Build and Test Automation

Shortening the Feedback Loop

Agile Teams Embrace Build and Test Automation for Shorter Feedback Loops and Leverage

- **Shorter Feedback Loops**
  Automation shortens the time between implementation and feedback, while recent changes are top of mind.

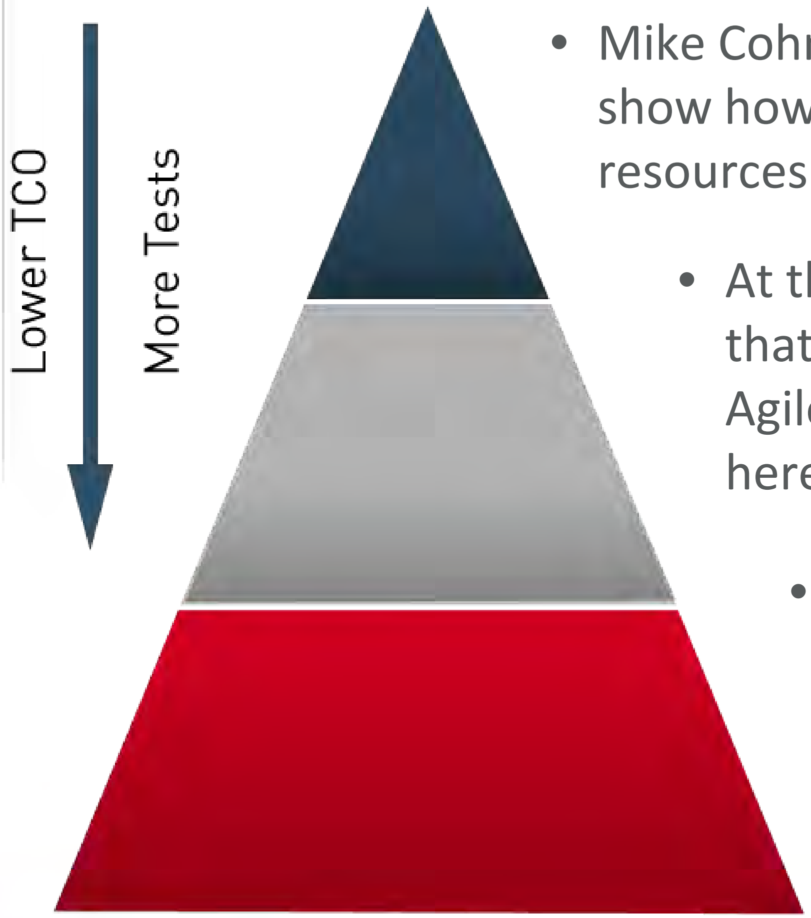
- **Leverage**
  Short, time-boxed iterations require us to be highly productive. Automation provides us with this leverage.

- **Lowering the Cost of Change**
  A key principle of Agile is to embrace change.
  Automated tests give us a safety net to refactor the system.

While testing out his new cereal mix on his horse, Dave gets some unexpected feed-back.

