Week 3 – Collaborative Software Development

CSC 317 / INF 307 – SOFTWARE ENGINEERING

2022/2023 Academic Year

Semester 1

Recommended Reading

Mandatory reading material (*PDFs will be made available*)

- "Modern Code Review: A Case Study at Google" by Sadowski et. al (2018)
- "Version Control Systems" by Spinellis (2005)
- https://marklodato.github.io/visual-git-guide/index-en.html
- "Continuous integration" by Fowler, M. and Foemmel, M. (2006)

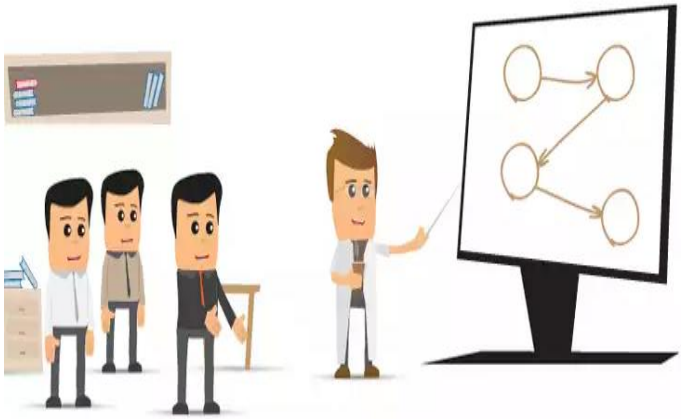
Additional Material

- **Chapter 9, 16 & 23** of "Software Engineering at Google" by Winters, T., Wright, H. and Manshreck, T.
- **Chapter 6** of "Software Engineering: a practitioner's approach" by Roger S. Pressman & Bruce R. Maxim.

Globalization in SE

Past: Tasks were performed by an assigned single person or a local in-house team.

Now: Teams consist of members around the world, working on the same project at any point in time.



Collaboration vs Working Independently

Q1. Why would you prefer to work collaboratively?

Q2. Why would you prefer to work independently?

Collaboration vs Working Independently

Reasons for working collaboratively

Knowledge sharing/learning
Teamwork
Productivity
Increased code quality
Professional activities
Positive if it is asynchronous
Improved review

Reasons for working independently

Work on personal interests
No pressure\own pace
Working collaboratively is time consuming
Dependency on others
Different timezone
Work on isolated contributions
Not a core contributor
Coding is an independent task
Only collaborate if help is needed

Source: Constantino et al. (2020)

Challenges/Barriers to Collaborative Development

- Knowledge
  - Bus factor (the risk of the project stalling due to a missing key contributor)
  - Knowledge sharing
- Time
- Missing or out-of-date documentation
- Guideline incompliance

Today

- Version Control
- Continuous Integration
- Code Review

Version Control

Discussion

How have students been keeping track of the changes to their source code (assignments/projects/side hustles) over time?

a) No method used

b) Regular backup

c) Version control

Version Control

- Version control (aka., source control, revision control) is a practice of keeping track of all changes made to the source code and other files of a software project.

- Version control systems are specialized systems that record every change made to the source code in some form of database.

Version Control

- Version control systems:
  - It allows to revert back to a previous state (version) of the project
  - easily make changes to a project without disrupting the work of other team members.
  - Popular examples SVN, git

