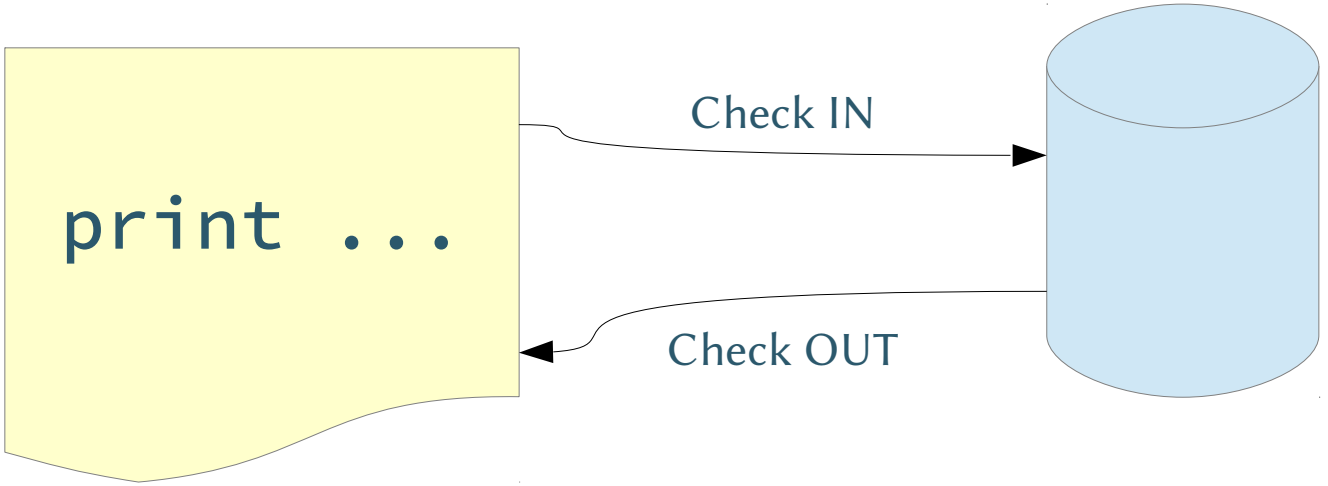
storia
(ma anche *log*)

Commit

Si genera un commit *o*
si fa commit *quando*
si da un nome
ad uno stato

Facciamo un programmino

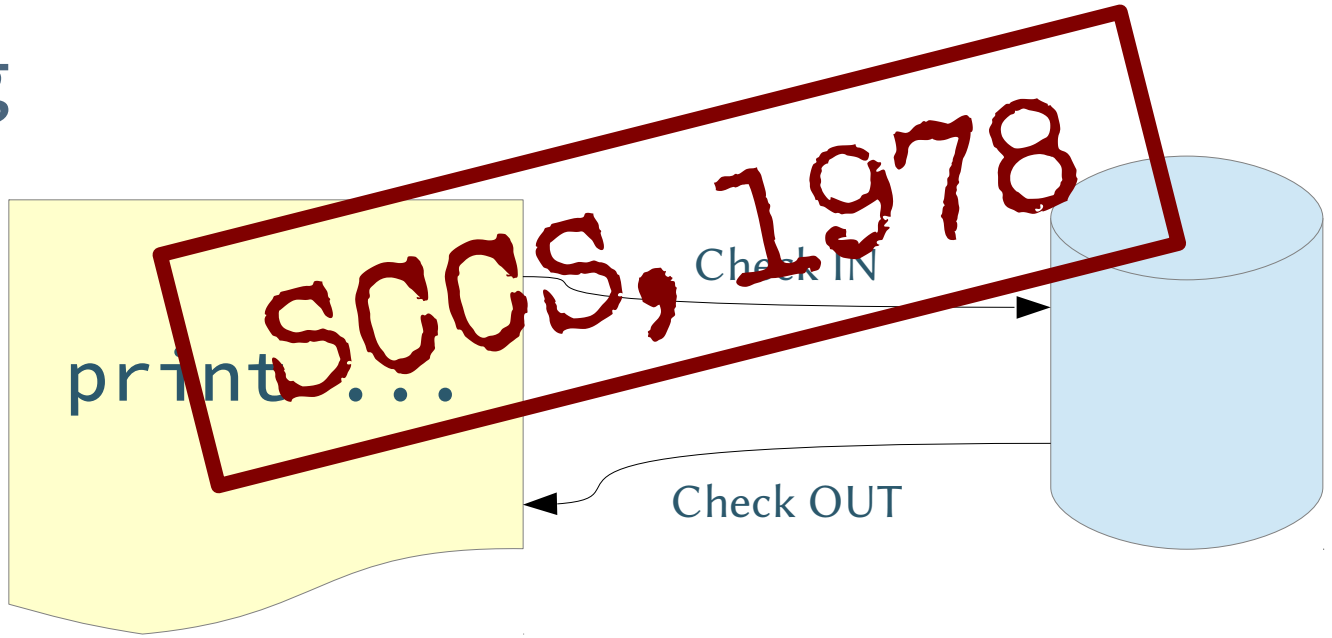
```
checkin robo.c
checkout robo.c -v 1.2
log
```



robo.c Repository

Facciamo un programmino

```
checkin robo.c
checkout robo.c -v 1.2
log
```



Bello il robo, ma...

«potresti fare una versione che usa la nuova libreria per i calcoli libfast?»

- Le modifiche sono molto invasive;
- bisogna cambiare molti punti;
- funziona solo con computer nuovissimi.

Bello il robo, ma...

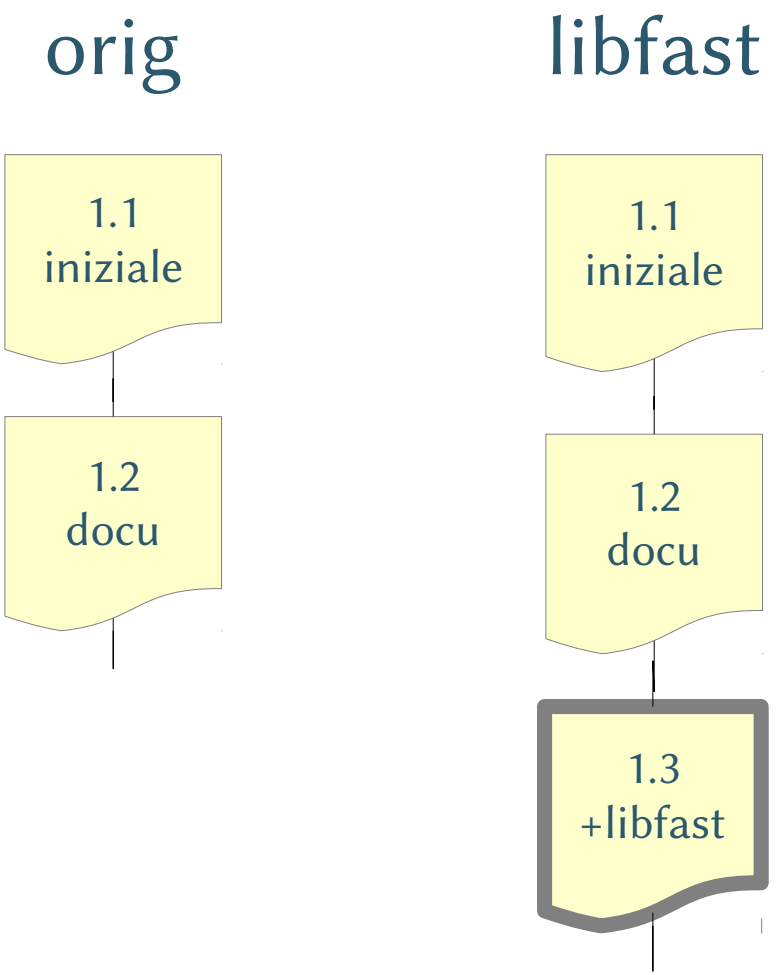
«potresti fare una versione che usa la nuova libreria per i calcoli libfast?»

