Software Testing Methodology

Continuous Integration and Continuous Deployment (CI & CD)

2019-11-29

Daniel Graves

daniel.graves@huawei.com

Contents

- **Background**
  - Why test software?
  - Common software development pipelines
  - Unit testing and integration testing
- Automating the development pipeline
  - Continuous integration
  - Continuous deployment
- Summary

Background: Why test software?

- Hive thermostat
  - App sometimes repeatedly reset users' homes to 32° C
  - Bug in app software to control home temperature

https://hexus.net/ce/news/gadgets/90947-hive-smart-thermostat-dumb-moment-sets-homes-32c/

Background: Why test software?

- Knight Capital Group
  - Company trading algorithm had a software bug
  - Started to buy high and trade low for 30 minutes
  - $440 million lost – 4 times more than net income the previous year

https://www.bloomberg.com/news/articles/2012-08-02/knight-shows-how-to-lose-440-million-in-30-minutes

- Mars Climate Orbiter (1998)
  - Sub contractor used imperial units for the navigation system instead of metric specified by NASA

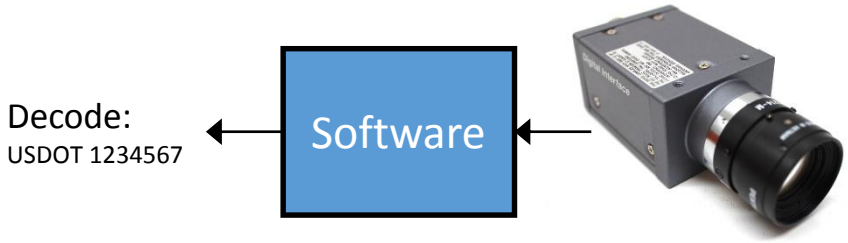
https://mars.jpl.nasa.gov/msp98/news/mco990930.html

- World War III … almost (1983)
  - Soviet early warning satellites interpreted sunlight reflections off clouds as missile launches
  - Lt Col Stanislav Petrov intercepted the messages as faulty

https://www.pri.org/stories/2017-09-21/soviet-officer-who-averted-nuclear-war

Background: My experience

- Writing machine vision software to read USDOT numbers on trucks



- Elusive bug that caused the software to crash at random times
- Extensive testing needed to find the root cause (process and memory profiling)
- Cause:
  - Debugging revealed when the garbage collector (GC) moved code in memory thinking it was managed code but in fact it contained unmanaged code that contains direct pointers to memory locations. An access violation was thrown
  - This was a result of the GC not being informed that the function pointer needed to be pinned in memory since it was unmanaged. Pinning unmanaged memory basically prevents it from being moved by GC

What is software testing?

- Unit testing
- Integration testing
- Functional testing
- End-to-end testing
- Acceptance testing
- Performance testing
- Exploratory testing
- …

What is software testing?

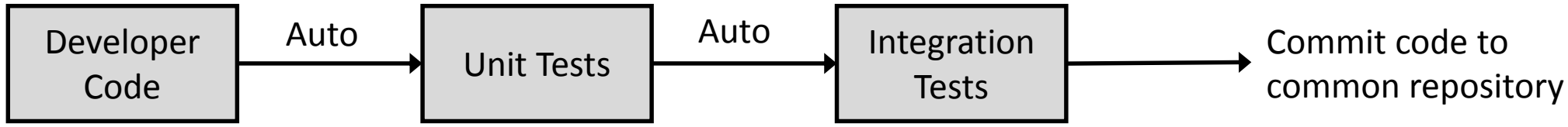
- Unit testing – test methods of classes and components of modules
- Integration testing – test that modules work well together
- Functional testing – test outputs of actions against requirements
- End-to-end testing – test user behaviors from start to finish
- Acceptance testing – test entire application against requirements
- Performance testing – test system under heavy load
- Exploratory testing – unscripted testing to find bugs

What can be automated?