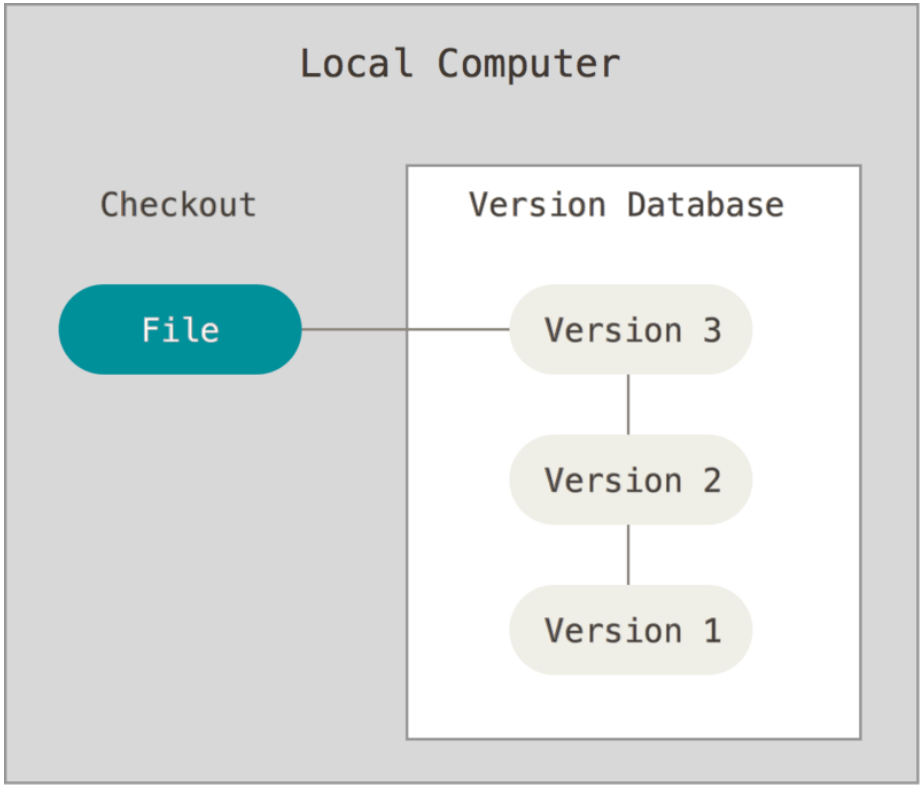
Version Control - Types

- Three (3) types of VCS:
  - Local VCS
  - Centralized VCS
  - Distributed VCS

Version Control - Types

Local VCS

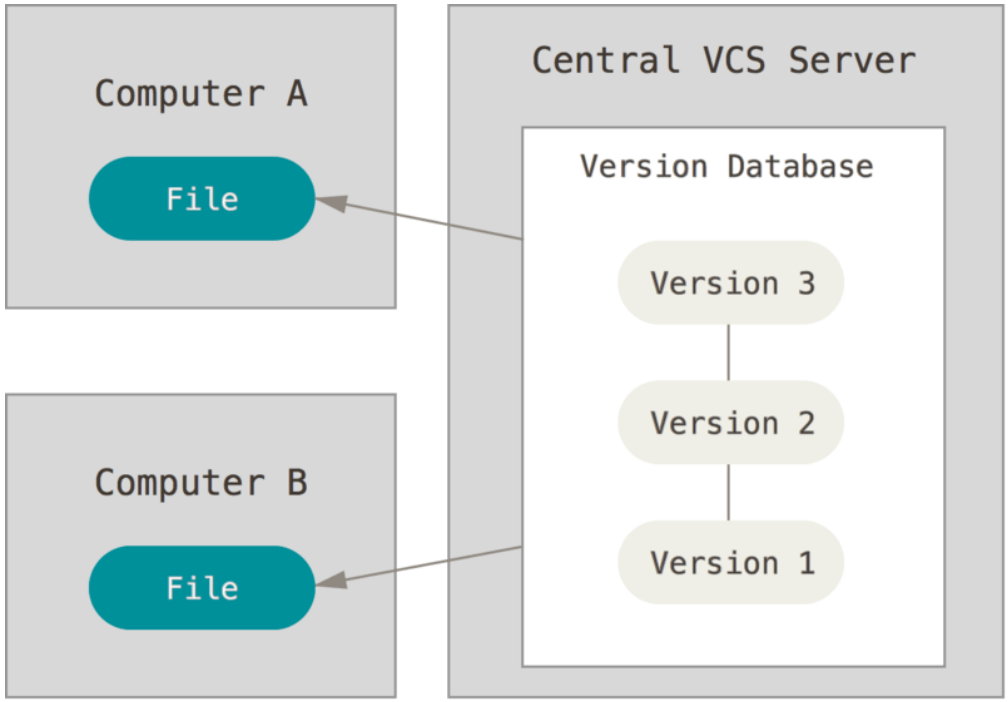
- Changes stored in a database on the local machine only
- Limits collaboration