- Le modifiche sono molto invasive;
- bisogna cambiare molti punti;
- funziona solo con computer nuovissimi.

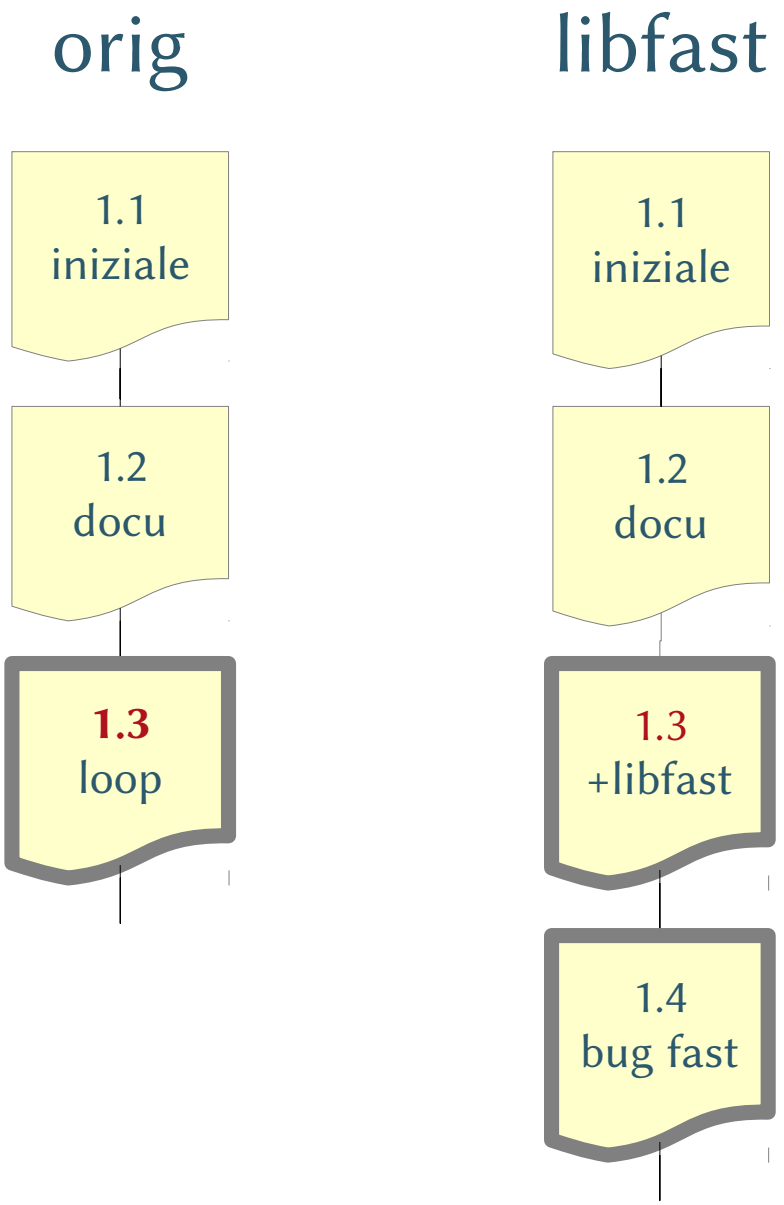
...faccio una copia della cartella,

l'originale non lo tocco

Vite parallele



Vite parallele... quasi

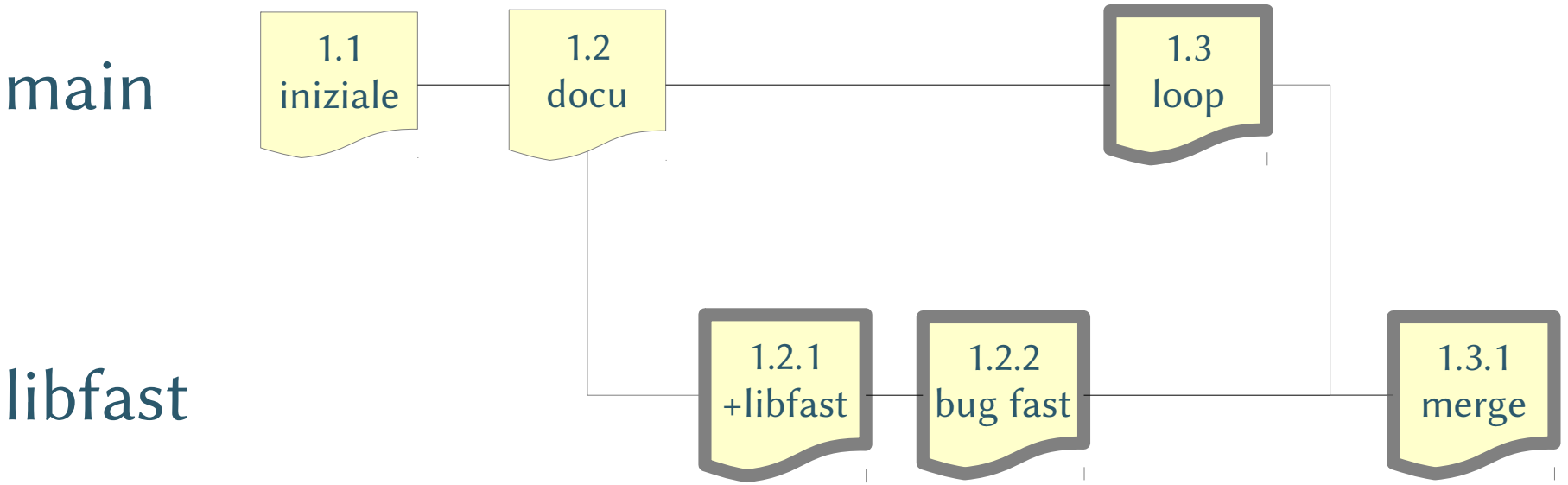


Il concetto di *branch*

Un branch *è una serie di versioni.*

Alcuni branch partono da un branch genitore poi seguono una loro strada (parzialmente) indipendente.

Branch


main
libfast
1.1
iniziale
1.2
docu
1.3
loop
1.2.1
+libfast
1.2.2
bug fast
1.3.1
merge

Modifichiamo il programmino

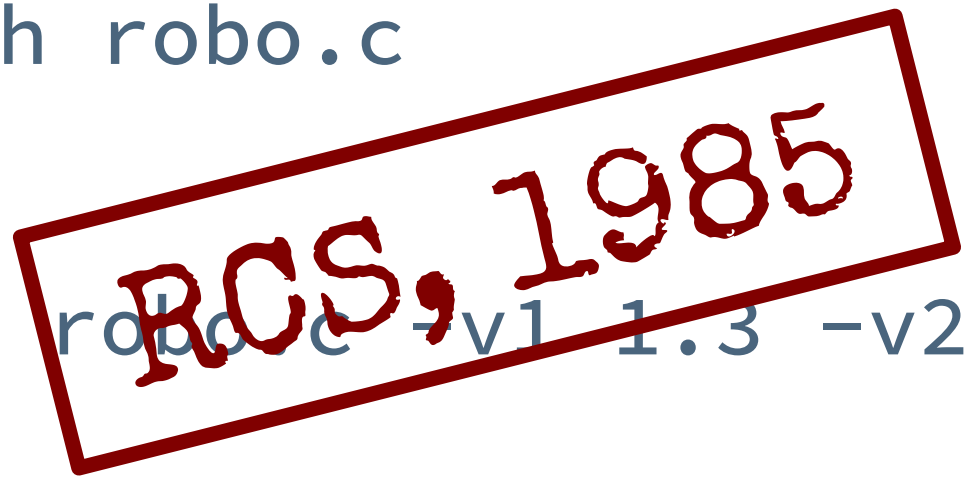
```
branch robo.c

merge robo.c -v1 1.3 -v2 1.2.2
```

