Continuous Integration and Testing

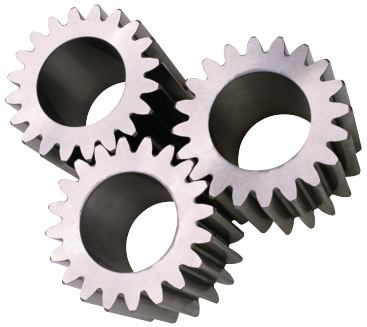
Integration Testing

CI & Test

Peas and Carrots

Name integrations (non-computer) you do every day

1. Schedules – you and your spouse (and kids?)
2. Integrate class into daily life
3. Buffet style eating
4. Integrate work and life
5. Technology into life
6. Integrating cooking

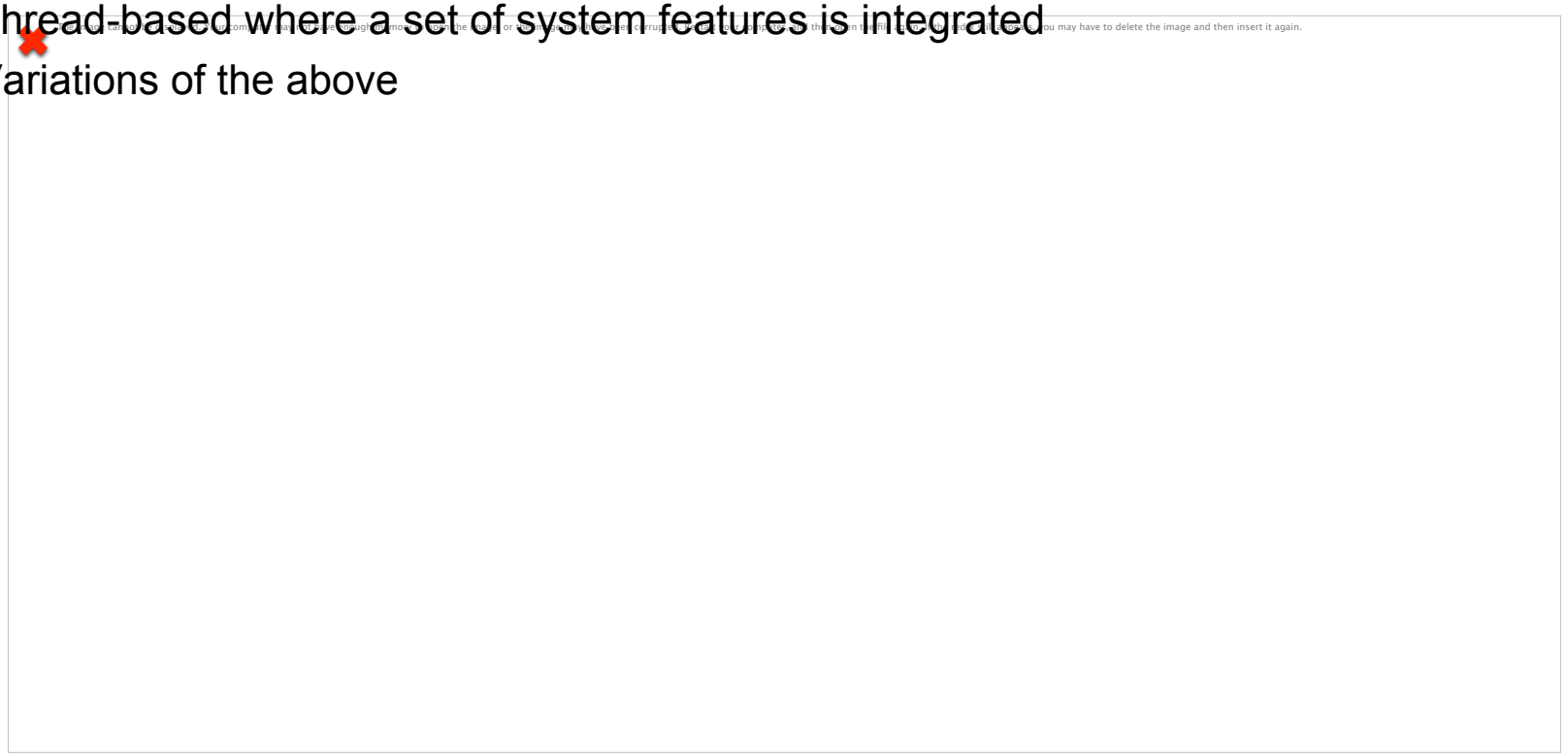


Integration Testing

The entire system is viewed as a collection of subsystems (sets of classes) determined during the system and object design.