Version Control - Types

Centralized VCS

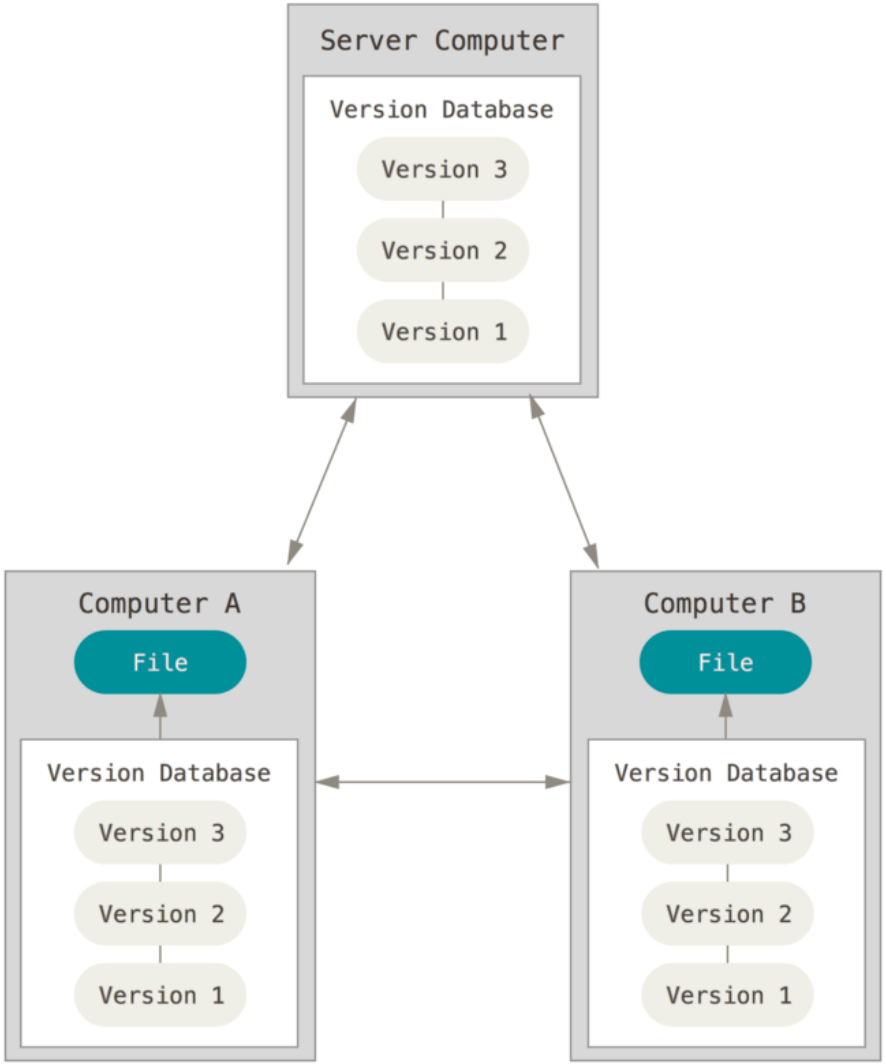
- Changes stored in a database on a central server
- Individual developers keep a local copy of the project, not the changes
- Drawback = single point of failure



