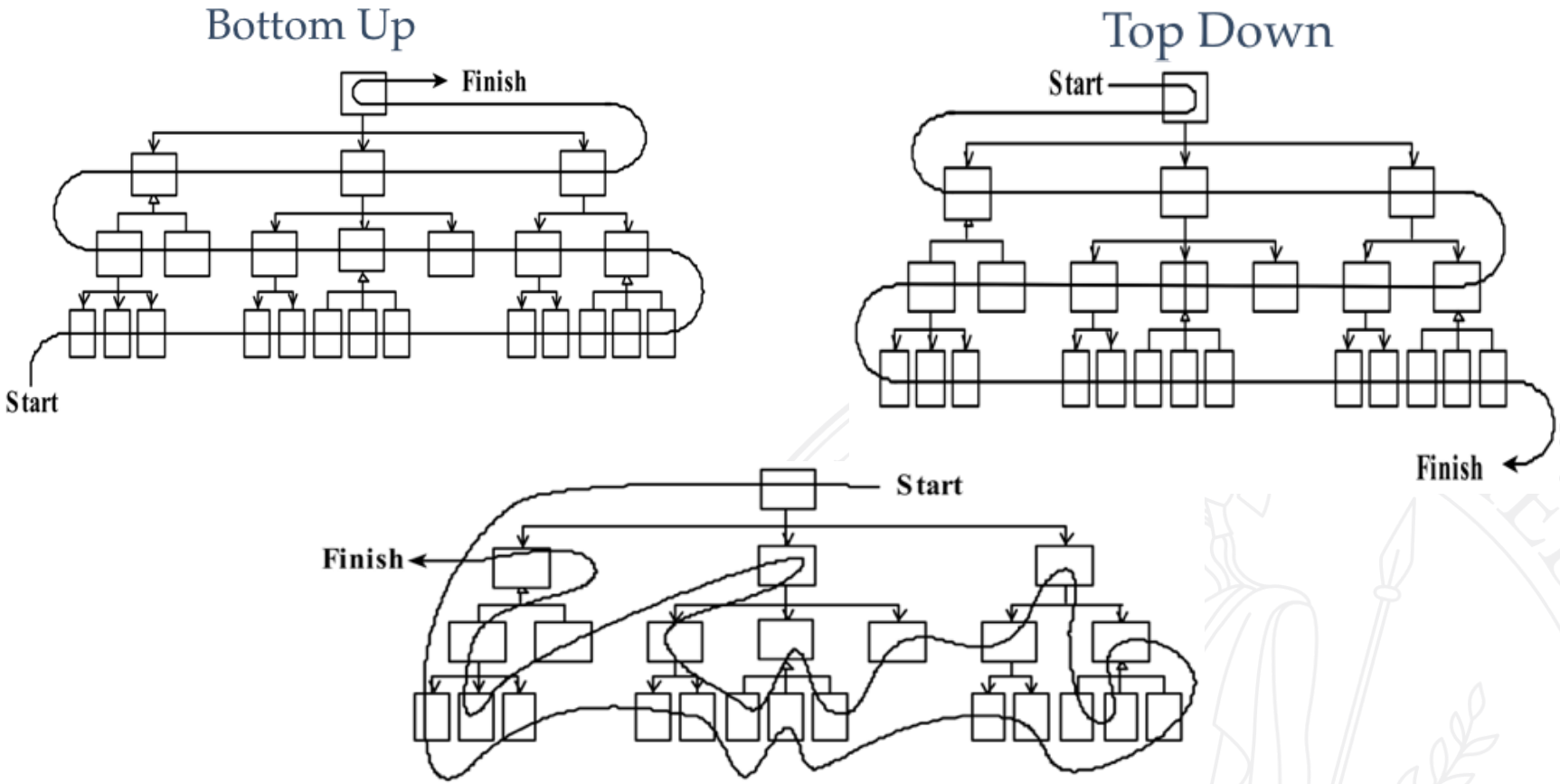
UNIVERSITÀ DEGLI STUDI
DI MILANO

7. Intro Continuous Integration

Integration



Unit vs Integration testing

https://youtu.be/4XpnKHJAok8?si=bLgVEhnDOCMYSWvY&t=3025

Continuous Integration

Definizione:

> Allineamento frequente (es. "molte volte al giorno") dagli ambienti di lavoro degli sviluppatori verso l'ambiente condiviso (mainline)

- Non è più una fase… viene affogata nello sviluppo normale…
  - finita una parte (una feature, una patch, …) si integra
- l'integrazione diventa un NONEVENT