The order in which the subsystems are selected for testing and integration determines the testing strategy

- Big bang integration (Nonincremental)
- Bottom up integration
- Top down integration
- Thread-based where a set of system features is integrated
- Variations of the above



Traditional Testing Practices

Testing occurs once, near end of project

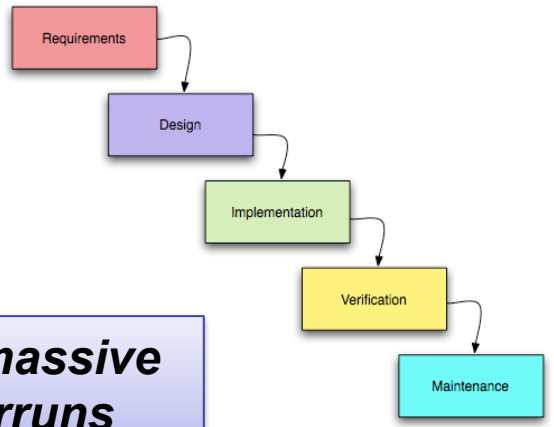
- Lots of lead time for test planning, test case generation, test lab and infrastructure setup

Test cases don't change (or don't change often)

- Cost of creating is paid once, not continuously
- Few changes to system once it is specified and designed

Tests executed periodically

- Initially to ensure system meets requirements
- Regression testing after significant change to ensure nothing broke



The problem: integration tests would fail, leading to massive rework, failure to meet delivery dates, and cost overruns

Agile: Continuous Integration & Testing

What is Continuous Integration?

- Integrate & build the system several times a day
- Integrate every time a task is completed
- Let's you know every day the status of the system

Continuous integration and relentless testing go hand-in-hand

By keeping the system integrated at all times, you increase the chance of catching defects early and improving the quality and timeliness of your product.

Continuous integration helps everyone see what is going on in the system at all times.

*If **testing** is good, why not do it all the time? (continuous testing)*

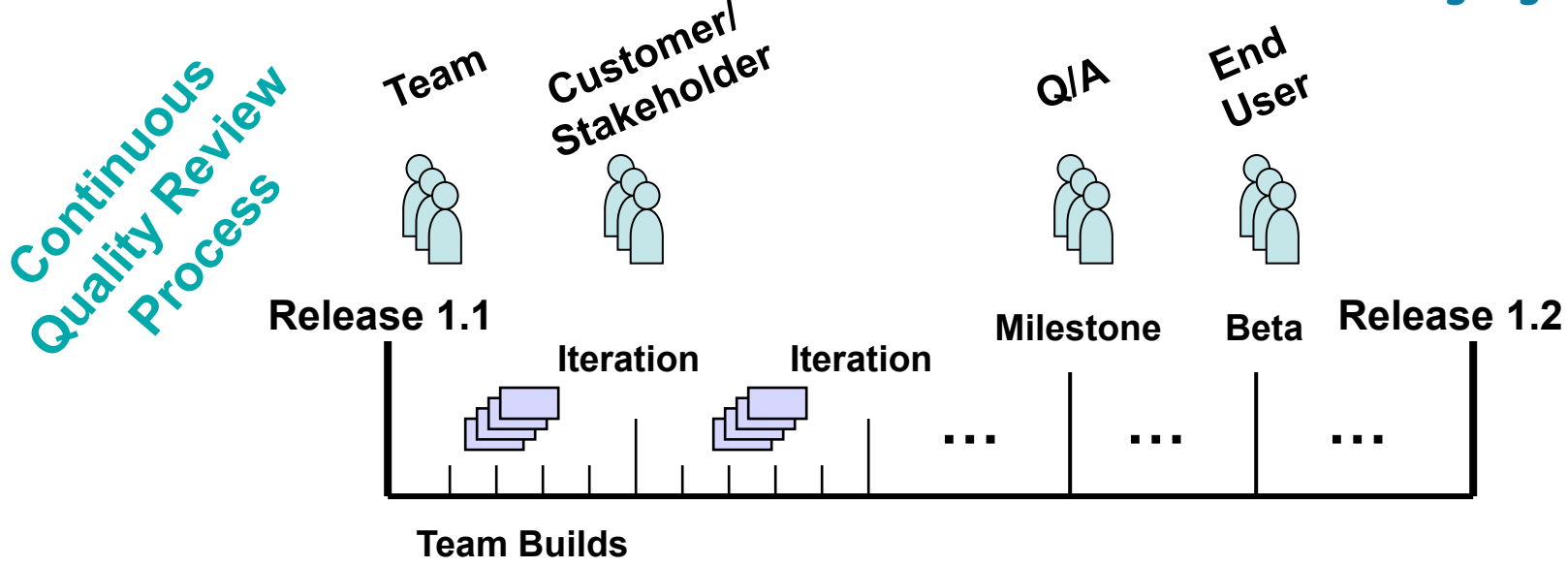
*If **integration** is good, why not do it several times a day? (continuous integration)*

