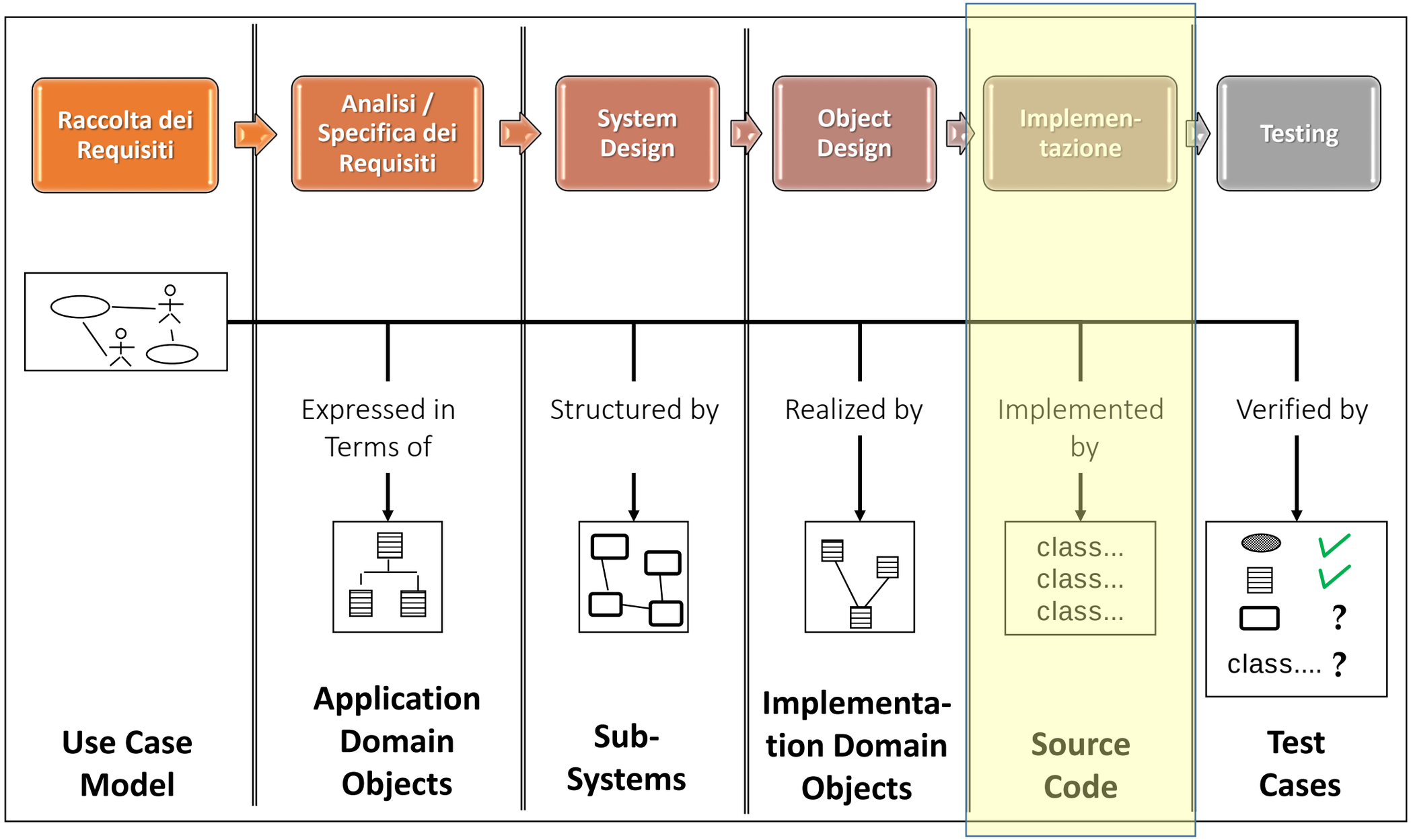
UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

Ingegneria del Software – Il Versioning

Prof. Sergio Di Martino

Ciclo di Vita del Software



Outline

- The problems with lots of code and lots of people
- Version control systems
  - what are they?
  - how are they used?
  - centralised versus distributed version control
  - Features of version control including branching
  - A demo of SVN

Dealing with Change

- How do you manage your coursework?
  - Modifying existing code
  - Backing up working code
  - Checking if an idea works (Do I use a Hashtable or a HashMap?)
  - Sharing code in group projects?