> "One of the things I want to say about performance is that a lot of people seem to think that performance is about doing the same thing, just doing it faster, and that is not true. That is not what performance is all about.
> If you can do something really fast, really well, people will start using it differently. One of the things I wanted to make sure is that merges go really really quickly because I want people to merge often and merge early, because as it turns out it becomes easier to merge. If you merge every day, suddenly you never get to the point where you have huge conflicts that are hard to resolve.
>
> *Linus Torvald at a Google Tech Talk in cui presenta Git*

http://www.martinfowler.com/articles/continuousIntegration.html

Articolo di Martin Fowler

- prima versione del 2000 prima versione
- revisione del 2006 su webArchive
- a gennaio 2024 pubblica nuova versione

Modifiche non banali

- 2000: non parla mai di *branch*
- 2006: usa la parola *branch* solo 4 volte

> One of the features of version control systems is that they allow you to create **multiple branches**, to handle different streams of development. This is a useful, nay essential, feature - but it's **frequently overused and gets people into trouble**. Keep your use of branches to a minimum.

- 2024: usa la parola branch più di 50 volte

> A developer may have been working for several days on a new feature, regularly pulling changes from a common main branch into her feature branch.

Maintain a Single Source Repository

Put everything in a version controlled mainline

I should be able to walk up with a laptop loaded with only an operating system, and by using the repository, obtain everything I need to build and run the product.

Rilevava importanza di mantenere in una sola posizione tutti i sorgenti (source, scripts) avere una “mainline” cioè una chiara indicazione di quale è la versione di riferimento

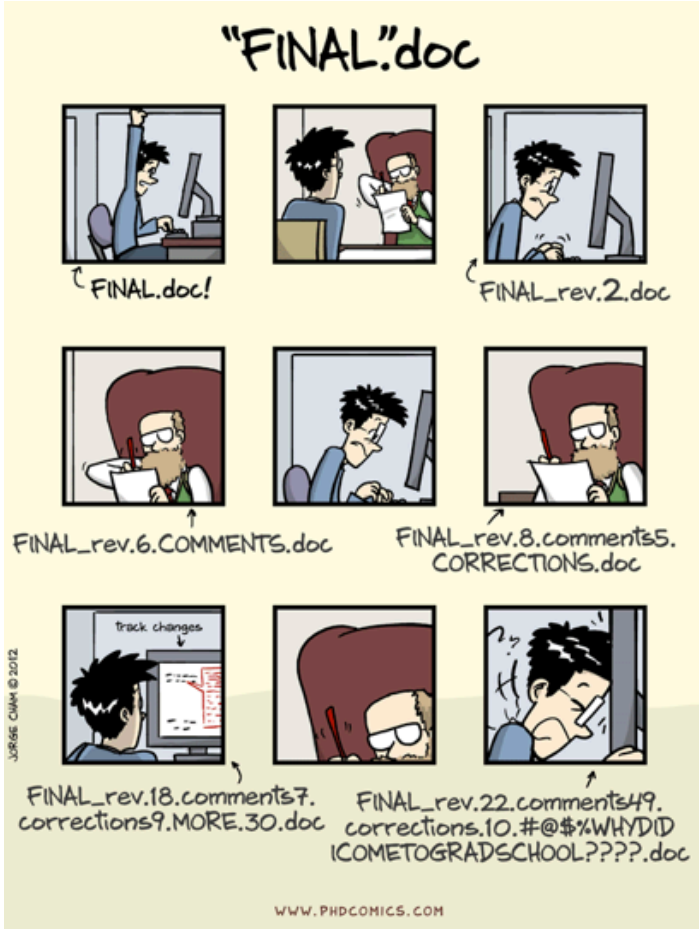
o meglio chiara visione di quale è il codice di **qualunque** “versione”

Ora ha aggiunto un *"put everything"*

tutto il necessario per ricostruire il software su una macchina "nuda"