*If **customer involvement** is good, why not show the business value and quality we are creating as we create it (continuous reporting)*

Development Process is Continuous

- No separate “test” phase – integrate and test continuously
- Features change during release – testing must adapt
- Testing starts on project’s Day 1
  - Initial plans, strategies, infrastructure required very early

Nightly builds
+ Adaptive planning
+ Continuous integration
= Testing nightmare



Continuous Development => Test Automation

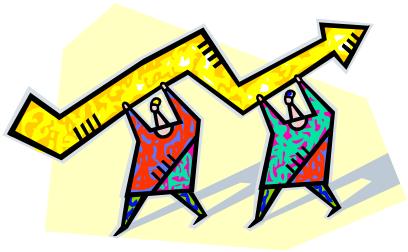
Continuous delivery and builds require automated testing

- Each build must be validated so future integrations build on a known quantity

Test frameworks provide infrastructure to quickly standup unit testing

- Governance & visibility: which test, on which build, metrics, trends

Type of build	What tests?	Level of automation
Developer delivery to CM	• "Unit" tests (per component)	All automated
Team "nightly" builds	• Add "Smoke test" for integration	Most automated, limited manual
Iteration	• Add quality tests for coverage, static analysis, metrics, etc.	Quality numbers obtained automatically
Milestone iteration	• Add additional scripts per test plan – performance, scalability, stability, etc.	Mixed automation/manual, but as automated as possible

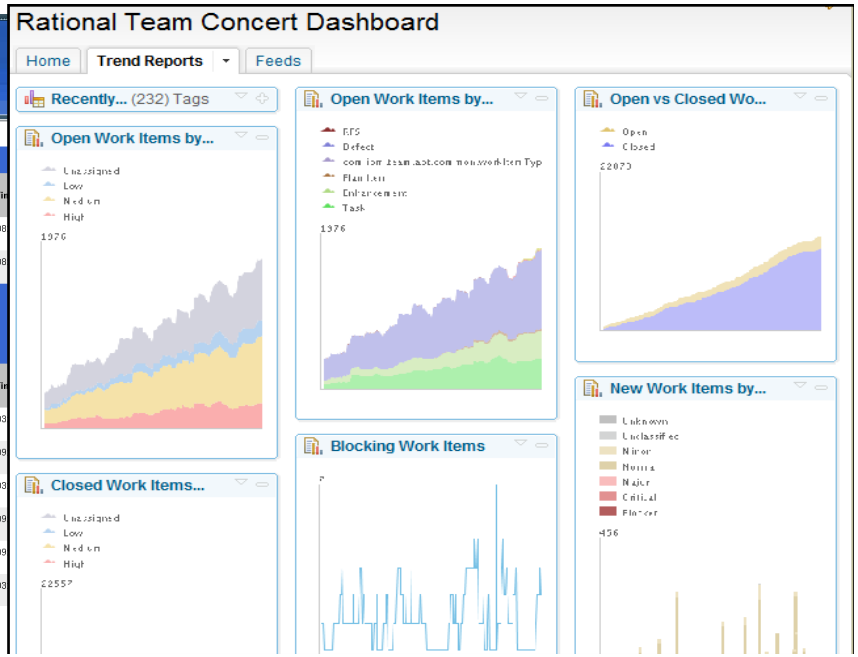


Agile Best Practice: Continuous Testing

Unit, System, and Integration tests can be run continuously!