(Bad) Solutions

- Copying (Coursework_working.java, Coursework_tmp.java, Coursework_working_10112015.java)
- Copy & Paste code snippets
- Copy entire directories
- Emailing code to people

Open Source

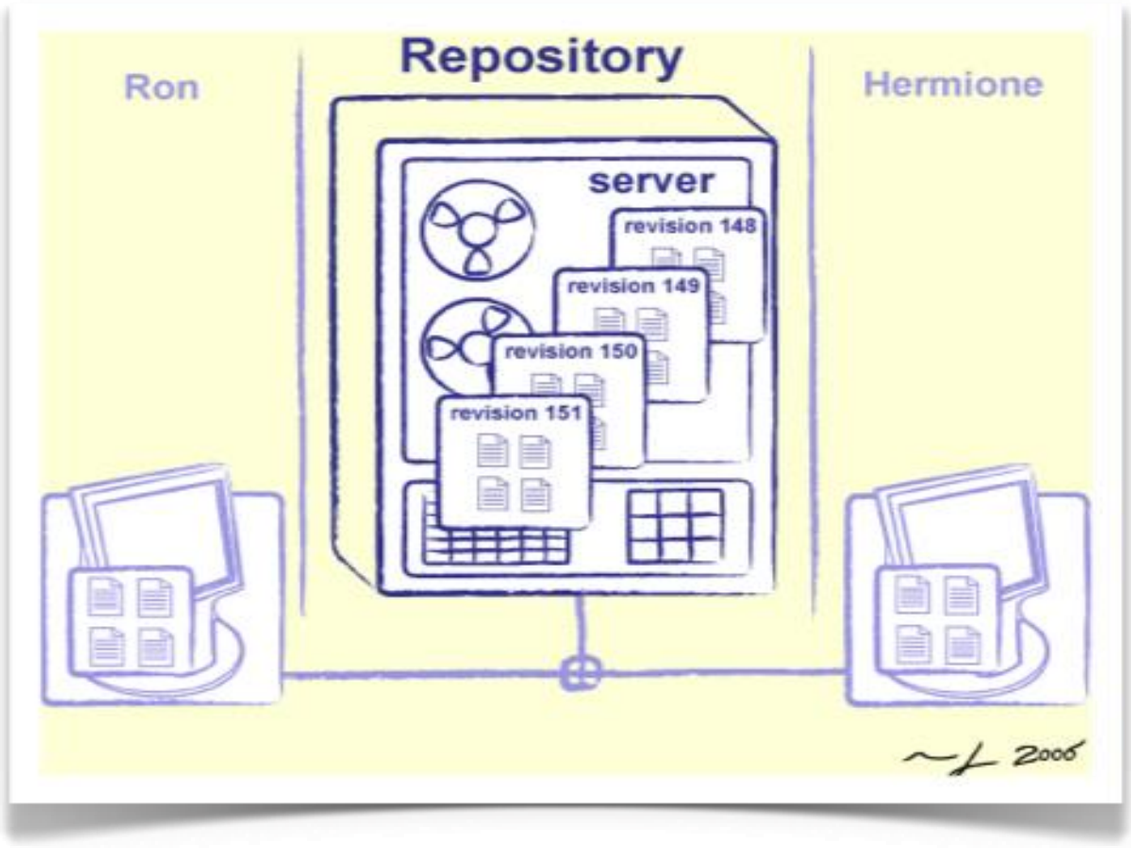
- You thought coursework was bad?
- Linux kernel has thousands of regular developers, millions of files.
- Developers spread over the globe across multiple time zones

Making a mess

- The Linux kernel runs on different processors (ARM, x86, MIPS). These can require significant differences in low level parts of the code base
- Many different modules
- Old versions are required for legacy systems
- Because it is open source, any one can download and suggest changes.
- How can we create a single kernel from all of this?