Version Control - Types

Distributed VCS

- Changes on server are mirrored on each developer's local machine



Version Control

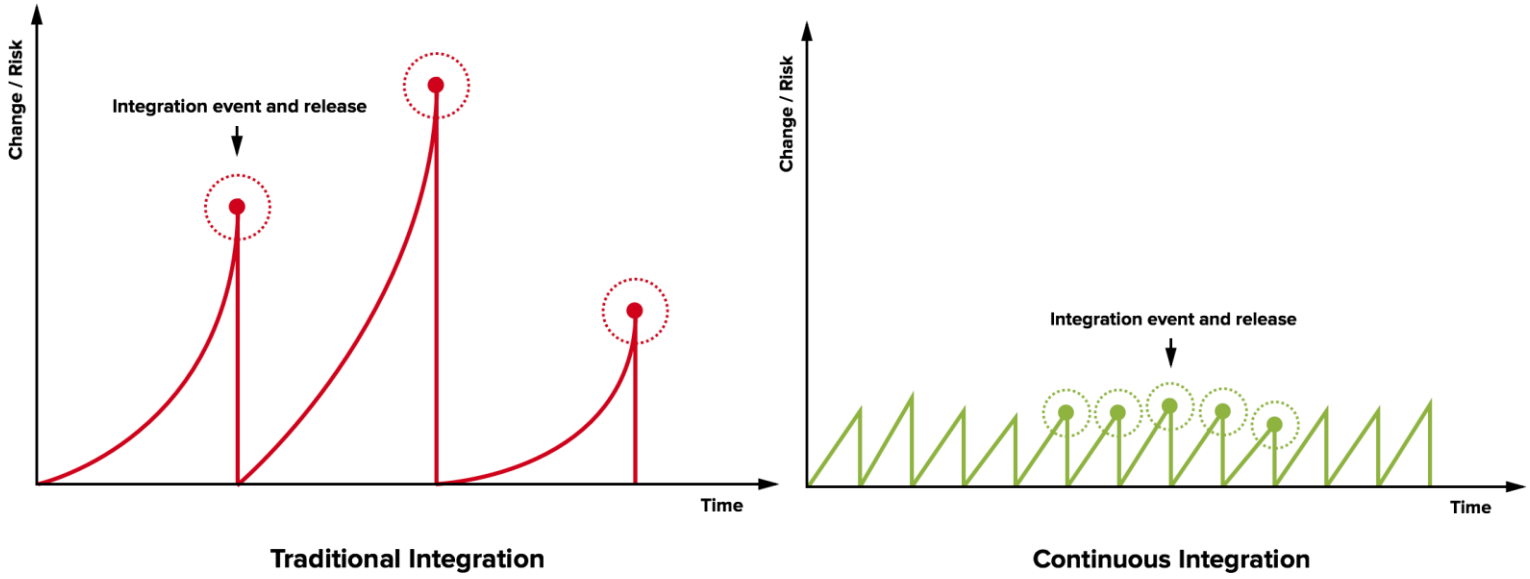
- Let’s have a short demo

Continuous Integration (CI)

- A major challenge during collaboration is **change conflicts**
  - Assuming developer A commits the changes they made to *file abc* into the central VCS. Developer B who also made other changes to the same file in his local computer and wants to commit the changes to the central VCS. What could go wrong?
  - Which of the changes should be kept, and which should be discarded?

Continuous Integration (CI)

- **Change conflicts** occur when the local copy of a project goes stale or out of date
- **Continuous Integration (CI)** is a practice aimed at preventing this from happening