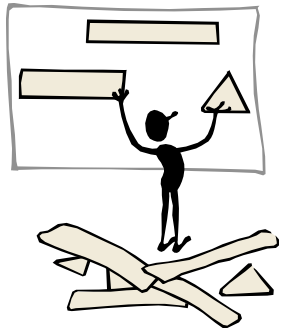
- Requires build/test automation and reporting framework
- Post results to a dashboard for all to see
  - Daily standup in the morning starts by checking if the dashboard is “green”
  - “*WHO BROKE THE BUILD???*” ← don’t let this be YOU!

Together with burndown charts, these show business value being built, with an attention to quality, at a sustainable pace



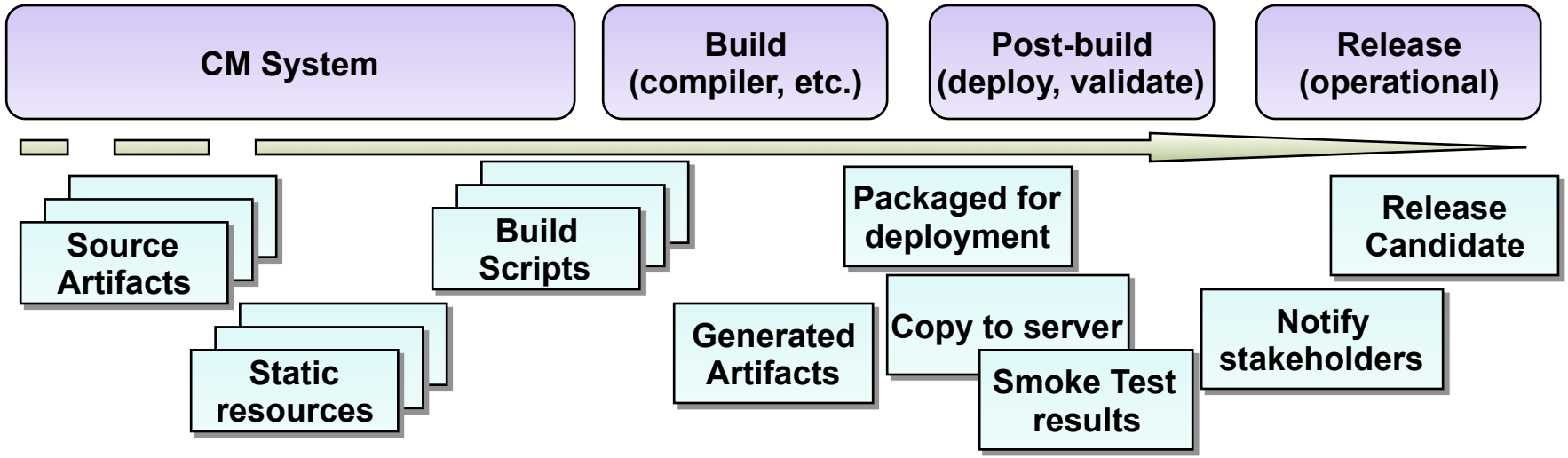
Fowler's 10 Best Practices for CI

From http://martinfowler.com/articles/continuousIntegration.html:



1. Maintain a Single Source Repository
2. *Automate the Build*
3. Make your Build Self-testing
4. Everyone Commits Everyday
5. *Every Commit should Build the Mainline on an Integration Machine*
6. Keep the Build Fast
7. Test in a Clone of the Production Environment
8. Make it easy for Anyone to get the Latest Executable
9. *Everyone can see what's Happening*
10. *Automate Deployment*

Putting it Together: Peas and Carrots



- All activities provide status & results to the Build Management System
- Build system executes build scripts against specific code configuration
  - Configuration Management System identifies artifact versions for build
- Deployment tools package build artifacts for target environment
- Post-build may include executing “smoke” tests
  - Tests designed for broad exercise of the system including known risk spots
- Release Candidate is your organization putting forward a release you claim is ready for acceptance by the customer