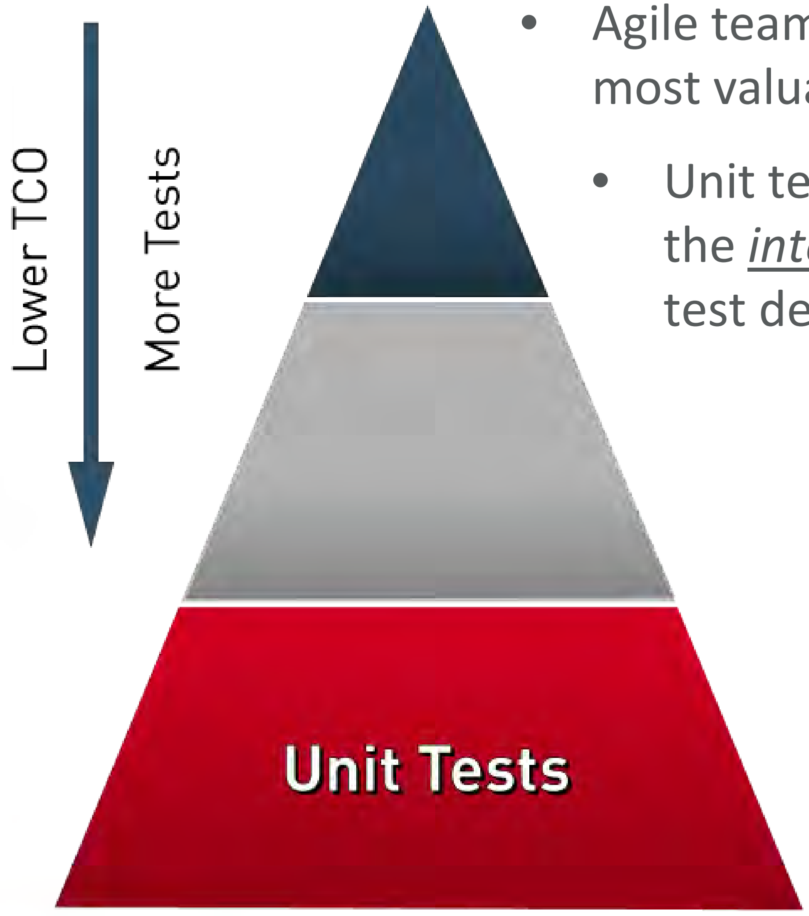
Best, Least Cost Automation

Mike Cohn's Testing Pyramid

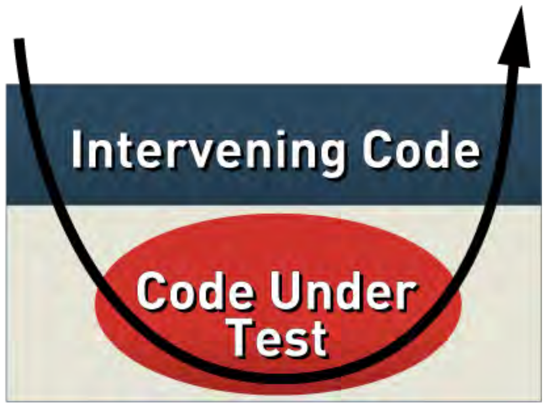


- Mike Cohn devised the notion of a testing pyramid to show how Agile teams best allocate their automation resources.
  - At the top of the pyramid we have tests that are the most expensive to maintain. Agile teams put the least investment here.
    - At the bottom of the pyramid, we maximize use of automated tests with the lowest Total Cost of Ownership (TCO).
      - So what are the three levels?...

Unit Tests have the highest ROI



- Agile teams have found that Unit Tests are the single most valuable approach to automated testing.
    - Unit tests have low TCO, because they minimize the *intervening code*, which is the major source of test death (when product changes break the test).



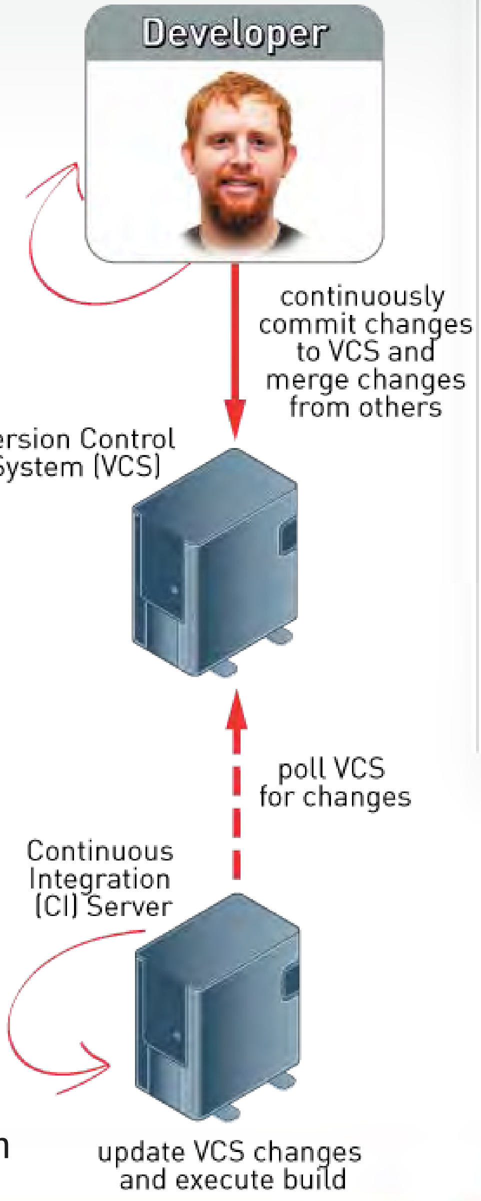
From Brian Marick's
"When Should a Test be Automated" –
http://exampler.com/testing-com/writings/automate.pdf

Unit Testing frees Resources for Acceptance and Exploratory Testing

- Unit tests are written by programmers, in the same language they develop in.
- Unit tests take the smallest piece of testable code, isolate it from the rest of the system, and determine if the results are as expected.
- Since unit tests isolate the code under test, they remain stable over time, with very low Total Cost of Ownership (TCO).
- Unit tests serve as the base of the regression suite. They lower cost of change as the team can modify code knowing there is a safety net.
- Unit Tests free Testers to do exploratory testing, automate acceptance tests and test non-functional requirements.

