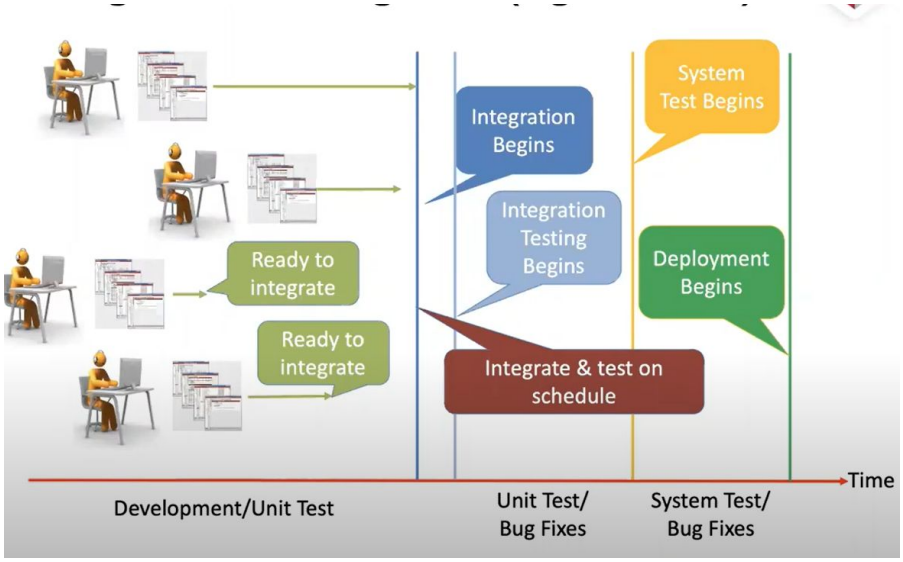
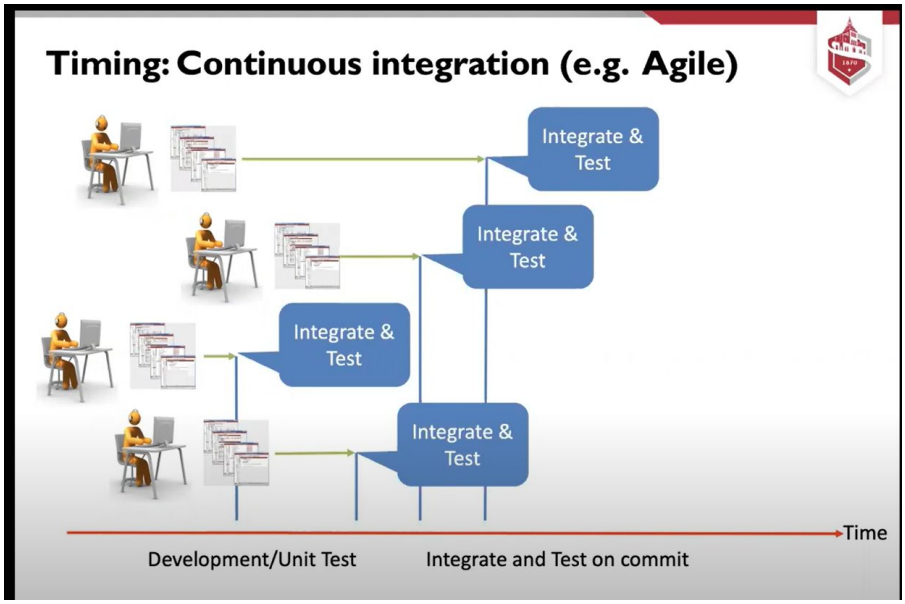
SSW-555
Agile
Lecture 5

Continuous Integration and Pair Programming

Deferred Integration (waterfall)



Continuous Integration (Agile)



CI Best Practices

- Maintain a single source code repo
- Keep everything needed to build in the repo
- Automate the build process
- Automate testing new builds
- Frequent code commits from all developers: at least daily if not more frequent
- Build and test in deployment environment

- Keep builds fast
- Fix build problems quickly
- Make it easy for everyone to get latest build

CI Benefits

- **Reduced risk**, detect and fix bugs more quickly and easily
- Converge on a solution more quickly
- Relatively little new untested code at any given time
- Developers fix bugs when code fresh in their minds
- Fewer bugs associated w/ automated testing
- Enables continuous delivery to customers and increase communication between devs and customers

CI Tools

- Github, jenkins, bamboo

Pair Programming overview

- 2 programmers sit in front of the same computer
- 1 programmer the driver types
- 1 programmer the navigator watches, catches mistakes, suggests alternatives, designs tests
- The 2 programmers switch roles frequently: every 15-20 minutes
- Works best if both are co-located but it can also work if not

Pair Programming Guidelines

- Change roles often
- Work with someone at the same level of experience
- Take breaks
- Communicate: 15 seconds without talking is a long time, 30 seconds without talking is an eternity, constant communication - explain what you're doing
- Listen to your partner and be a good listener

Research Results

- Pairs work almost twice as fast as individuals
- Higher quality work than on own
- Some pairings may not work effectively
- Higher quality leads to less time and effort in testing and fixing bugs

Pair Programming Myths

- Take twice as long, working alone, only good for training, share credit for everything, the navigator finds only syntax mistakes, etc.
- Take twice as long: evidence pairs are twice as fast as individuals; quality increases
- Never get to work alone: 30% of time working alone
- Only works with the right partner: seems to work with almost anyone with similar skills
- Only good for training: different people bring different experiences and skills to bear
- Share credit: rewards can take many different forms, peer evaluation helps reward those who help the project
- Navigator finds only syntax errors: navigator has to be seeing the big picture, thinking at higher level of abstraction, driver thinking through the details
- Won't be able to concentrate: pairs engage in pair mental flow

Synergistic behaviors of Pair Programming

- **Pair pressure:** keep each other on task, less likely to be distracted, pairs treat their shared time as more valuable; pairs follow standard processes more readily (following coding style guidelines)
- **Pair negotiation**: share the same goal, healthy debate, bring different ideas and POVs, congratulate one another when they work out the best solution
- **Pair courage**: easier to get started if you know you have help, feedback from your partner is encouraging
- **Pair reviews**: formal code reviews are uncommon w/ agile methods, informal code reviews, more fun to pair than to do code inspections
- **Pair Debugging:** sometimes you need to describe the problem to someone else in order to solve it, thinking out loud
- **Pair learning:** observe and learn from how someone else solves the problem, pairs learn about the domain by working with others
- **Pair Trust:** the good of the many outweighs the good of the one; may lead to better quality