Modern Release Engineering in a Nutshell Why Researchers should Care

+ How the release process impacts your software analytics

Presented by: ZHENYU BAI (Tim)

Authors: Bram Adams and Shane McIntosh

Polytechnique Montreal and McGill University

Few slides were taken from original presentation.

UNIVERSITY OF WATERLOO

Outline

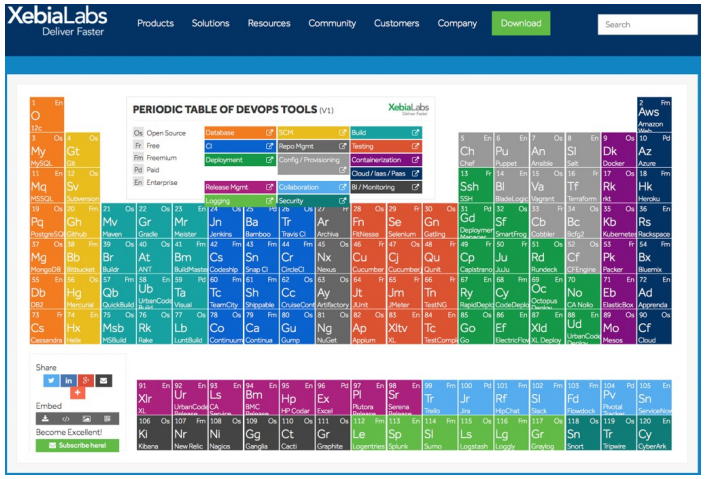
- Background of Release Engineering
- Six major Phases of Release Engineering Pipeline
  - *Integration: Branching and Merging*
  - *Continuous Integration: Building and Testing*
  - *Build System*
  - *Infrastructure-as-Code*
  - *Deployment*
  - *Release*
- A Release Engineering Checklist
- Summary

UNIVERSITY OF WATERLOO

Release of a Software

UNIVERSITY OF WATERLOO

Release of a Software



UNIVERSITY OF WATERLOO

openstack
6 months

6 weeks

2 weeks
(mobile)

twice/day
(web)

NETFLIX
daily

UNIVERSITY OF WATERLOO

7

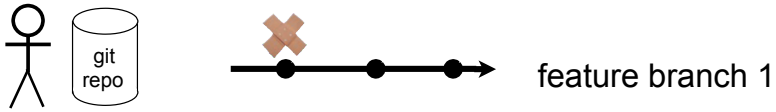
Release engineering innovation **led by industry**!

UNIVERSITY OF WATERLOO

Integration: Branching and Merging

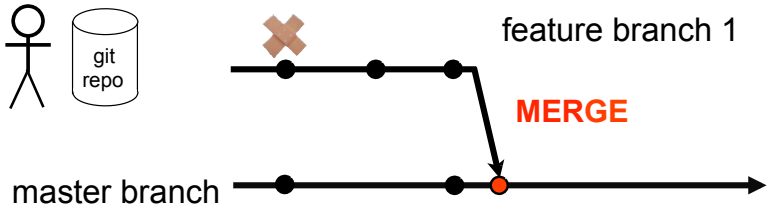
One of the major concerns for software organization is to bring different developers' codes to the project's master branch.

How to achieve this as fast as possible without compromising code quality?



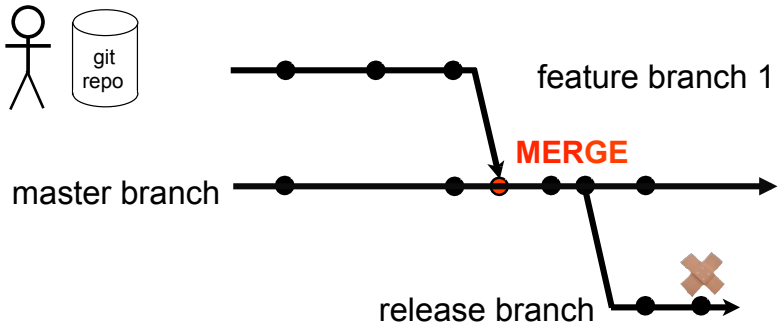
UNIVERSITY OF WATERLOO

Integration: Branching and Merging



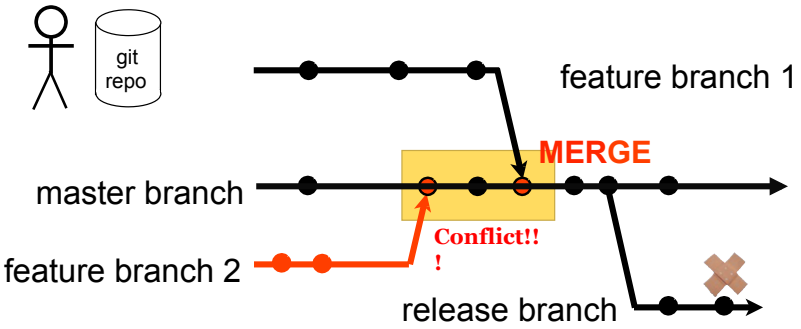
UNIVERSITY OF WATERLOO

Integration: Branching and Merging

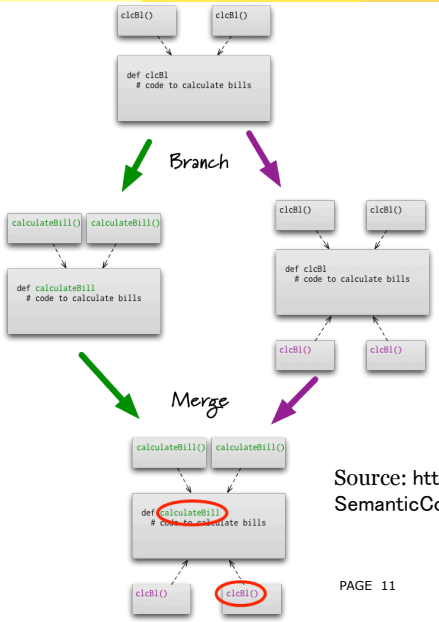


UNIVERSITY OF WATERLOO

Integration: Branching and Merging



UNIVERSITY OF WATERLOO



Source: https://martinfowler.com/bliki/SemanticConflict.html

UNIVERSITY OF WATERLOO

Integration: Branching and Merging

The best way to mitigate conflict risk is to **Merge Often**.