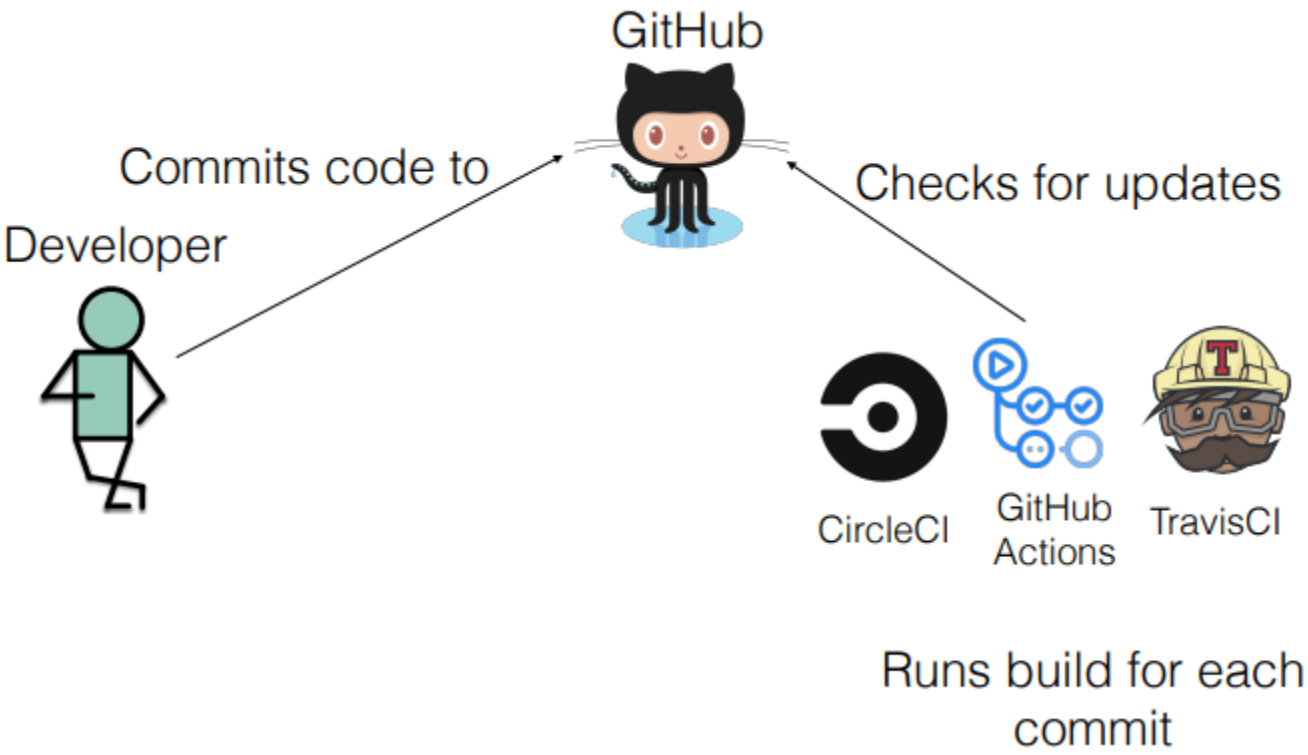
Continuous Integration (CI)

- CI is a software development practice that requires team members to frequently integrate their work.
  - Fundamental goal is to catch problems as early as possible
  - Allows for early feedback on the local changes
  - New commits are automatically verified
  - Changes are integrated if the build is successful