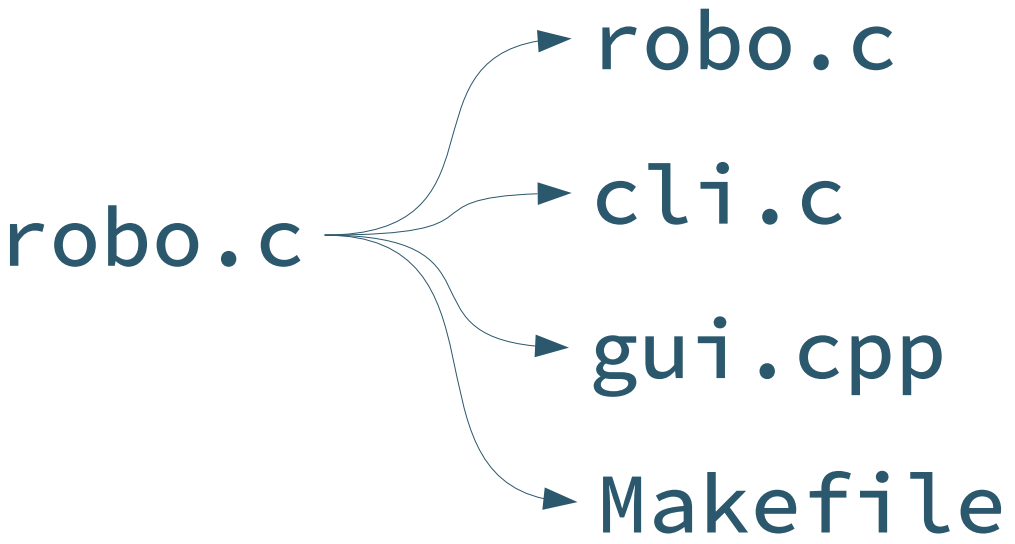
Modifichiamo il programmino

```
branch robo.c

merge robo.c -v1 1.3 -v2 1.2.2
```



Robo just got serious



Un po' di lavoro e otteniamo

```
robo.c      1.31, 2.18
cli.c       1.14
gui.cpp     1.16
Makefile     1.5
fast.c       1.1, 2.21
```

Commit di una directory, non di un file

~~commit robo.c~~

```
commit robo/
```

Commit di una directory, non di un file

~~commit robo.c~~

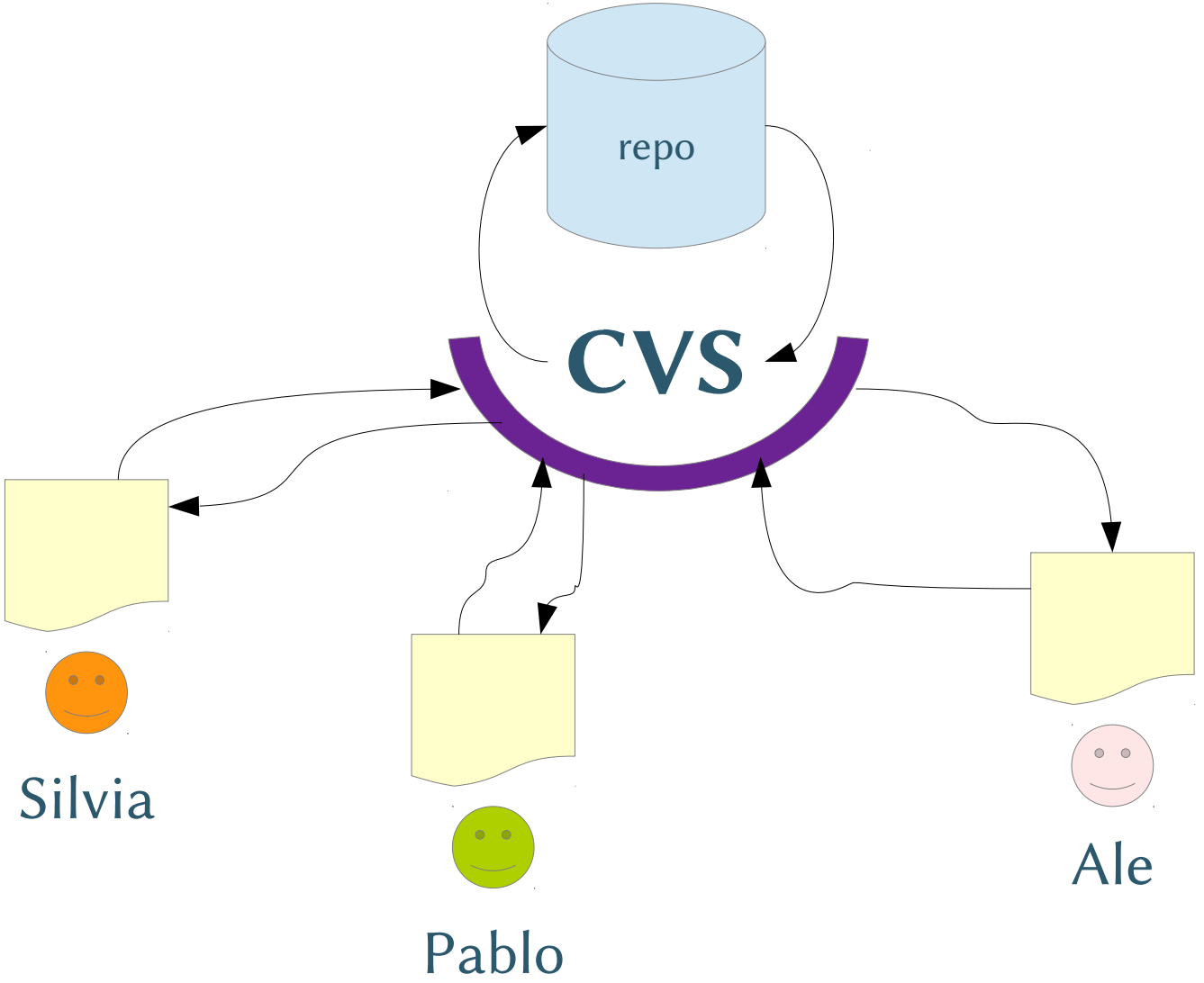
commit robo/

Robo non sentirti solo

Il progetto è usatissimo, arrivano richieste di nuove funzionalità, uno sviluppatore non basta più.

È ora di passare da sviluppatore a gruppo di sviluppatori.

C is for Concurrent



Prima di fare commit CVS

- Controlla che
  - l'utente abbia l'ultima versione;
  - non ci siano stati commit tra il suo checkout e adesso.

  Se una di queste cose non è vera allora c'è un conflitto da risolvere manualmente.

  Poi ci sono le cose che CVS non controlla:
  - atomicità, dipendenze, etc...

