Introduction

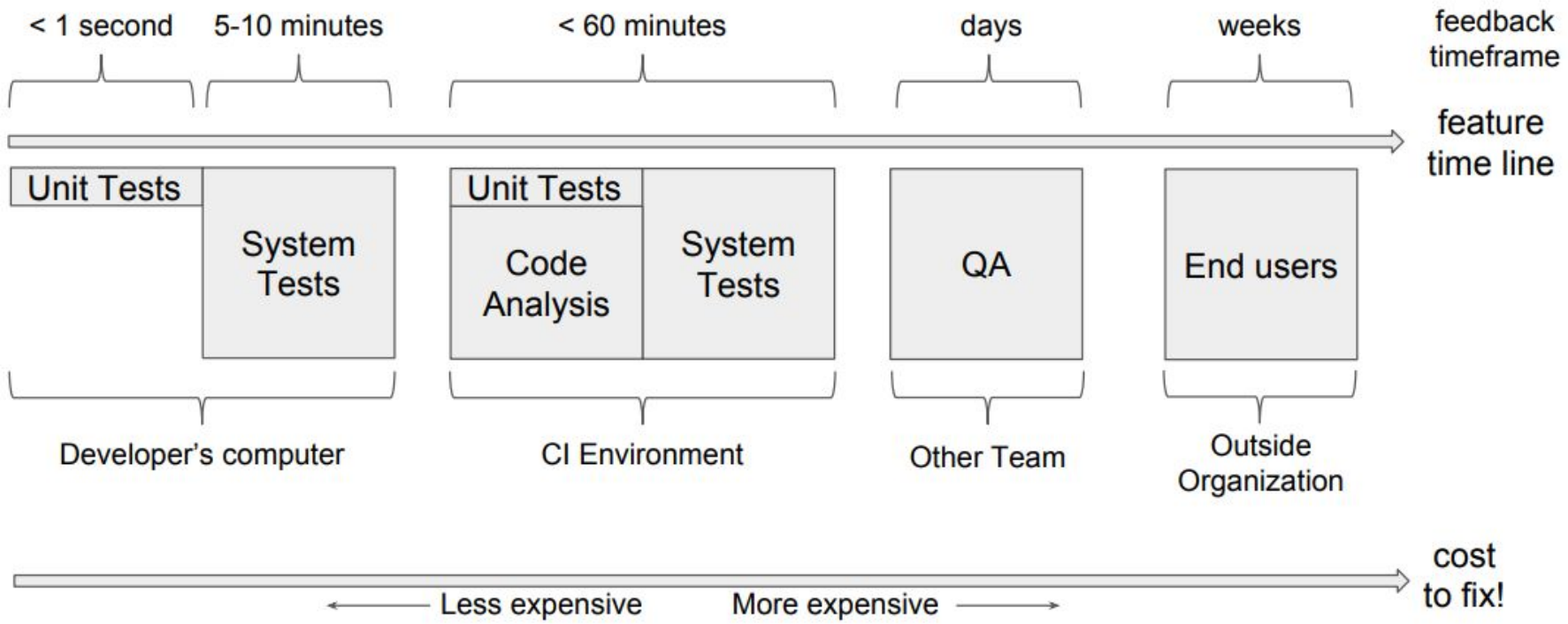
Continuous Integration and Test Driven Development

Continuous Integration

- What is Continuous Integration (CI)?
  - A software development practice where members of a team integrate their work frequently
  - Usually each member integrates at least daily - leading to multiple integrations per day.
  - Each integration is verified by an automated build and test
    - To detect integration errors as quickly as possible.
- Why continuous integration?
  - Easier to handle smaller integrations often, than one big at the end.
  - **All** tests are **always** run automatically for **all changes** to the codebase; not at the developer's discretion
  - Code quality feedback comes sooner rather than later
  - Reduces "It works on my computer"-related issues
  - Increases confidence in the product