Unit Tests are used in conjunction with Continuous Integration (CI)

- CI is a software development practice, where…
- Team members integrate their work frequently, usually with each check-in to the version control system…
- An automated server builds the system, *runs all the unit tests* and reports “green” or “red” to indicate if the build is stable/broken.
- Building and Testing are done on a dedicated machine to eliminate “it works on my machine” dependencies.

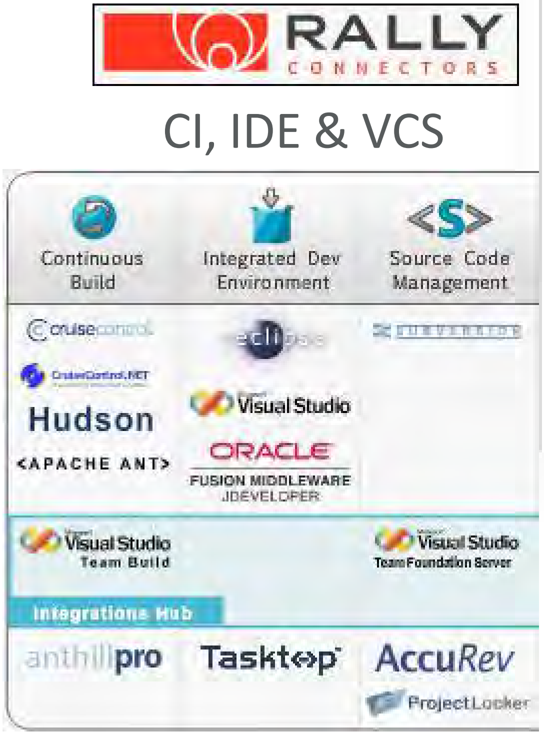

Developer
continuously commit changes to VCS and merge changes from others
Version Control System (VCS)
poll VCS for changes
Continuous Integration (CI) Server
update VCS changes and execute build

Jacob Orshalik
solutionsfit.com

How to Get Started with Continuous Integration

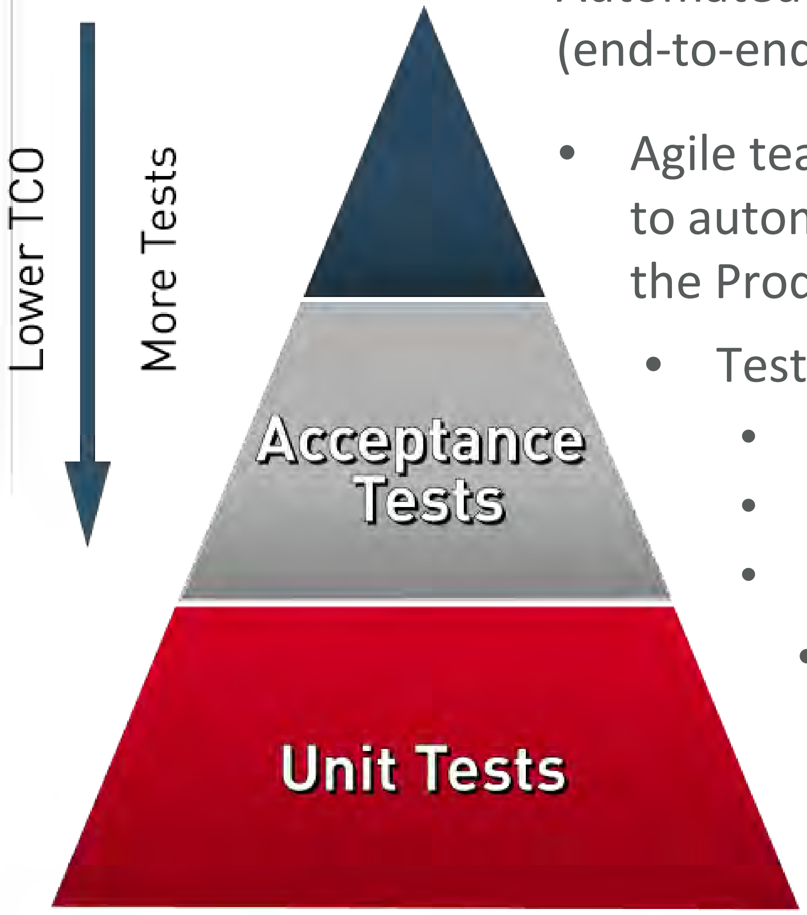
- Very High ROI. One of the first engineering practices an Agile teams should adopt.
- Many open source and commercial tools (/en.wikipedia.org/wiki/Continuous_integration)
- Keep builds and tests fast (move slow running tests to a nightly build)
- Adopt a "Stop the Line" mentality (all work stops until a broken build is fixed)
- Test in a clone of production (Every environmental difference results in risk)