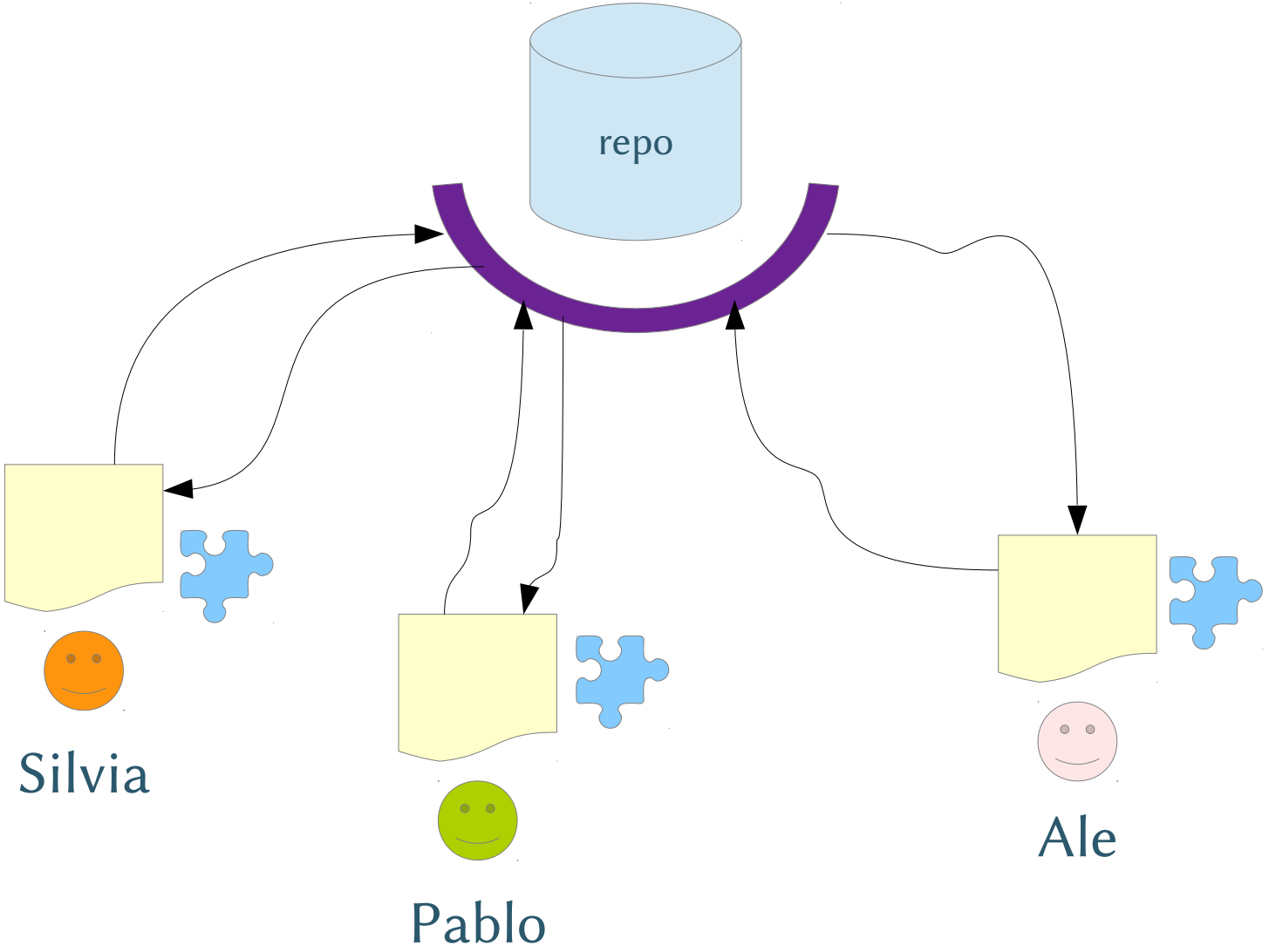
Il guardiano CVS

- Per garantire la concorrenza degli accessi e dei commit, tutto passa per il server.
  - commit
  - checkout
  - log
  - diff

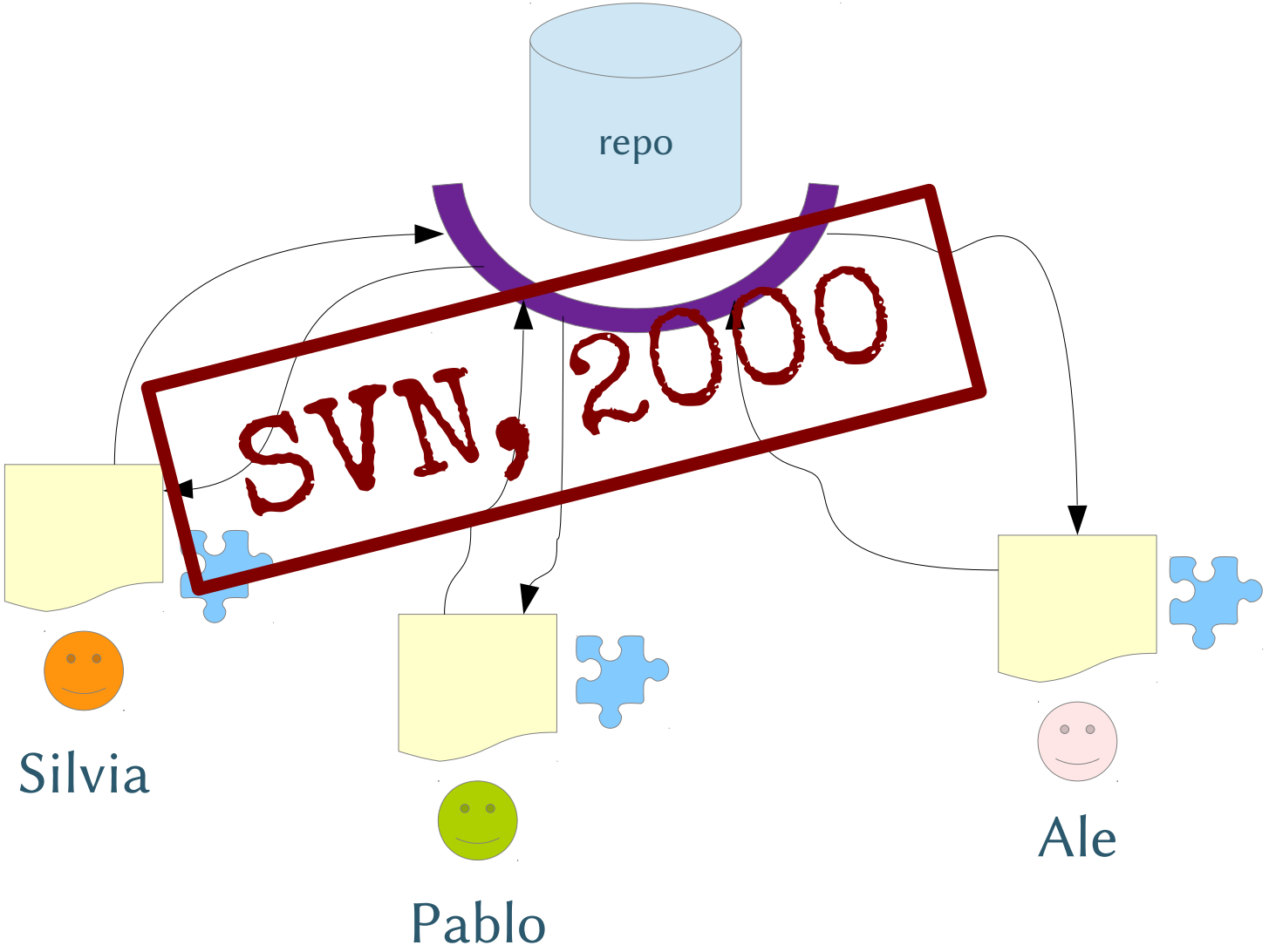
Silvia, Pablo e Ale sono sempre in viaggio

- Vorrebbero lavorare un po' al robo ma non c'è connessione in
  - treno, aereo, bus, bar, talvolta a casa...
- E come lavoro io senza diff, log checkout e via dicendo?

Checkout file + storia



Checkout file + storia



Da CVS a SVN

CVS	SVN
✘ checkout	✘ checkout
✘ commit	✘ commit
✘ diff	✔ diff
✘ log	✔ log
✘ branch	✔ branch

✘ = rete, ✔ = locale

Da CVS a SVN

CVS	SVN
✘ checkout	✘ checkout
✘ commit	✘ commit
✘ diff	✔ diff
✘ log	✔ log
✘ branch	✔ branch **(più o meno)**

✘ = rete, ✔ = locale

Farewell to branches

~/robo/

- trunk/
  - SVN/
  - robo.c
  - Makefile
- branches/
  - libfast/
    - SVN/
    - robo.c
    - Makefile
  - darwin/
    - SVN/

      ...

Bella idea. In teoria.

> if you know what you're doing, you shouldn't (and shouldn't ever have to) re-merge a branch in the first place. (Of course it's difficult to do when you're doing something fundamentally wrong and silly!)

And therein lies the source of your confusion and the whole problem in general.

You say that merging branches is "fundamentally wrong and silly". Well, that's exactly the problem: you're thinking of branches as things that *shouldn't* be merged. Why? Because you're an SVN user who knows that merging branches is *hard*. Therefore, you never do it, and you encourage others to not do it. You have been *trained* to avoid merging; you've developed techniques that you use to *avoid* merging.

I'm a Mercurial user. Even on my own projects, where I'm the only developer, I merge branches *all the time*. I have a release branch, which I put a fix into. Well, I merge that back into the main-line so that the fix goes there.