Continuous Integration

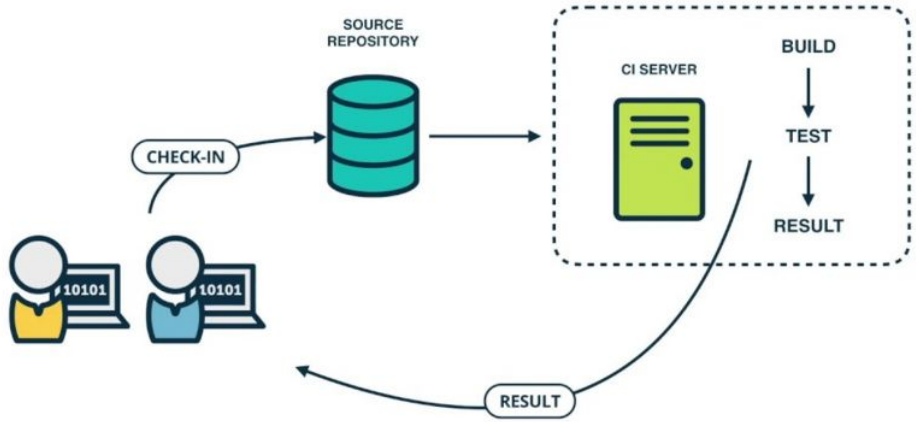
- Why continuous integration?
  - Finding and fixing bugs earlier reduce the cost.



Continuous Integration

- How do we make it happen?
  - Maintain a single source repository
  - Automate the build
  - Automate testing and make the build self-testing
  - Everyone commits to the **mainline** every day
  - Every commit should build the **mainline** on an integration machine (CI server)
  - Keep the build fast
  - Test in a clone of the production environment
  - Make results easily accessible
- A CI server is used to automate
  - Building, testing, reporting and etc.

Some CI system examples

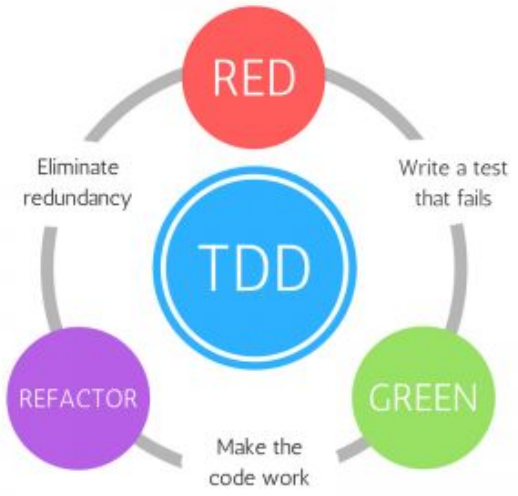


Test-Driven Development (TDD)

- Test-Driven Development is a technique for building software incrementally
  - Small steps help form better components
- A failing test is followed immediately by code satisfying the test
  - Healthy code growth, components are less prone of bugs.
- The focus is on the requirements instead of coding
  - Tests shall be based on the requirements
  - Tests shall enforce a specific behaviour
  - Tests shall encourage refactoring and make components easier to understand
    - Refactoring is the process of restructuring code without changing its external behavior.
- Test automation is key to TDD
  - Testing is automatically performed by machine over and over. No need for manual testing.

Test-Driven Development (TDD)

- Three rules of TDD
  - Write production code only to pass a failing unit test.
  - Write no more of a unit test than sufficient to fail.
    - Compilation failures are failures
  - Write no more production code than necessary to pass the one failing unit test.
- The steps of the TDD cycle
  - Add a small test
  - Run all the tests and see the new one fails
  - Make the small changes needed to pass the failing test
  - Run all the tests and see the new one passes
  - Refactor to remove duplication and improve expressiveness



Continuous Integration

- Some useful links
  - CI/CD - An Overview
  - Professional Guides: CI/CD
  - What is CI/CD?
  - Continuous Integration
  - What is TDD?
  - What is Test Driven Development (TDD)?
  - What is Test Driven Development?
  - What are TDD and automated testing?
  - What is Automated Testing?