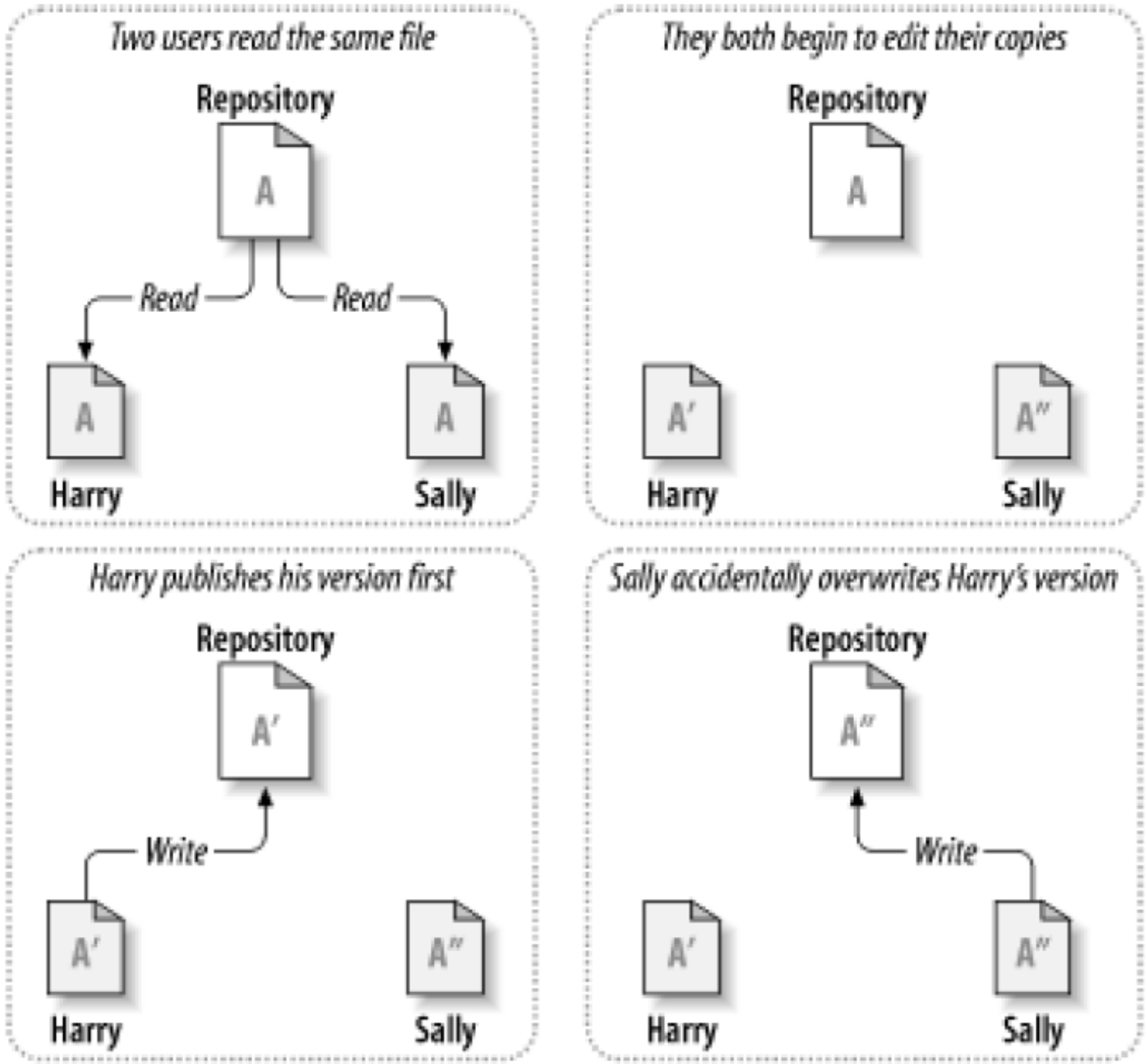
Control the process automatically

- Manage these things using a Version Control System (VCS)
- A version control system is a system which allows for the management of a code base.



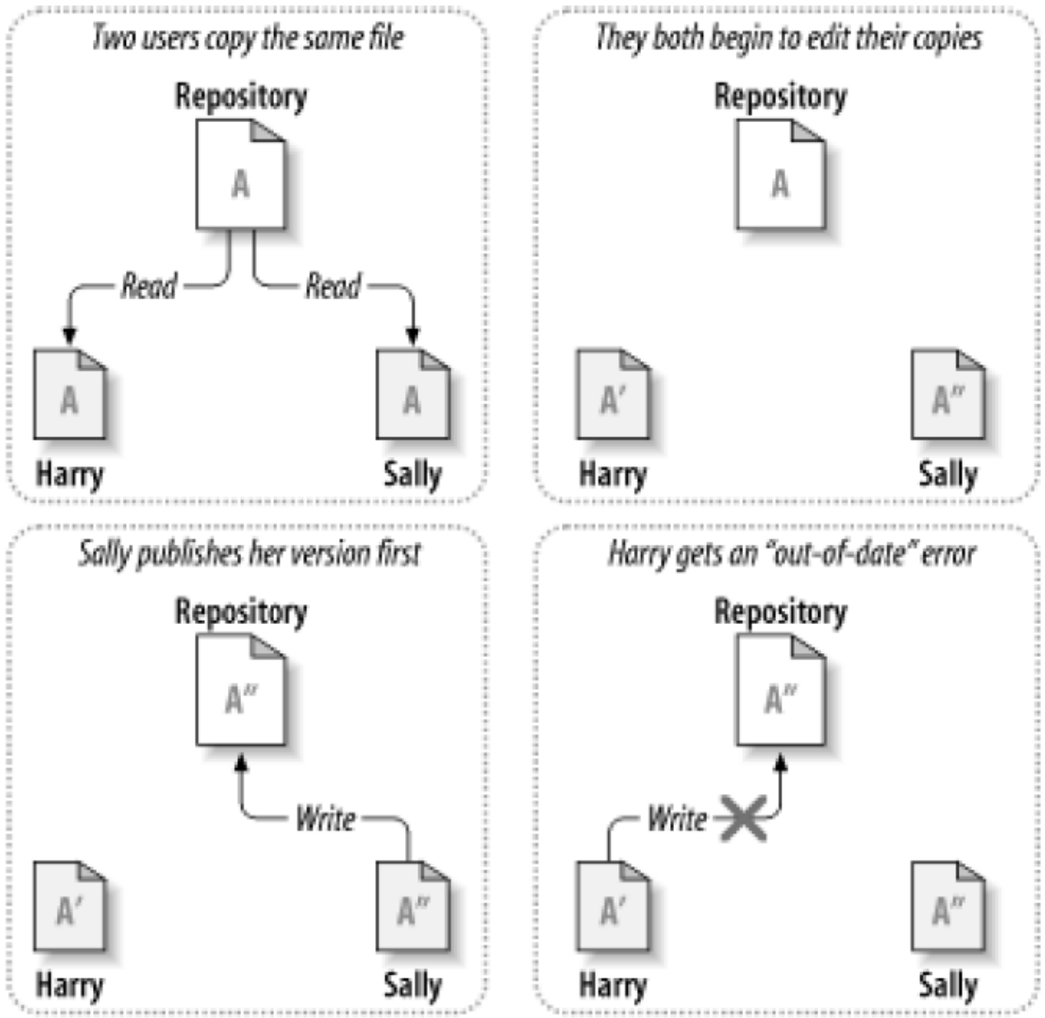
Problema: l'accesso concorrente

- In un file server "standard" le modifiche effettuate da Harry vengono sovrascritte da Sally