If I were using SVN, I would adopt a completely different structure of the codebase. Why? Because SVN makes merges hard, and therefore you develop idioms and techniques to *avoid* doing complex merges.

DVCS's make complex merges easy because they are the *default state*. Everything is a branch, more or less, in a DVCS. So the entire structure of them is built from the ground up to make merging easier. This allows you to develop a workflow that uses merging on a daily basis, rather than the SVN workflow where you never use merging.

The simple fact is this: you should approach a DVCS in a different way than SVN. You should use the proper idioms for these very different kinds of version control systems. In SVN, you adopt idioms that don't involve merging because merges are hard. In DVCS's, you adopt idioms that frequently use merges because they're no big deal.

Right tool for the right job.

The thing is, the merge-focused workflow is a *lot* nicer and easier to use than the SVN-style workflow where you don't merge things. It's easier to see when something from the release branch was brought into the dev branch. It's easier to see the various interplay between branches. It's easy to create test branches for things, then clip them off if the test doesn't work. And so on.

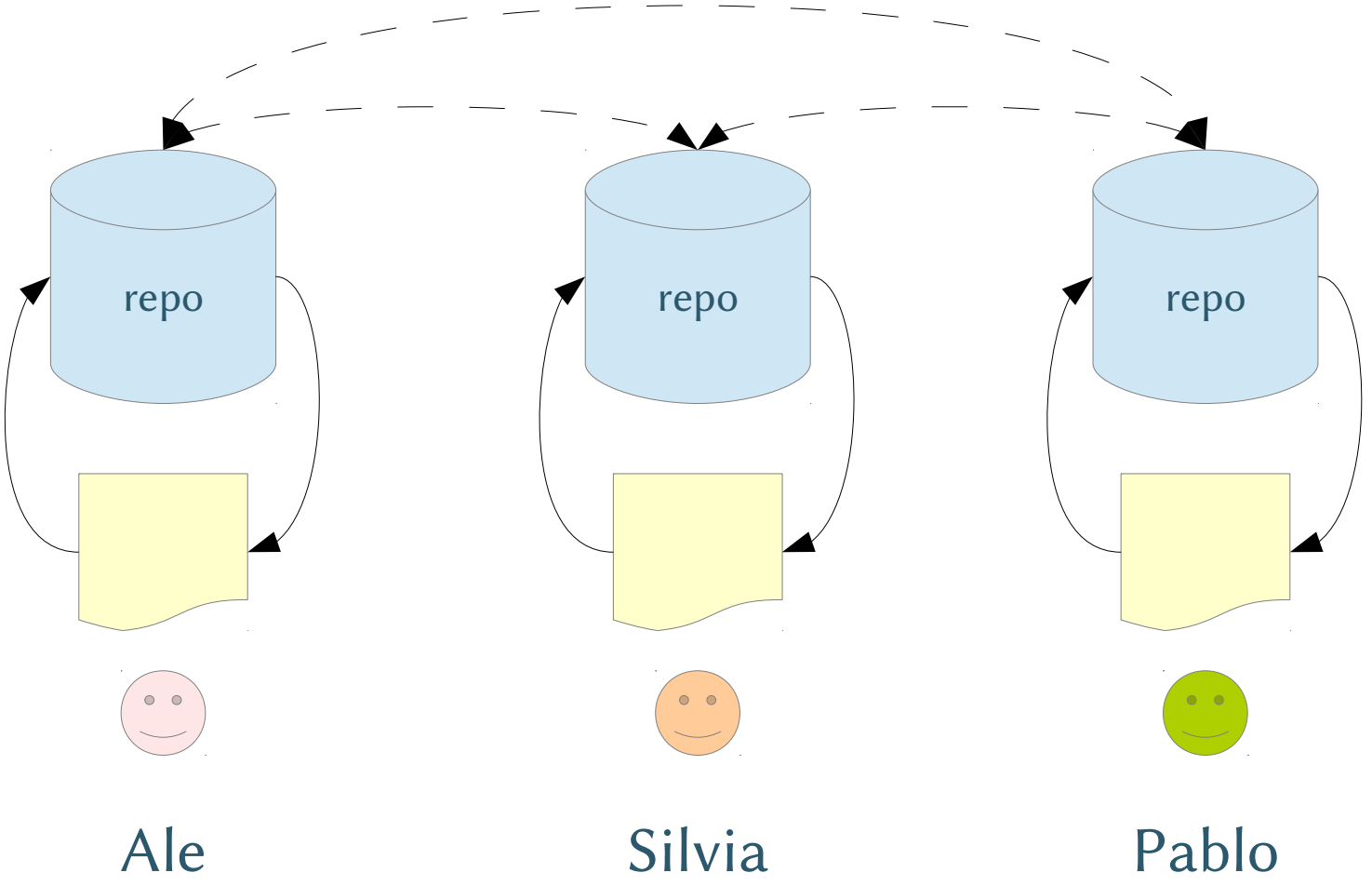
Really, Joel explains this a lot better than I can. You should have a good read of that.

E non parliamo di politica...

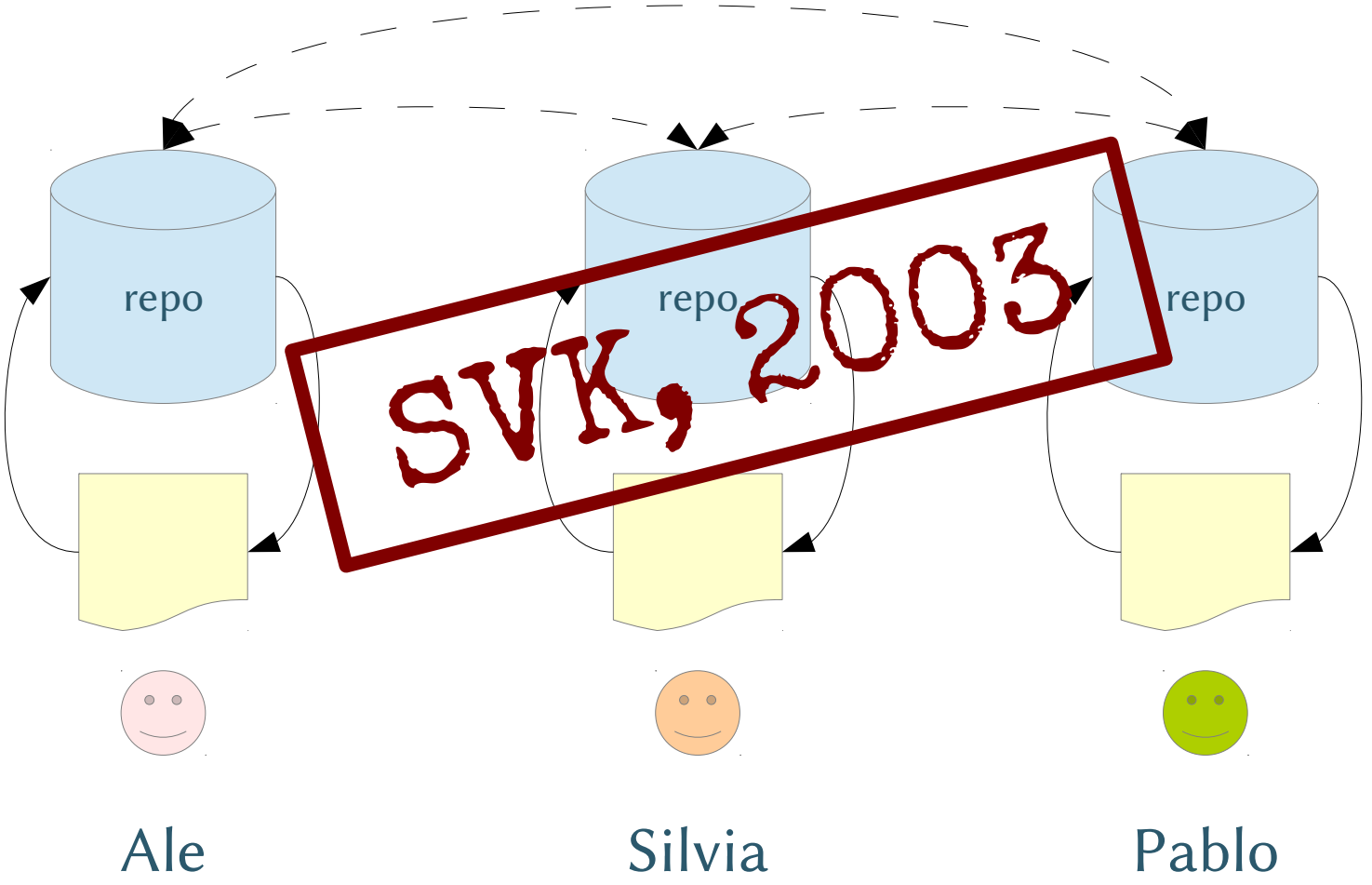
Chi ha accesso all'SVN?