- Unit testing – **yes**, simple to implement
- Integration testing – **yes**, a little more complex to implement
- Functional testing – **yes**, complex to implement, correct output
- End-to-end testing – *partially*, expensive/hard to maintain
- Acceptance testing – *partially*, expensive/hard to maintain
- Performance testing – *partially*, costly to implement
- Exploratory testing – **no**, not until we have AI

Testing in industry

- Developers focus on unit testing and integration testing
  - Unit tests are run with every change in software (should be fast)
  - Integration tests are run frequently (typically these tests are slower)



  - Unit tests and integration tests are typically separated in test code
- Common tools
  - "Build servers" – software is tested, integrated and compiled by these servers
    - Software is reproducible and traceable, versioning, archives
  - Continuous integration – tools for testing software builds frequently

Testing in industry

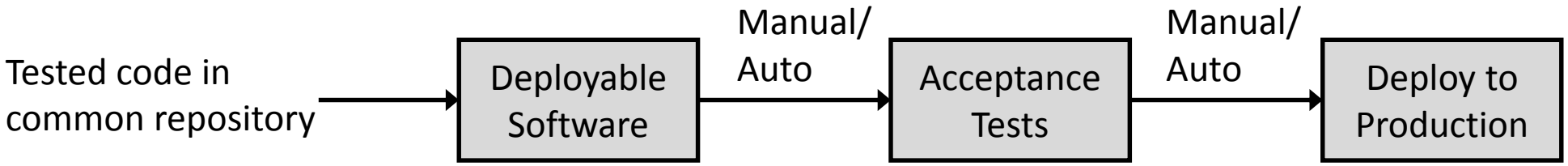
- Who writes unit tests and integration tests?
  - Developers **write their own tests**
    - In large projects, separate test teams may write integration tests
  - Programmers know the right tests!
  - Tests should be comprehensive
    - Cover important cases include edge cases
  - Write project code along side test code

Testing in industry

- Why write test code and development code in parallel?
  - Allows developers to evolve software quickly
  - Developers can test immediately after making changes, adding features, fixing bugs, refactoring code, or making improvements
  - Enables rapid software development and frequent improvements

Testing in industry

- Who writes acceptance tests?
  - Quality Assurance (QA) or test teams usually **write these tests**
  - These include performance tests, functional tests, end-to-end tests, etc
  - Tests the business logic of the software
  - Software must pass these tests before going to production
  - We automate as much as we can (but it's a lot of work)



Contents

- Background
  - Why test software?
  - Common software development pipelines
  - Unit testing and integration testing
- **Automating the development pipeline**
  - Continuous integration
  - Continuous deployment
- Summary

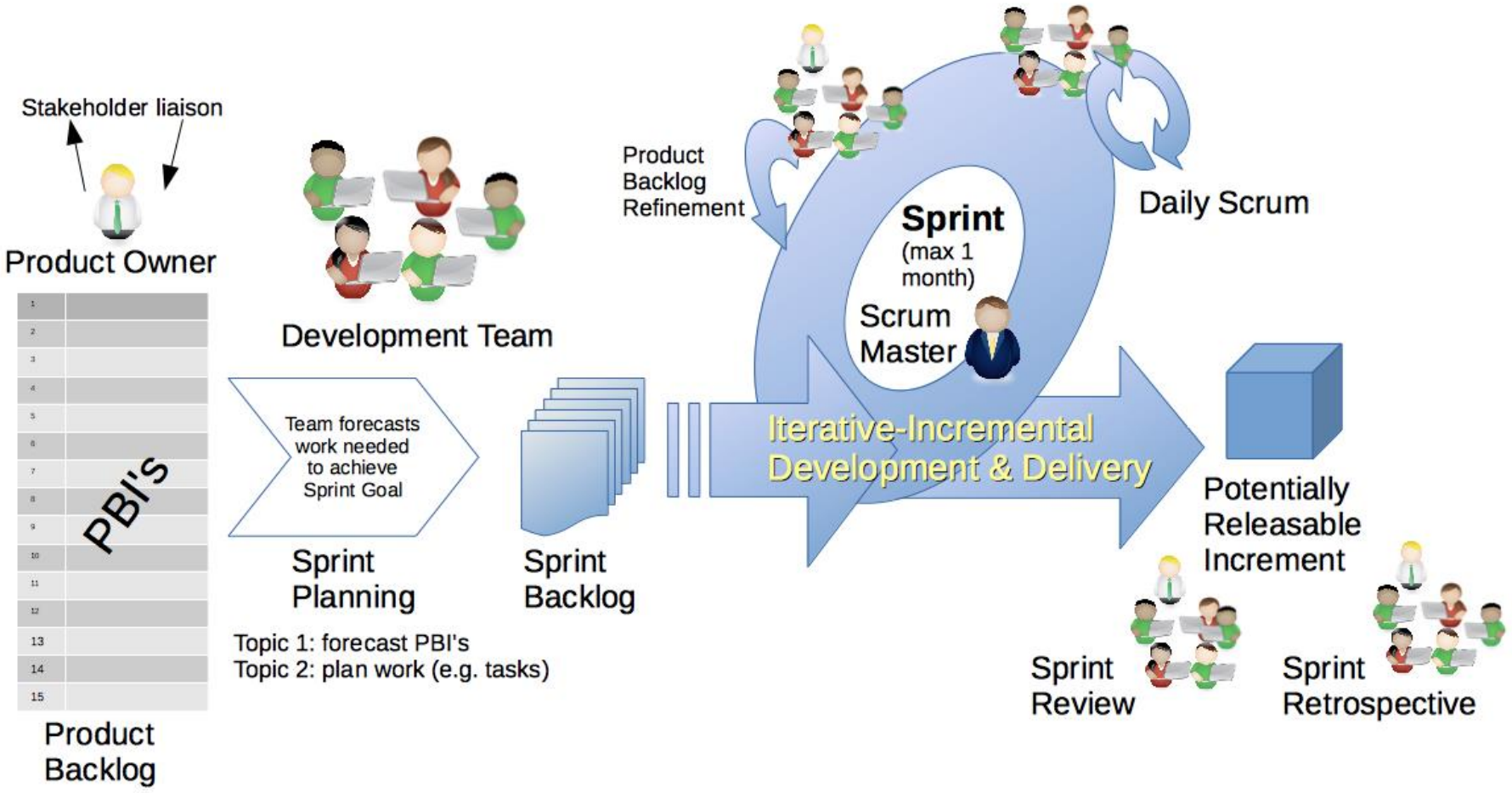
Software development in the “wild”