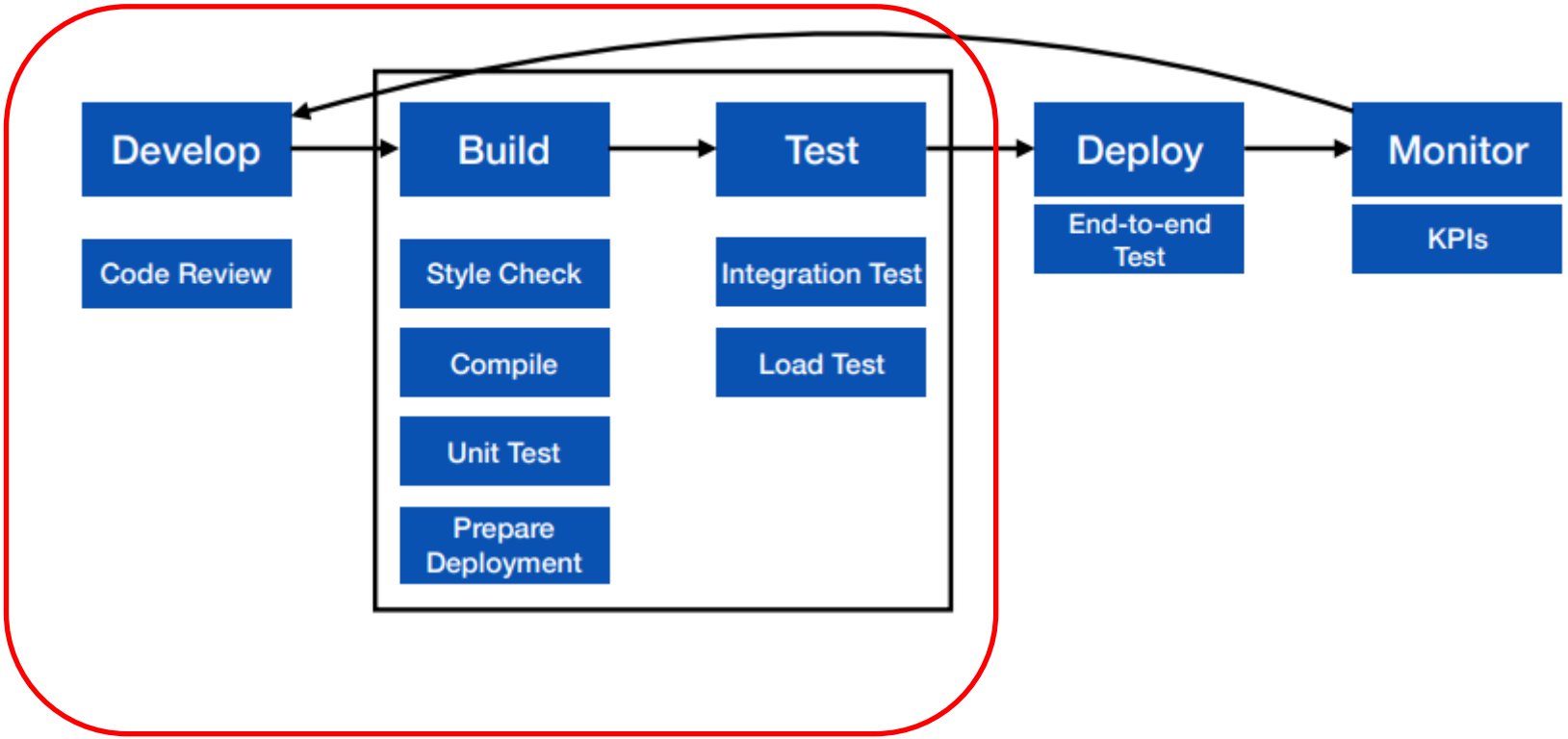
Continuous Integration (CI)

- The CI pipeline has three main stages
  - **Source stage** – the activation point of any CI action. Changes are made to code and a commit is issued. Commit triggers build stage
  - **Build stage** – checks if committed code compiles correctly. Possible for code to build on local machine but fail when combined with code on server.
  - **Test stage** – after a successful build, all tests are run to ensure new commit did not introduce any new bugs