Chi ha accesso *in scrittura* all'SVN?

Distribute all the things



Distribute all the things



Da SVN a SVK

SVN	SVK
✘ checkout	✔ checkout
✘ commit	✔ commit
✔ diff	✔ diff
✔ log	✔ log
✘/✔ merge	✘ merge (!!!)

✘ = server, ✔ = locale

Perché i merge sono difficili?

- Perché SVN e SVK non sanno

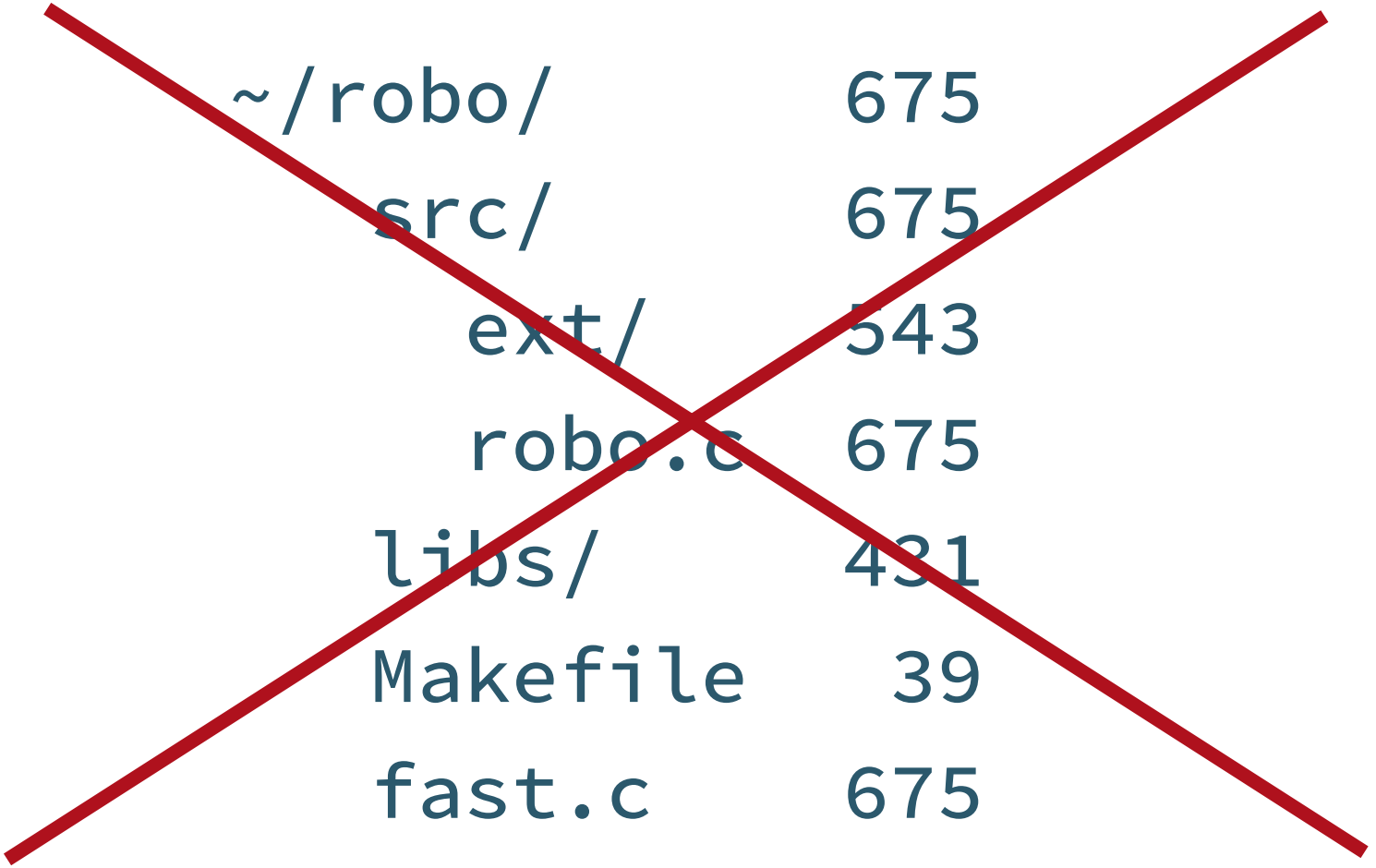
  - da quale version partire con il merge
    - qual è l'avo in comune più vicino?
    - versioni lineari, ma solo in un repo

  - qual è la situazione generale delle modifiche
    - commit per directory, non per progetto

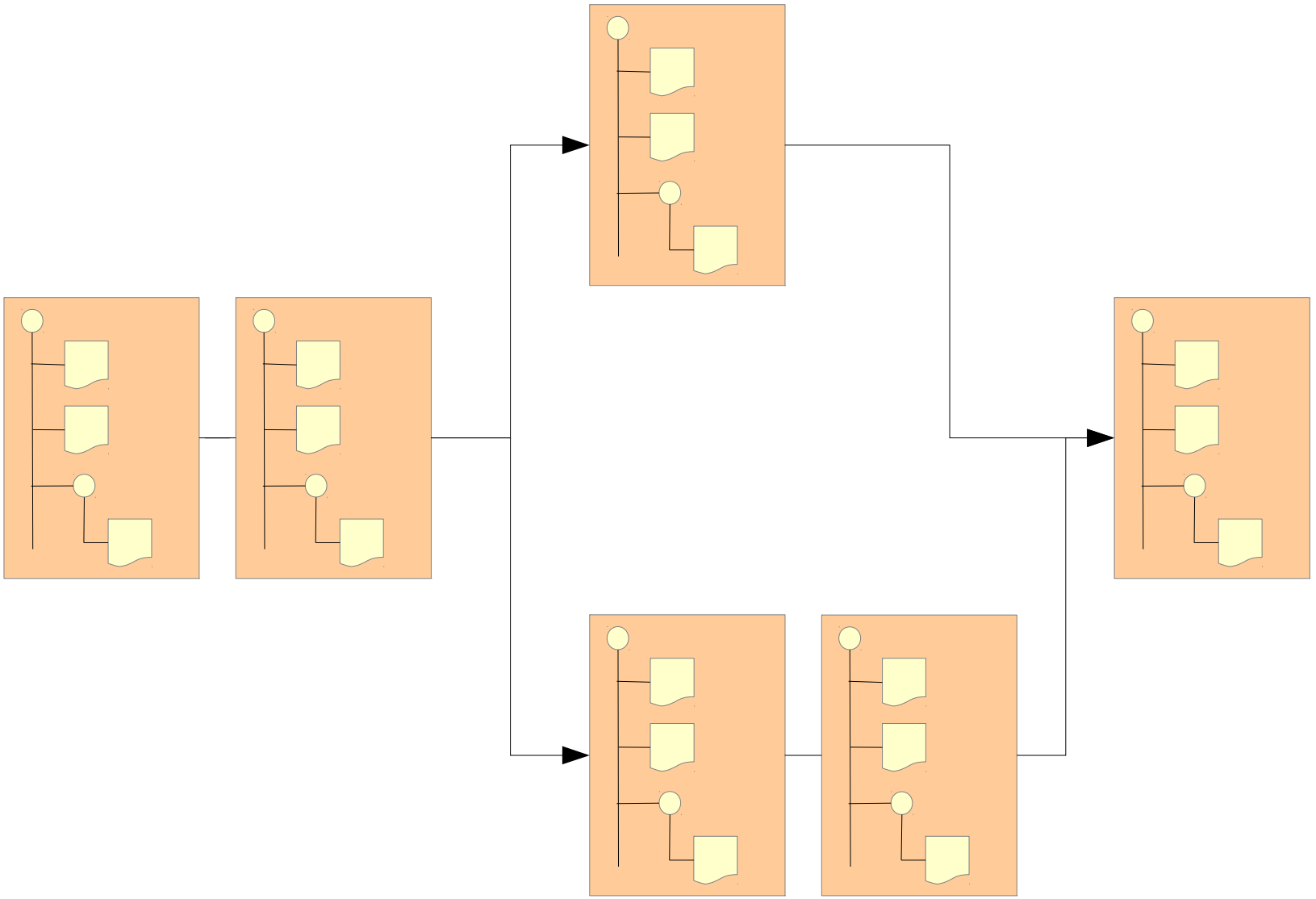
Soluzione: contenuto totale albero, non directory

```
~/robo/          675
   src/          675
      ext/       543
      robo.c     675
   libs/         431
   Makefile       39
   fast.c        675
```