CI, IDE & VCS



Rally seamlessly integrates with Build Automation Solutions

Functional/Acceptance Tests

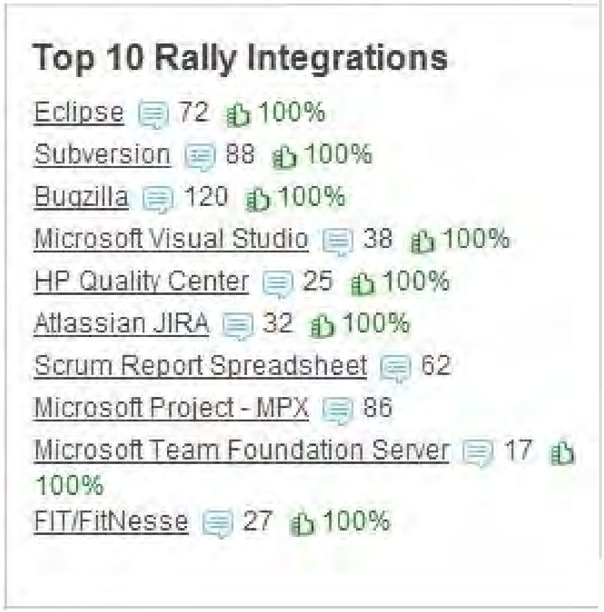


- Automated Functional Tests (end-to-end, not through the GUI) form the middle tier.
  - Agile teams typically have at least one per user story to automate the Acceptance Criteria established by the Product Owner.
    - Tests are developed iteratively
      - Start with the happy path
      - Automate the Acceptance Criteria
      - Add details as a result of exploratory learning
        - System Testability (e.g., services APIs, database migration, etc.) impacts cost. An Agile benefit is that testers raise testability issues while the team can do something about it.

Getting Started with Functional Testing

- Many commercial, open source tools. Rally integrates with most including HP Quality Center

- Start small and iterate – easy to get overwhelmed here. Use context-driven approaches to guide investment decisions



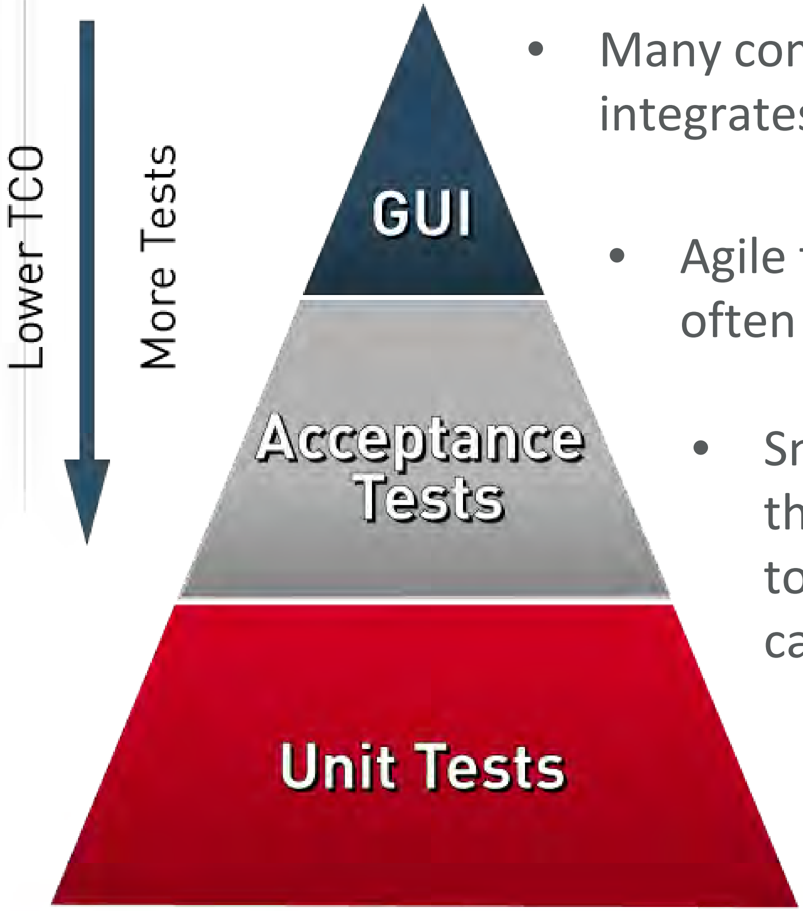
Top 10 Rally Integrations

Eclipse 72 100%
Subversion 88 100%
Bugzilla 120 100%
Microsoft Visual Studio 38 100%
HP Quality Center 25 100%
Atlassian JIRA 32 100%
Scrum Report Spreadsheet 62
Microsoft Project - MPX 86
Microsoft Team Foundation Server 17 100%
FIT/FitNesse 27 100%