- Software today is often highly iterative and flexible
  - Agile software development process is pervasive from small startups to big
  - Frequently plan, develop, deliver, and improve software
  - Being flexibility and adaptive is key
- Examples of Agile software development processes
  - Scrum
  - Kanban
  - …
- Automation is important to making agile development efficient and flexible

Scrum as agile software development

- Divided into “sprints”, e.g. a few weeks or a month
- Define some goals (which can change) that can be completed within a sprint
- Start by doing sprint planning, i.e. identify the goal, required backlog items
- A backlog is a “todo” list for the project
- Developers accept “doable” backlog items for a sprint and breaks the task down into a sprint backlog, i.e. what needs to be done in the sprint
- The development happens with daily “stand up” meetings to re-plan and adapt quickly as needed
  - What did I complete yesterday toward the sprint goal?
  - What will I complete today toward the sprint goal?
  - Are there any roadblocks for the team to achieve the sprint goal?
- Sprint ends with a review - review work and present to the stakeholders (e.g. possibly a demo)
- Sprint retrospective – reflect and agree on actions to improve the process

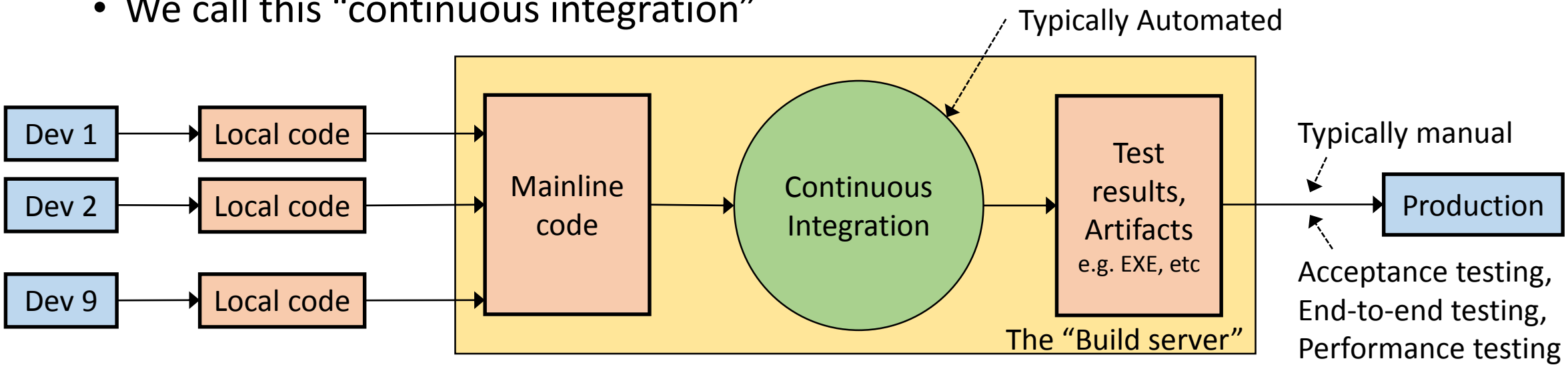
What does this looks like?