http://www.phdcomics.com/comics/archive.php?comicid=1531

Single Source Repository



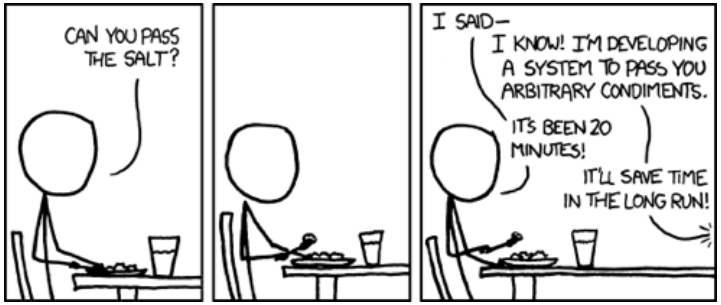
Automate the Build

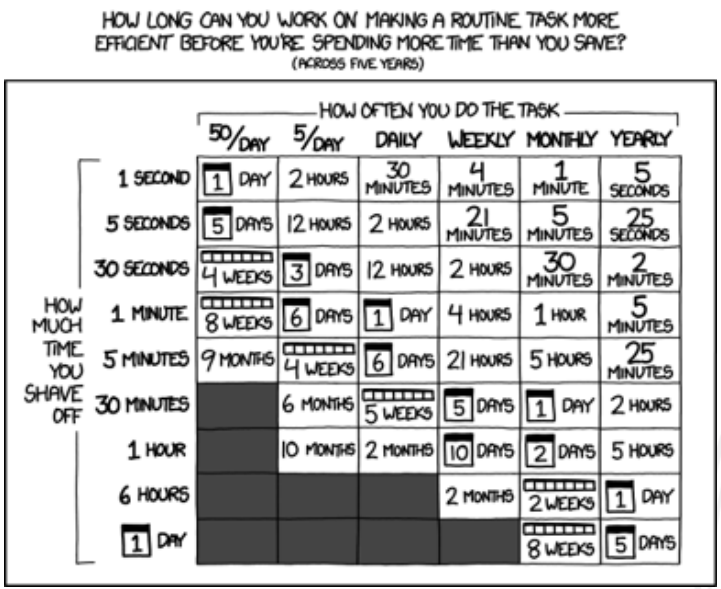
Computers are designed to perform simple, repetitive tasks. As soon as you have humans doing repetitive tasks on behalf of computers, all the computers get together late at night and laugh at you.

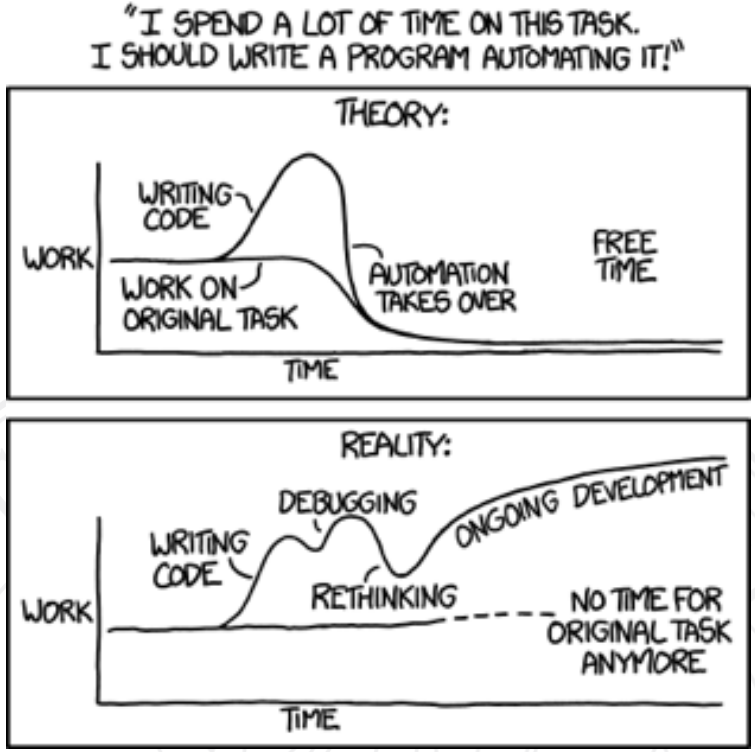
- il ricostruire il sistema deve essere fatto con un singolo comando
- build ripetibile deterministica
- non solo compilazione
- diversi target (build, check, run, clean, distribuite,…)
- gestione dipendenze e generazione solo del necessario

https://xkcd.com/974 https://xkcd.com/1205 https://xkcd.com/1319

Automate the build







Make Your Build Self-Testing

Self-testing code is so important to Continuous Integration that it is a necessary prerequisite. Often the biggest barrier to implementing Continuous Integration is insufficient skill at testing.

- i tool di versioning vedremo che faranno primo check di conflitto
- con compilazione (build) c'è controllo ulteriore
- con esecuzione di suite di test facciamo ulteriore controllo
  - test di unità, di integrazione, di sistema, di non regressione
- Scrivere codice auto-testante influisce sul flusso di lavoro del programmatore
- TDD è uno dei modi migliori per scrivere codice auto-testante (anche se non è essenziale)

Everyone *pushes* commits to the mainline every day

If everyone pushes to the mainline frequently, developers quickly find out if there's a conflict between two developers. The key to fixing problems quickly is finding them quickly.