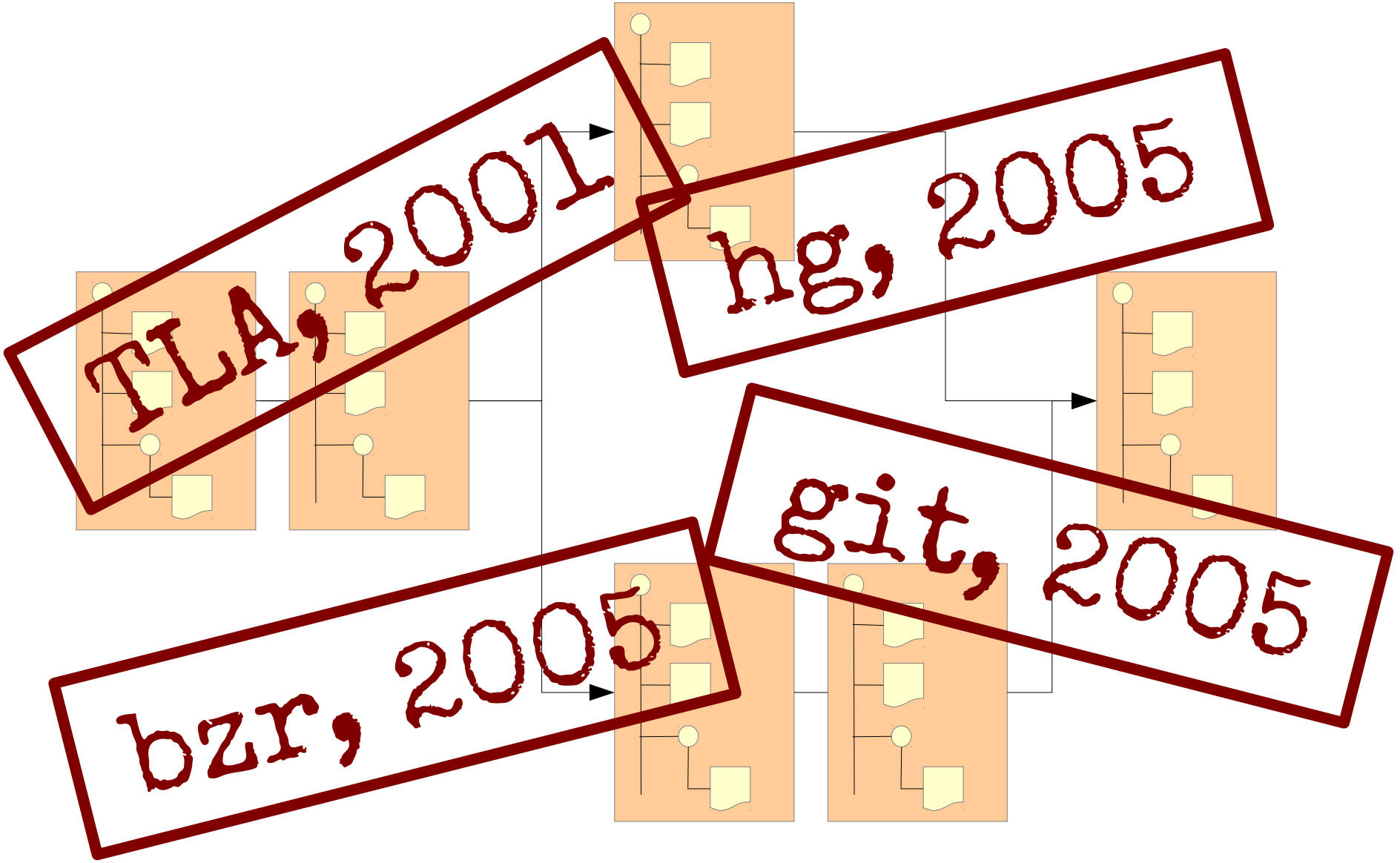
Soluzione: contenuto totale albero, non directory



Albero completo / content tracking



Albero completo / content tracking



Content tracking

+

relazioni tra snapshot

risolve tutto!

Content tracking risolve tutto!

Falso, purtroppo falso.

1. Non è possibile scaricare solo una sottodirectory.

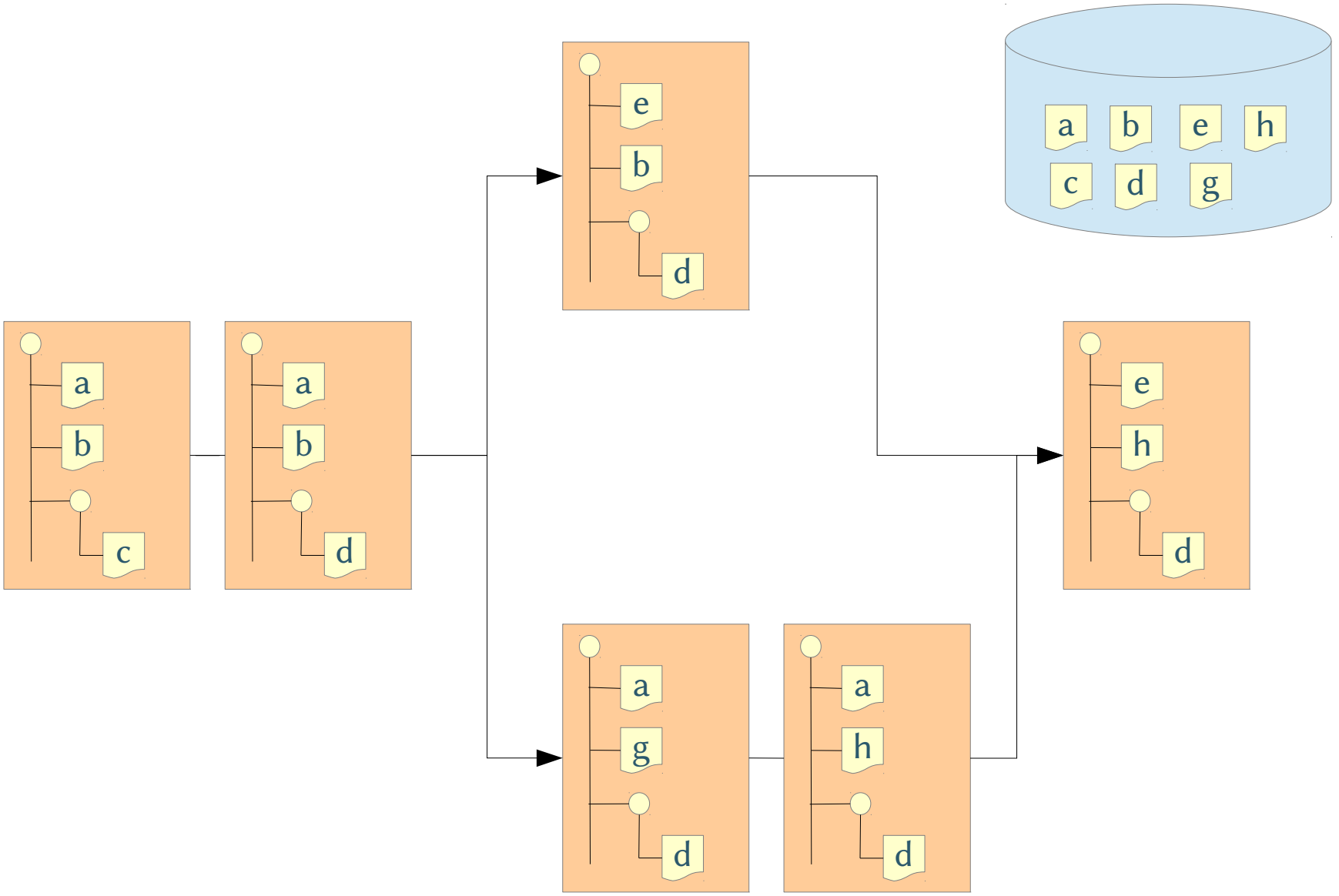
2. Ogni minimo cambiamento è (concettualemente) un cambiamento di tutto l'albero