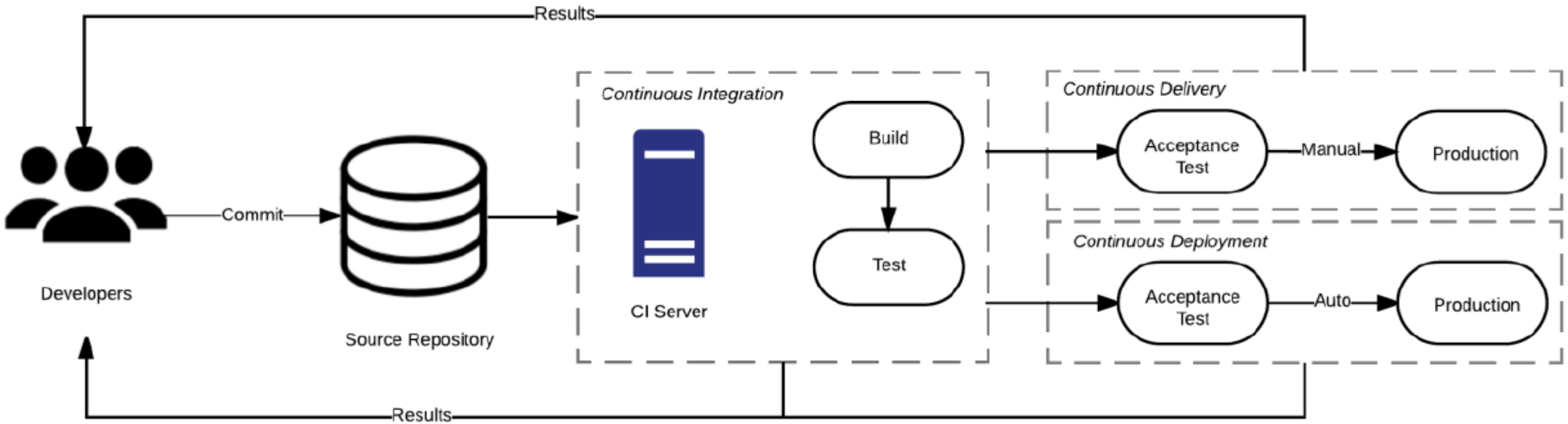
Source: https://en.wikipedia.org/wiki/Scrum_%28software_development%29

Where does automation fit in?

- Agile development is highly iterative, so ….
  - Software must be tested and retested frequently
  - Automated testing is the key to tackling this problem
  - Developers must develop code to automate the testing of their contributions
- We use tools for automatic integration, testing and building
  - We call this “continuous integration”



Continuous deployment



*Developers run their unit tests and integration tests locally before commit

*CI server runs all the unit tests and integration tests for the project

Source: Shahin, M., Ali Babar, M., Zhu, L., “Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices”, IEEE Access, 2017

Common tools for continuous integration

- Versioning: e.g. Github, etc
- Build: Make, Ant, MSBuild, etc
- CI Tools: e.g. Jenkins, CircleCI, TeamCity, Github Actions, etc
- Testing: JUnit, NUnit, UnitTest

Configuring continuous integration

- If you wish to get started with continuous integration
- There are many tutorials online
- There is a popular tutorial for git and teamcity (free)
- TeamCity is a free build server that supports unit tests, continuous integration and compiling of software
- https://www.tutorialspoint.com/continuous_integration/index.htm

Contents

- Background
  - Why test software?
  - Common software development pipelines
  - Unit testing and integration testing
- Automating the development pipeline
  - Continuous integration
  - Continuous deployment
- **Summary**

Benefits and Challenges of CI

- Benefits
  - Rapid completion of deployable code
  - Flexible and adaptable
  - Less bugs in code
  - Happier stakeholders – they receive the solution they want
    - Features are developed quickly for fast feedback and rapid re-planning
    - Requirements can change overtime

Benefits and Challenges of CI

- Challenges with continuous integration
  - Making sure to keep unit tests and integration tests up-to-date
  - Making sure tests are comprehensive
    - Missing tests can be difficult to catch bugs later
  - Making sure unit tests run quickly for rapid feedback on changes to code
    - Otherwise developer productivity is lost