No code sits unintegrated for more than a couple of hours. – Kent Beck

- Ingegneria del software si occupa di "comunicazione"
- L'integrazione riguarda principalmente la comunicazione
- L'integrazione permette agli sviluppatori di comunicare ad altri sviluppatori i cambiamenti che hanno fatto

Every Commit Should Build the Mainline on an Integration Machine

Every Push to Mainline Should Trigger a Build

Sebbene possa essere fatto manualmente, avere preparato un ambiente di build automation non legato a una GUI permette di automatizzare l'operazione.

- Perché necessario visto che sicuramente avremo eseguito i test prima di fare il commit?
  - pigrizia del programmatore
  - mancata aggiunta di un file al repository
  - diversità del contesto di esecuzione (versioni compilatori e librerie)

Fix Broken Builds Immediately

Usually the best way to fix the build is to revert the faulty commit from the mainline, allowing the rest of the team to continue working

- tutto il team ha problemi ad andare avanti se non compila
- nessuno può integrare la propria parte
- diversi approcci:
  - correggere (se è una cosa banale)
  - revert (e correggere con più calma e meno stress)
  - prevedere un workflow con una fase di pending

Keep the Build Fast

Probably the most crucial step is to start working on setting up a Deployment Pipeline.

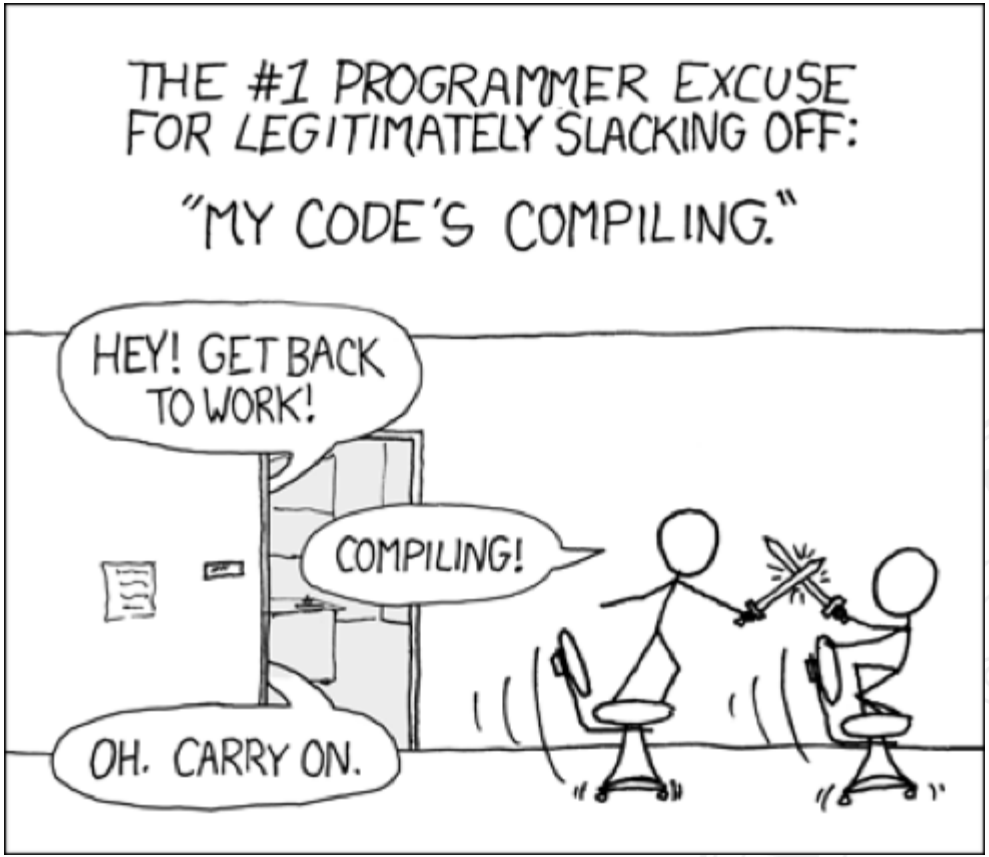
Necessario per l'*immediatamente* del punto precedente

Staged Build:

1. **Commit Stage**: Build veloce + test critici
2. **Acceptance Stage**: Test più lunghi e complessi
3. **Production Stage**: Deploy in ambiente simile a produzione

Keep the Build Fast

https://xkcd.com/303/



Hide Work-in-Progress

How to deal with latent code: code that's part of an unfinished feature that's present in a live release.

- Non sempre è possibile rilasciare una feature completa ogni giorno
- *Keystone interface:* rilascio codice non raggiungibile dalla UI, ma che viene comunque testato (unità e integrazione)
- *Feature Flag:* esecuzione resa possible in base a variabili di configurazione