Copy-modify-merge

- Copy: ogni utente crea una copia locale del progetto, detta working copy
- Modifica la copia locale
- Merge: Eventuali conflitti vengono identificati prima della scrittura e risolti



Details of the process

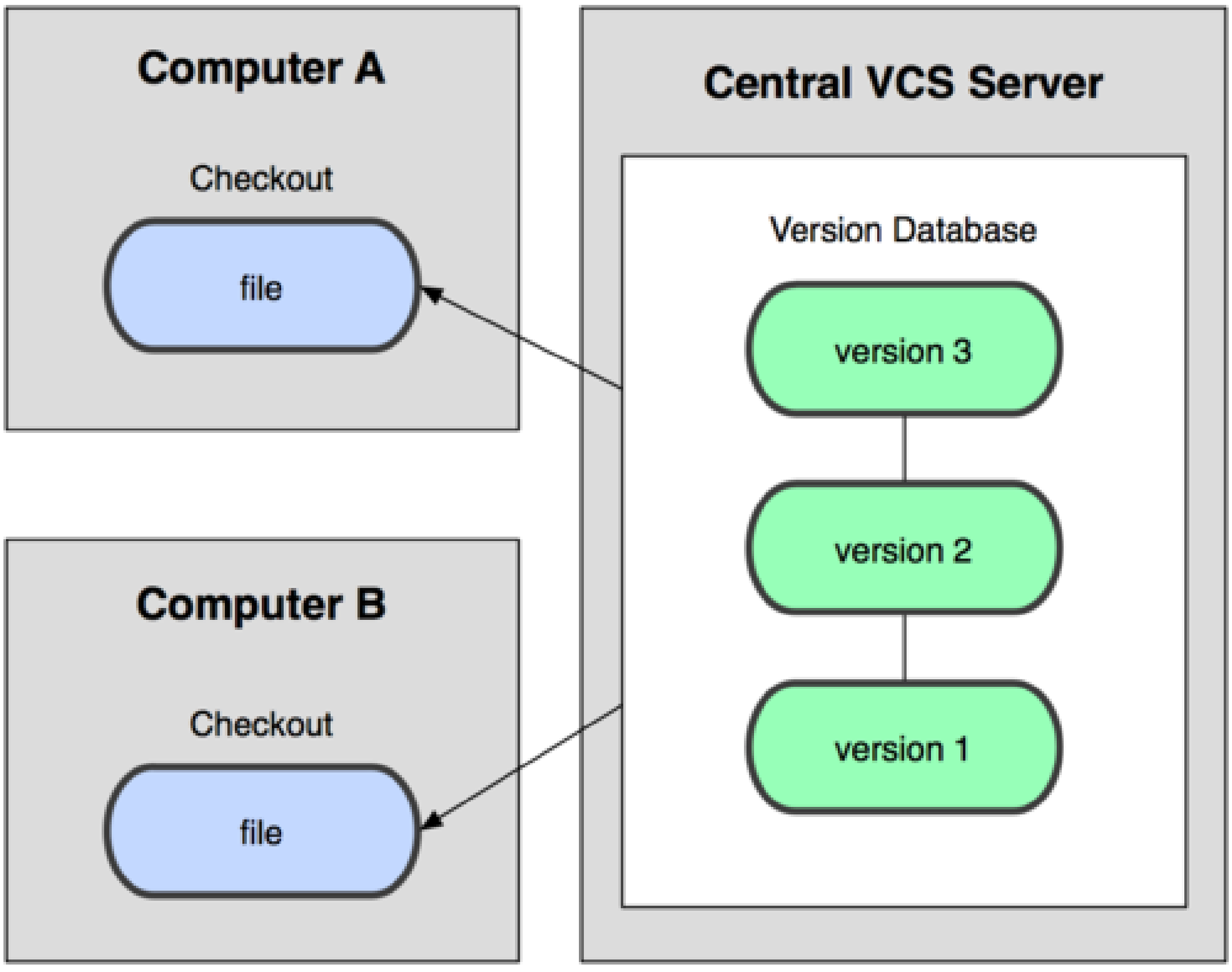
- Files are kept in a **repository**
- Repositories can be local or remote to the user
- The user edits a copy called the **working copy**
- Changes are **committed** to the repository when the user is finished making changes
- Other people can then access the repository to get the new code
- Can also be used to manage files when working across multiple computers

