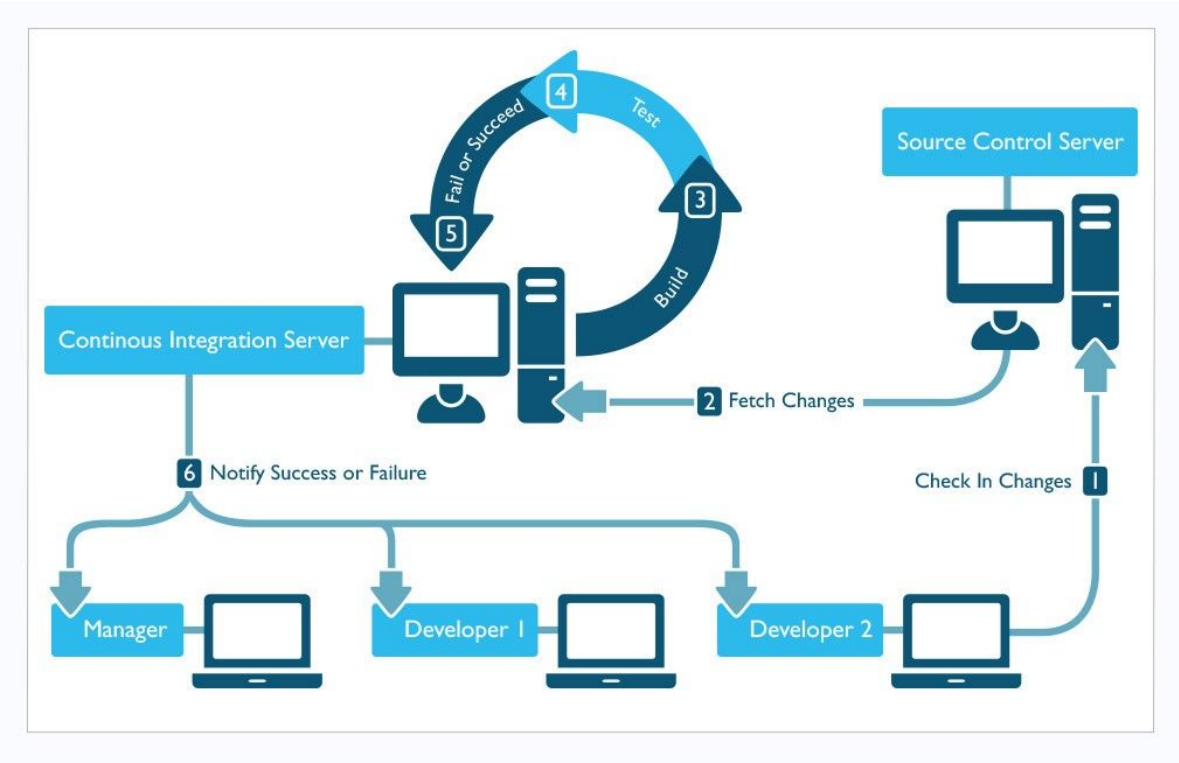
Lecture Notes
October 4, 2018

Reminder: A specific IA mentor is assigned to each team. The entire team should meet with their mentor to discuss the team project *before* submitting the revision (CVN teams should arrange for an online meeting with the designated CVN IA = Anthony).

You can and should attend office hours of *other* IAs, but each IA is only responsible for their own teams.

The Revised Project Proposal is due Tuesday, October 9, 11:59pm. The revised proposal should take into consideration feedback provided by your IA mentor and any other members of the teaching staff.

Continuous integration hooks a version control system with build and testing tools that run after every commit to the shared repository, or nightly for very large codebases, so errors can be detected quickly



Continuous Deployment adds automatic deployment to continuous integration - Installs on the production server(s) and/or produces a patch for customers

- Demo Maven (Anthony)
- Demo Travis CI (Siddharth)

CI assumes that testers can write assertions that automated testing tools can use to check that the software under test produced the expected output for the given input

But how do the testers know in advance what the expected output should be?

Or, alternatively, how can testers write assertions to check that the actual output is correct for the given input?

Customer requirements, such as user stories, may not specify exactly how to determine the correct output for every possible input, and may not even define was "correctness" means

Need a "test oracle"

Possible test oracles:

Human tester "knows"

Naive - if it doesn't crash, hang, or produce obviously nonsensical output, it works!

Formal specification - mandatory for safety-critical systems, rare for business/consumer applications

Wait until users complain that the software didn't do what they expected, then write test cases to check for what they say they expected 😁

Pseudo-oracles may be available in some cases, where one test case acts as an oracle for another test case

Regression testing is the most common form of pseudo-oracle, where we repeat test cases with known results on old version to see if they have the same or different results on new version.

Some of those test cases *should* change results, e.g., when we fix a bug that caused a test case to fail, we expect it to pass. But unrelated test cases should not - if they do, that indicates a new bug.

"Differential testing" is another kind of pseudo-orcale, which requires a second independently developed implementation of the same functionality

- If they disagree, one or both is wrong
- If they agree, does that mean both are right?

Say we have a machine learning application that is supposed to learn from series of traffic images how to control the steering of a self-driving car, so it obeys traffic laws and doesn't cause accidents.

How do we know what is "correct" steering for a given real-world driving scenario? How would we write test cases?

Show DeepTest images

"Metamorphic testing" is another approach to pseudo-oracles: We have an original input with its original output. We create a new test case by generating a new input from the original input, and predicting the expected new output from the original output. If the actual new output deviates from the predicted output, there is (probably) a bug.

Read for Tuesday:

- User Stories
- Estimation
- Iterations
- Planning

Team assignment Revised Project Proposal is due next Tuesday, 11:59pm.

Next pair assignment Bug Hunt due Tuesday, October 16, 10:10am.