Another solution is “Trunk-based development”
- Allow developer commit to master branch directly after review.

However, it kills the safe isolation. Solution?

Feature toggles:
- It puts source code of incomplete feature inside conditional blocks

UNIVERSITY OF WATERLOO

Integration: Branching and Merging

Open Questions:

Predict merge conflict or suggest merge order to minimize conflict.

Mining VCS repository to build models pinpoint bottle neck in this phase.

Continuous Integration: Building and Testing

What is Continuous Integration?

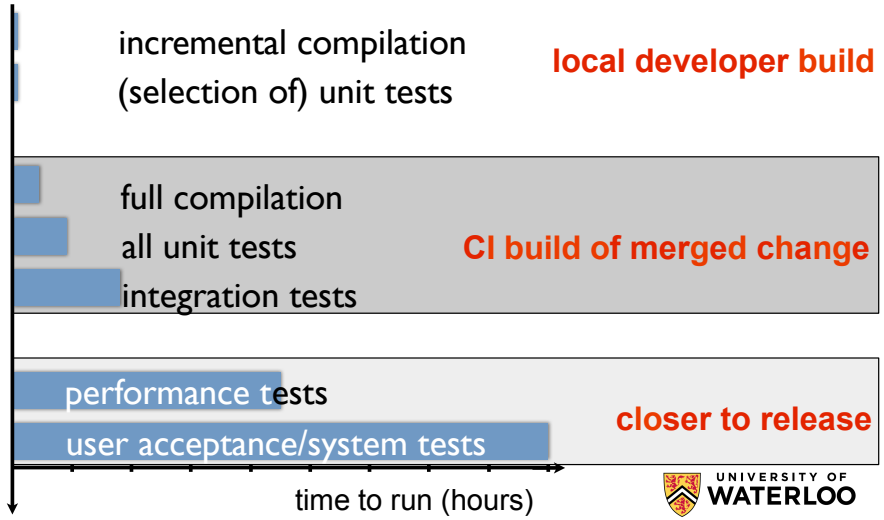
It is the practice to pull new commits or merges and build them on some dedicated server.

Usually a small set of test will be executed after the build.

Recent empirical studies suggest that the rapid feedback loop provided by CI has a positive effect on team productivity

There is one problem…

Continuous Integration: Building and Testing



Continuous Integration: Building and Testing

Another strategy is to predict a code change will break the build.
- There already exist some prediction models.

Open challenge is to perform prediction across platforms.

Regarding to energy consumption
- Network, disk I/O

One more research challenge is about security.
- Malicious script can inject a payload into CI's build result.

Build System

The build system is the set of build specification files used by the CI infrastructure (and developers) to generate project deliverables.

There are hundreds of tools built for different languages
- e.g Ant, Makefile, maven

The build correctness has been a challenge for a long time.

Qualitative analysis of build system evolution and maintenance is largely missing.

Build System

Additional challenge:

Identification and resolution of build bugs.
- i.e source code or build specification changes cause breakage

Prediction models can be built.

Infrastructure-as-Code

Infrastructure-as-Code is used to automatically generate environment based on a specification (e.g Puppet, Chef and Salt).

```
include_recipe "postgresql::server"
include_recipe "postgresql::client"

# Make postgresql_database resource available
include_recipe "database::postgresql"

# Create database for Rails app
db = node["practicingruby"]["database"]
postgresql_database db["name"] do
  connection(
    :host     =>
    :port     =>
    :username =>
    :password =>
```


Infrastructure-as-Code

Container vs Virtual Machine
- Containers are lightweight
- Saving disk space and memory, reduce run time overhead

Potential Research opportunities:
- Similar to build system, qualitative study is missing
- Find best practice and design pattern
- How developers address different infrastructure needs

Deployment

Deployment is the phase in which the tested deliverables for the upcoming release are staged in preparation for release.

- e.g pushing deliverables to web server
- submit app to app store

Deployment Approaches:

"Blue/Green Deployment"

- Deploy new version on a copy of production environment

"Canary Deployment"

- Deploy new version on a subset of production environment

"A/B Test"

- Deploy two versions and make comparison

Deployment

Challenges:

Empirical evidence of different deployment approaches.

- trail-and-error does not work for small company

Mobile App's inversion of deployment control

- better tools for quality assurance(defect prediction)

Release

To assure software quality, company typically make intermediate alpha, beta releases.

As mentioned in deployment section, even final release may only reach partial user groups at a time.

The challenge in this phase is to determine which code change is perfect.

Release

For web App:

Based on release logs, crash reports and user reviews, should the team roll a version back or forward?

For desktop and mobile app:

The additional challenge is to cover multiple platform.

The Checklist

1. Not All Releases are Equal

Understand the release schedule of the software is very important!
- time based vs feature based
- minor, major, patch release
- cycle time

2. Branches

- Most software projects have multiple concurrent branches.
- For some study, one should ignore merge commits and select correct branch to study.

The Checklist

3. Choose before you Build

Not all codes compile for a release.

Feature toggle even enable select features at run time.

Summary

Some background about release engineering.

Six phases of release engineering.

A checklist for those who want to do research in release engineering.

THANKS FOR YOUR TIME!