Tools for Software Configuration Management

- Permettono l'editing collaborativo, la condivisione ed il backup di informazioni
- Idea di base:
  - Un server tiene traccia di tutte le evoluzioni effettuate su interi sottoalberi di un File System
  - Un Client puo' scegliere quale versione dei Configuration Item scaricare, per lavorarci in locale. Default: viene fornita "l'ultima" versione del progetto.
  - Al termine delle modifiche sui CI, il client salva sul server le modifiche effettuate.
  - Il tool cerca di risolvere eventuali conflitti con le modifiche di altri client.
  - Il server memorizza TUTTE le modifiche effettuate ai file ed alla stessa struttura del filesystem

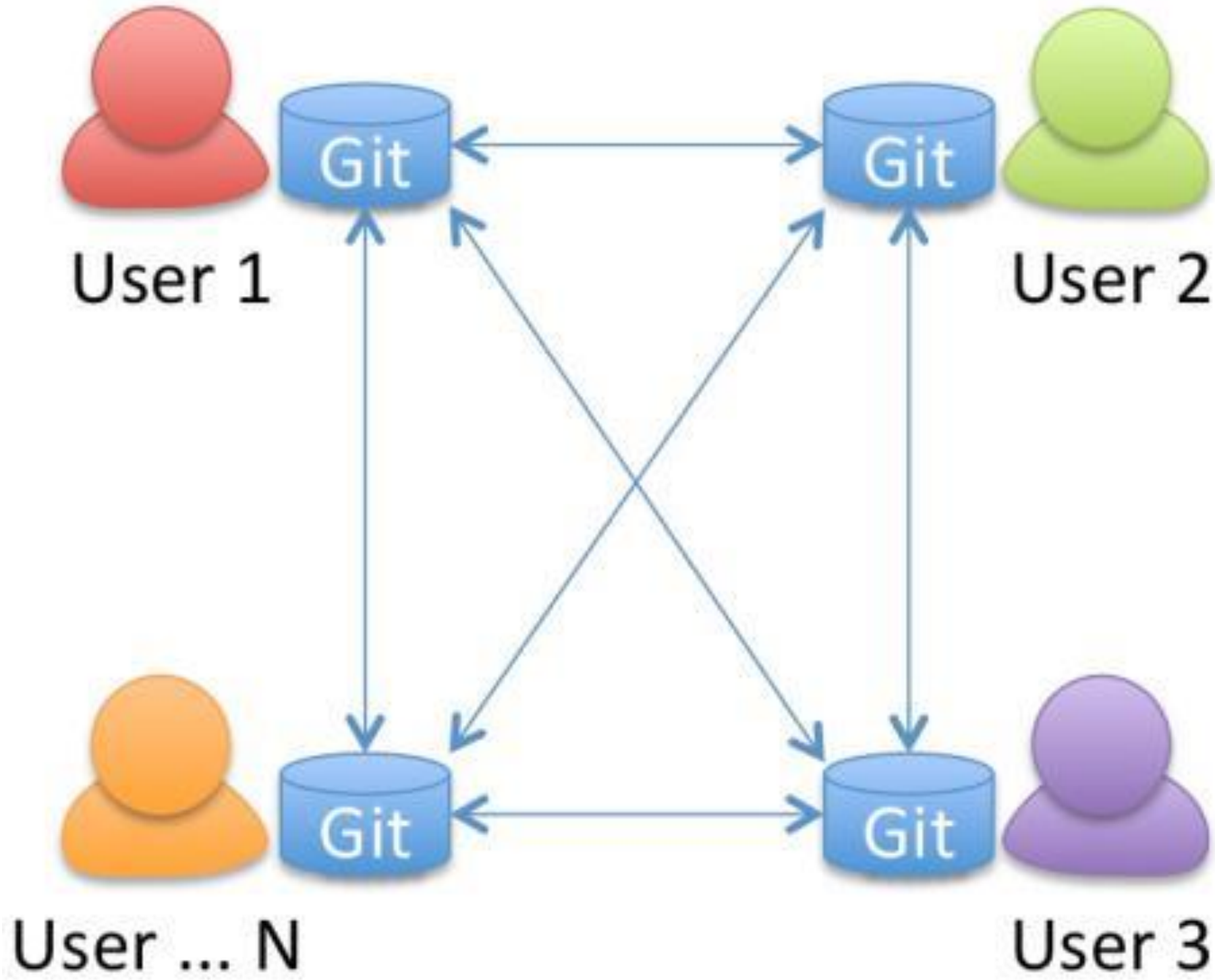
Centralised Version Control

- A single server holds the code base
- Clients access the server by means of check-in/check-outs
- Examples include CVS, Subversion, Visual Source Safe.
  - Advantages: Easier to maintain a single server.
  - Disadvantages: Single point of failure.


Computer A
Checkout
file
Computer B
Checkout
file
Central VCS Server
Version Database
version 3
version 2
version 1

Distributed Version Control

- Each client (essentially) holds a complete copy of the code base.
- Code is shared between clients by push/pulls
  - Advantages: Many operations cheaper. No single point of failure
  - Disadvantages: A bit more complicated!