CI in Practice

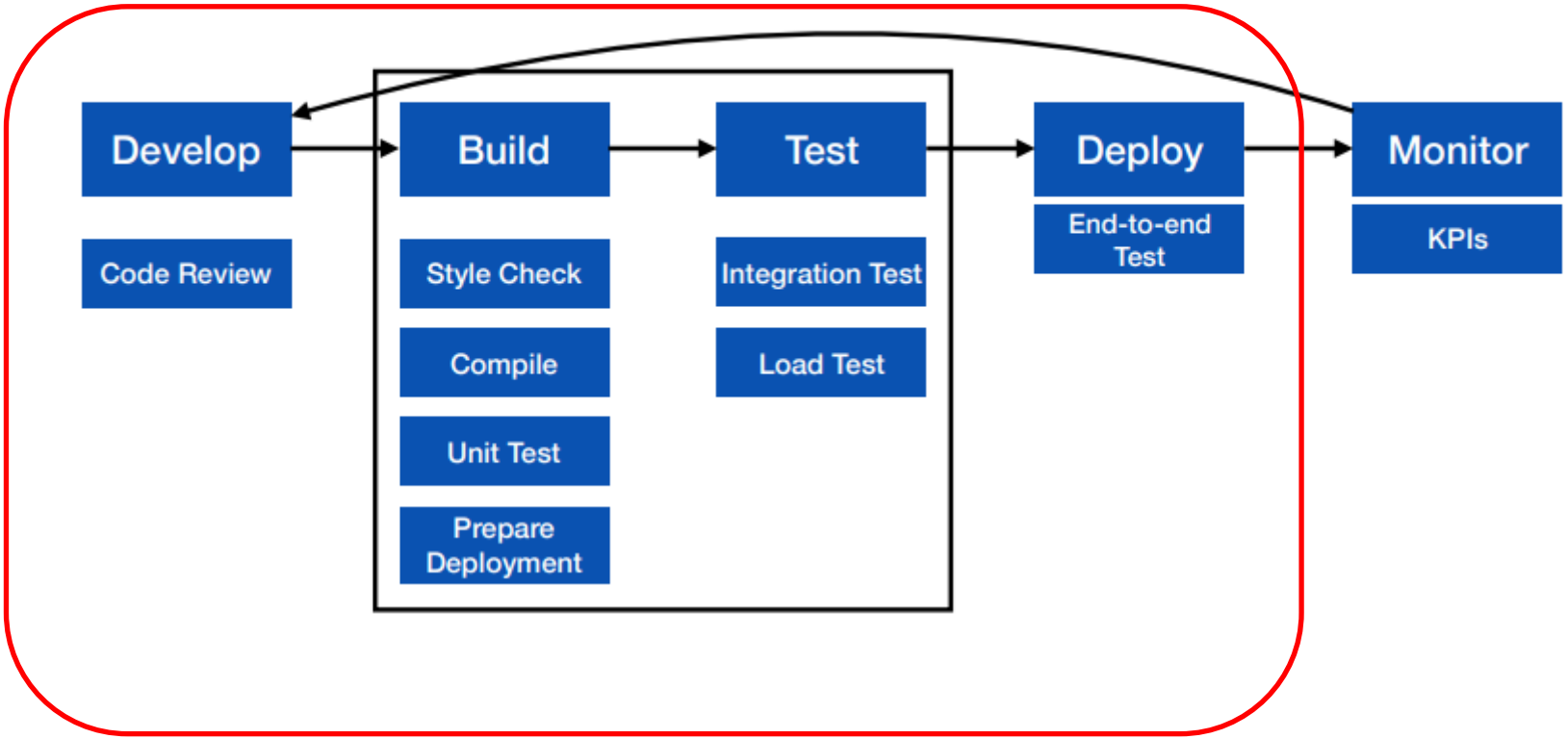


Continuous Integration (CI)



Continuous Integration

Continuous Integration (CI)



Continuous Delivery/ Continuous Deployment

Code Review

- Code review is “a manual inspection of source code by developers other than the author” (Sadowski et al., 2018)
- Common code review processes:
  - Asynchronous review by email (e.g., in the Linux kernel)
  - Tool-based (e.g., OpenStack uses Gerrit, Microsoft uses CodeFlow)
  - Pull-based review (e.g.,, OSS projects on GitHub)

Code Review

- Some tasks which are performed during code review:
  - Ensuring style standards are followed
  - Identifying code smells
  - Checking variable/method names
  - Providing adequate comments & documentation
  - Detecting the use of magic numbers

Code Review - Importance

- Expectations from the code review process include:
  - Finding defects
  - Improving code
  - Alternative solutions
  - Knowledge transfer
  - Shared code ownership
  - etc.

References

- Roger S. Pressman & Bruce R. Maxim (2014). Software Engineering: a practitioner's approach, 8th edition: McGraw-Hill
- Winters, T., Wright, H. and Manshreck, T. (2020). Software Engineering at Google: Lessons Learned from Programming over Time. 1st Edition. O'Reilly Media
- Constantino, K., Zhou, S., Souza, M., Figueiredo, E. and Kästner, C., 2020, June. Understanding collaborative software development: An interview study. In Proceedings of the 15th International Conference on Global Software Engineering (pp. 55-65).
- Sadowski, C., Söderberg, E., Church, L., Sipko, M. and Bacchelli, A., 2018, May. Modern code review: a case study at google. In Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice (pp. 181-190).
- Spinellis, D., 2005. Version control systems. IEEE Software, 22(5), pp.108-109.
- Fowler, M. and Foemmel, M., 2006. Continuous integration. Thought-Works) http://www. thoughtworks. com/Continuous Integration. pdf, 122(14), pp.1-7.