- In git anche i cambiamenti dei metadati

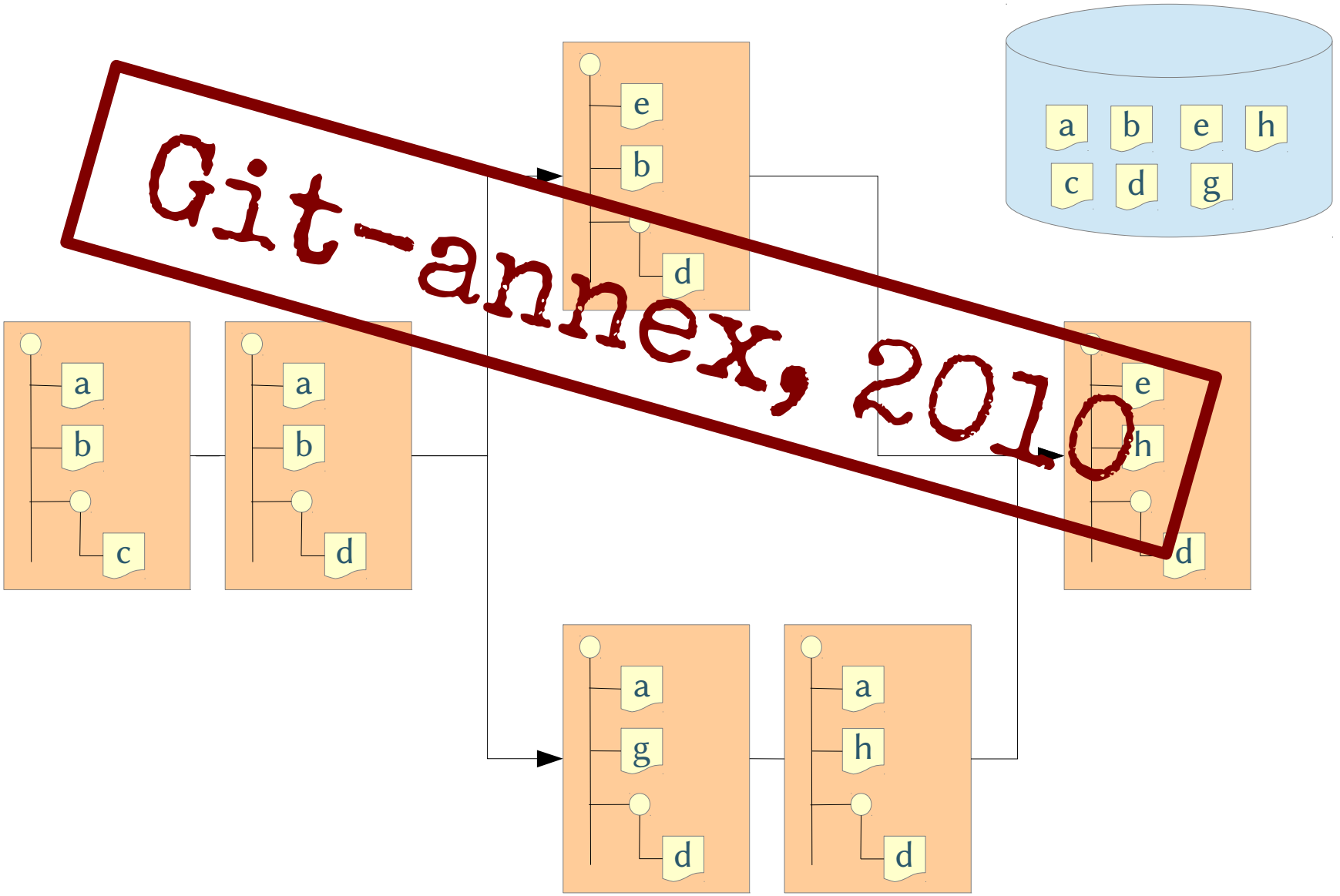
1 + 2 = difficile tracciare grandi file binari

All problems in computer science can be solved by another level of indirection

Due repo: contenuto e storia



Due repo: contenuto e storia



Riassumendo

- VMS/Files-11
- SCCS
- RCS
- CVS
- SVN
- SVK
- git & Co.
- git-annex

- Granularità
  - Linea ⟶ progetto
- Storia
  - Lineare ⟶ grafo
- Centralizzazione ⟶ decentralizzazione
- Codice sorgente ⟶ qualsiasi tipo di file

Public Pad

Pad Options | Import/Export | Saved revisions | Time Slider

100%

- On peut faire des rencontres
- ose des problèmes, point de vue du travail : licensiment
- Une fois qu'on publie une image elle ne nous appartient plus
- Mm aprés supression du compte les données sont sauvegarder
- FB permet de partager des photos et vidéos
- Les photos qui nous appartenaient qui sont affiché, ne nous appartiennent plus

On ne sait pas toujours avec qui on parle

Retrouver des amis qu'on a perdu de vu mais seulement ceux qui ont FB.

On peux partager plein de choses sur facebook

Arguments moraux :

- Les comptes sont facilement piratable
- Les réseaux sociaux nuisent aux relations interhumaines et isolent les individus !
- Les utilisateurs ont une vie social plus développés

Arguments psychologiques :

- Tous les gens peuvent voir nos photos
- Suicides , Harcelements .
- Facebook nuit aux relations humaines (isole les individus, interhumaines)

Fb peut être dangereux

Arguments politiques :

- Revendre des informations au gouvernement
- le business se base sur nos données.
-

Arguments économiques:

- On n'a rien a payer c'est gratuiiiiiiiiiit

Prof

- Anna et Sarah :)
- Ayse et Adeline :)
- Jordan et Laura
- Justine et Noémie
- Laura et Médina
- Lucas & maéva
- Marwa !
- quentin & Marianne
- steffen & Akop

Share this pad

November 23, 2012

Laura et Médina: FB permet de rester en contact avec des personnes qui vivent loin 11:04

Marwa ! 8): le business se base sur nos données. 11:05

Marwa ! 8): Harcelement 11:05

Justine et Noémie: :) 11:05

Marwa ! : Pose des problèmes, point de vue du travail : licensiment 11:06

quentin & Marianne : j'en parlerais a mon cheval 11:06

Ayse et Adeline :): MDR 11:06

Chat:

Non abbiamo parlato di

- Diff
  - È solo un'ottimizzazione per lo spazio?
    - Delta all'avanti / all'indietro
  - È necessaria per risolvere i conflitti automaticamente?
- Lock
  - Come posso segnalare che sto lavorando a qualcosa?
- Semantica
  - Merge di stringhe senza conflitti == programma funzionante?

Version control system, in teoria ed in pratica

Gioele Barabucci

Universität zu Köln