Git
Git
User 1
User 2
Git
Git
User ... N
User 3

More Uses of Version Control

- Version control is not just useful for collaborative working.
- It is essential for quality source code development
- Often want to undo changes to a file
  - start work, realize it's the wrong approach, want to get back to starting point
  - like "undo" in an editor…
  - keep the whole history of every file and a changelog
- Also want to be able to see who changed what, when
  - The best way to find out how something works is often to ask the person who wrote it

Essential features

- Check-in and check-out of items to repository
- Creation of baselines (labels/tags)
  - Version 1.0 released!
- Control and manipulation of branching
  - management of multiple versions
- Overview of version history

SubVersioN

Subversion

- E’ un sistema di versioning centralizzato di Apache
  - Accessibile concorrentemente da piu’ client attraverso 5 modalita’.
  - Memorizza le informazioni in un albero di filesystem
  - Memorizza automaticamente TUTTE le modifiche effettuate ai file ed alla stessa struttura del filesystem
  - Utilizza come elemento di controllo il concetto di “repository”, i.e., un insieme di file (“di controllo”) che tengono traccia delle diverse versioni del progetto.
- Sostanzialmente… un file server che offre servizi più avanzati.

Working copies

- Una working copy e’ una directory sul sistema locale nella quale vengono copiati I file di una directory del repository
- All’interno della working copy l’utente puo’:
  - Creare sottodirectory
  - Modificare file
  - Aggiungere/rimuovere file.
- Le modifiche vengono riportate sul server solo quando l’utente lo indica esplicitamente

Revision

- SVN indica con il termine *revision* lo *stato* del filesystem sul server.
- Le revision sono numerate con un intero che viene incrementato automaticamente dal server ad ogni aggiornamento.
- In subversion il revision number corrisponde all'intero albero e NON al singolo file.
  - Un file puo' avere lo stesso contenuto in due revision diverse.
  - File diversi della stessa revision possono avere revision number diversi

Subversion: Checkout

- Scarica dal server il contenuto di una directory (o di un file).
  - Utilizzata per “creare” la working copy.
  - La directory locale contiene una sottodirectory “di servizio” utilizzata dal client per tenere traccia dello stato dei file
  - Ogni utente puo’ avere piu’ working copies dello stesso progetto memorizzate localmente… attenzione alla confusione!

Subversion: Commit

- Commit: indica al client di scrivere sul server, le modifiche effettuate sulla working copy
  - Il commit e' eseguito in modo *atomico*
  - Per ogni commit e' possibile indicare un commento.
  - Importante: E' fondamentale specificare sempre nel commento le modifiche effettuate al progetto. Cio' rende possibile distinguere diverse versioni del progetto.
- L'esecuzione di un commit corrisponde alla creazione di una nuova revision.

Subversion: Update

- Update: Aggiorna il contenuto della working copy locale.
  - Il contenuto della working copy viene “sincronizzato” con il contenuto del server.
  - L’idea di fondo e’ “aggiornare la working copy utilizzando le versioni piu’ recenti dei file”
    - “Scarico” il file se la versione nel repository e’ piu’ recente di quella locale
- Attenzione:
  - Update e commit sono operazioni DISGIUNTE e “monodirezionali”.
  - Update. Aggiorna SOLO la working copy.
  - Commit. Aggiorna SOLO il repository. In caso di conflitto fallisce, I.e., NESSUN file viene modificato.

Gestione file

- Aggiungere un file (o creare una directory):
  - E’ necessario “creare” il file nella working copy
  - Eseguire il comando “add” per aggiungere il nuovo file alla directory di servizio
  - Eseguire il comando commit per aggiornare il repository.
- Eliminare un file:
  - Eseguire il comando “delete”
  - Rimuovere il file dalla working copy
  - Eseguire il commit
- Copiare un file.
  - Eseguire il comando “copy” seguito da “commit”
- Rinominare unfile
  - Eseguire il comando “move” seguito da “commit”
- Creare una directory
  - Eseguire il comando “mkdir” seguito da commit.

Verifica modifiche: status

- Subversion consente di verificare le modifiche “schedulate per l’esecuzione” prima del commit.
- Status: visualizza le modifiche schedulate per ogni file. Puo’ ritornare:
  - ?: File NON presente nella directory di servizio, e.g., non e’ stato eseguito “add”.
  - A: Nuovo file, per il quale e’ stato eseguito “add” (e presumibilmente non presente nel repository)
  - D: File “Schedulato per la cancellazione”
  - M: File modificato (la versione della Working copy e’ stata modificata successivamente al checkout/update)
  - C: Conflitto. IL file e’ nello stato “Modificato e non piu’ valido”… e’ necessario risolvere il conflitto.

Verifica modifiche: diff

- Il comando diff visualizza le “modifiche effettuate al file”.
  - Per default, differenza tra la versione corrente del file e la versione dell’ultima checkout o update.
  - Oppure differenza tra due revision di file nel repository o la working copy ed un file nel repository
- L’output di diff e’ in “unified diff format”
  - Le righe “aggiunte”/”eliminate” iniziano per “+”/”-”
  - Tutte le righe dei file “schedulati per l’aggiunta/cancellazione” iniziano per “+”/”-”
- E’ possibile utilizzare l’output del comando diff (di subversion) come input di patch

Annullare le modifiche: revert

- Per annullare “TUTTE” le modifiche effettuate ad un file, e’ possibile utilizzare il comando revert.
- L’annullamento riporta il file (nella working copy) alla versione ottenuta dall’ultima operazione di checkout o update.

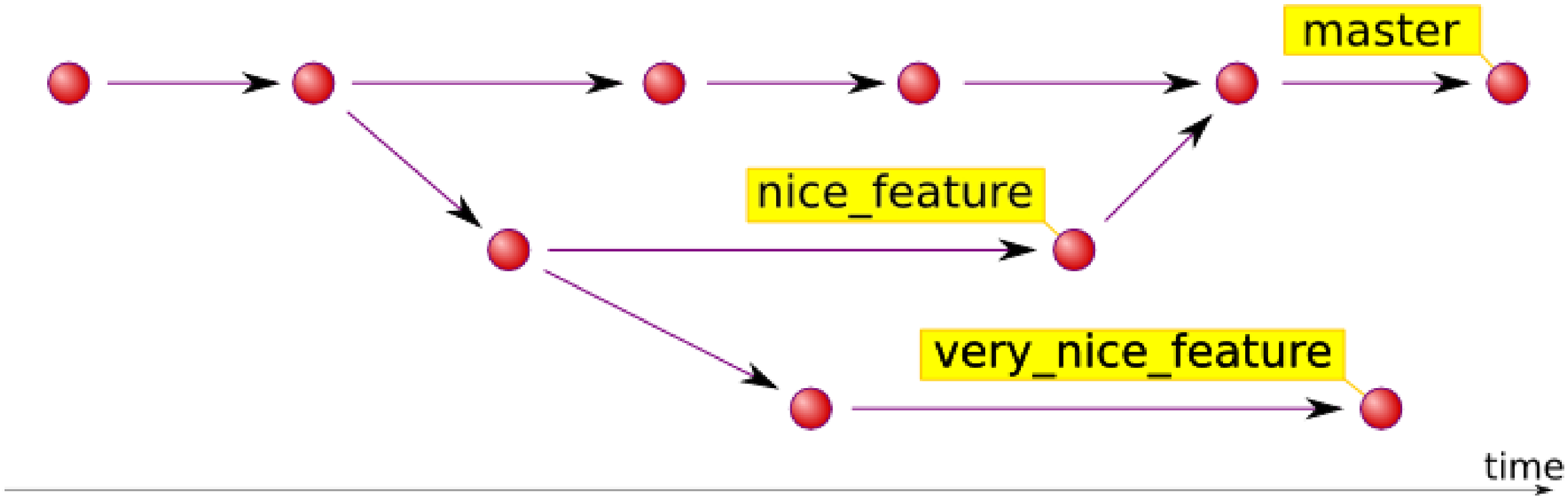
Ciclo di Lavoro “tipico”

- Allineamento della working copy
  - Update
- Esecuzione modifiche
  - Add, delete, copy, move, mkdir (e File editing..)
- Verifica modifiche
  - Status e diff
- In caso… annullamento modifiche
  - Revert
- Risoluzione conflitti
  - Update e resove
- Aggiornamento repository
  - Commit

Branching

- Branches allows multiple copies of the code base within a single repository.
  - Different customers have different requirements
    - Customer A wants features A,B, C
    - Customer B wants features A & C but not B because his computer is old and it slows down too much.
    - Customer C wants only feature A due to costs
  - Each customer has their own branch.
- Different versions can easily be maintained

Esempio di Branch



Perche’ creare un branch

- Lo sviluppo di una nuova versione di un servizio potrebbe essere sviluppata “localmente in autonomia”, senza eseguire il commit
- Pessima idea
  - Forza la gestione del backup locale
  - Non e’ possibile avere “feedback” sul lavoro svolto
  - Se lo sviluppo prende molto tempo, e’ possibile che il nuovo servizio non si integri perfettamente con i vecchi (che, nel frattempo sono evoluti)

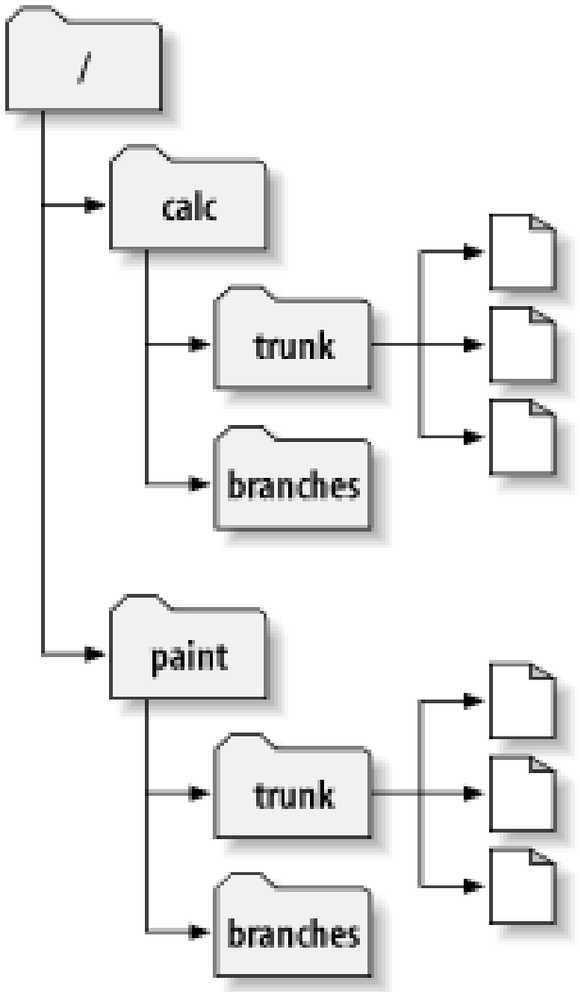
Creare un branch

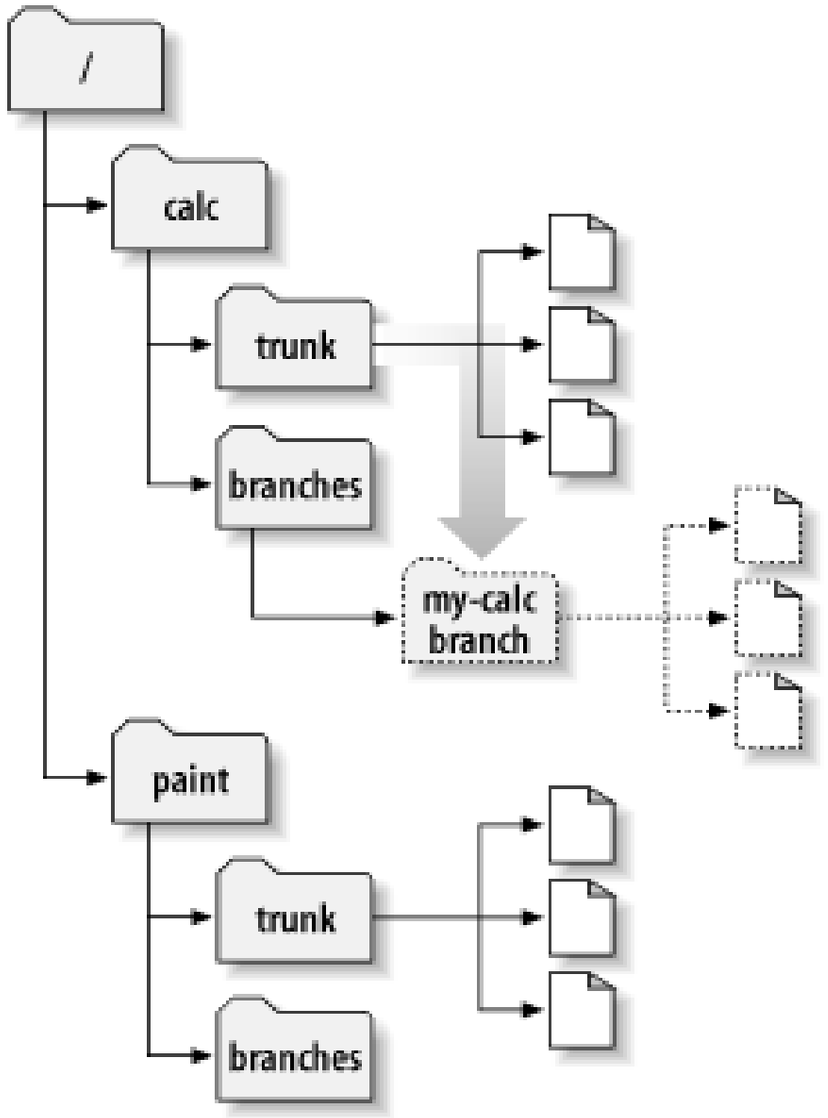
- Corrisponde alla creazione della copia di un intero progetto.
- La copia puo’ essere eseguita:
  - All’interno della working copy locale attraverso svn copy
    - Sconsigliabile. Prende tempo lineare.
  - Direttamente sul repository
    - Prende tempo costante.
- La copia di una directory NON duplica I file sul server… ma crea “hard link”.

Creare un branch (2)

- Subversion NON gestisce i “branch” in se.
  - Il nome della directory che conterra’ il branch, cosi’ come la sua posizione nel FS deve essere stabilita attraverso una politica locale
  - Subversion gestisce le copie di file/directory
  - Il log mantiene informazioni sullo stato del branch corrente.
- Dopo la creazione del branch, le due linee di sviluppo sono “indipendenti”
  - L’esecuzione del commit su una NON modifica le altre

Esempio


/
calc
trunk
branches
paint
trunk
branches


/
calc
trunk
branches
my-calc
branch
paint
trunk
branches

Esempio

- r341: Branch
- r342: modifica a my-calc-branch/button.c
- r343: modifica a my-calc-branch/integer.c
- r344: modifica a trunk/integer.c