- Emphasize Tester/Developer collaboration to support Testability

- Most teams run these tests in a nightly build as they are too slow to run at check-in. Your CI server should support nightly builds.

GUI Tests are Used for Smoke Tests



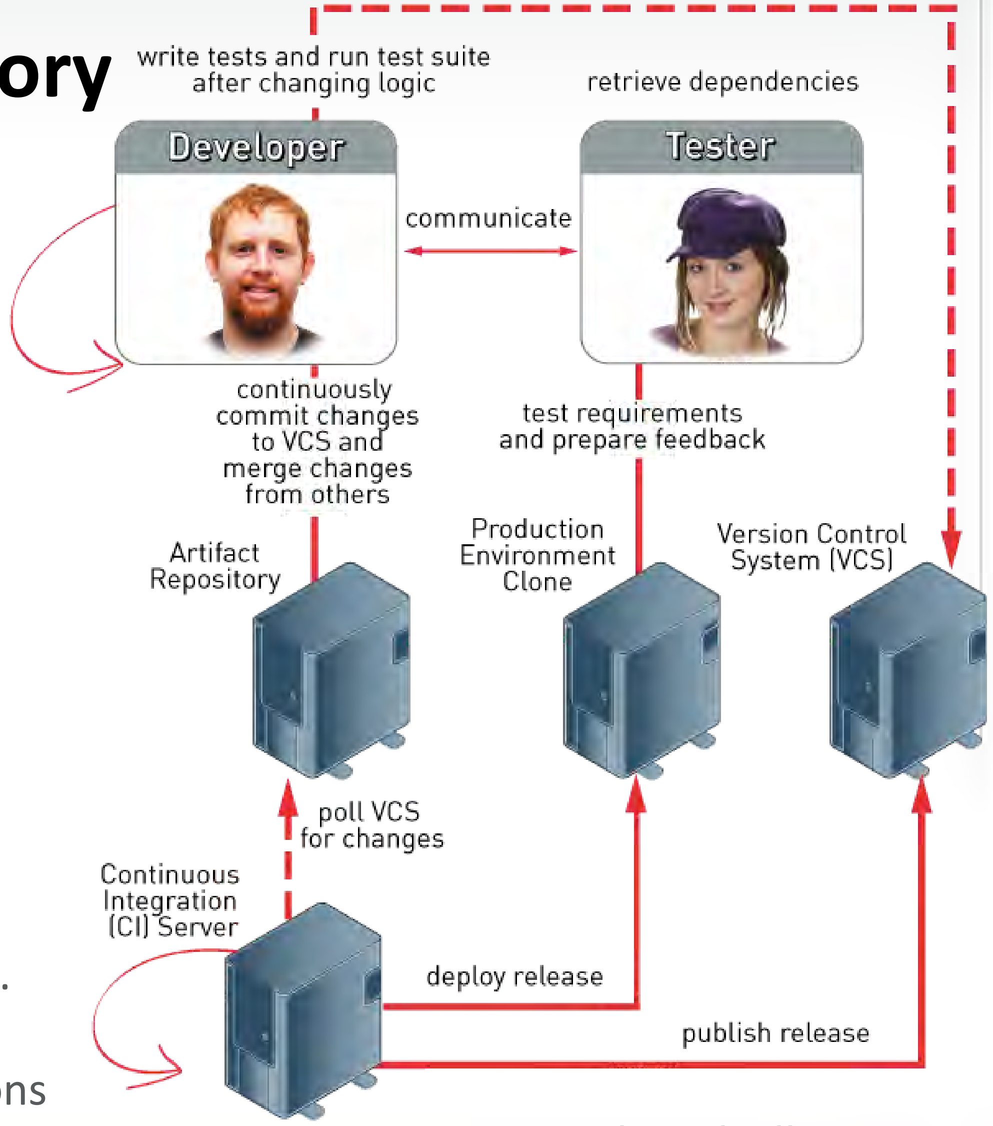
- Many commercial and open source tools. Rally integrates with most, including HP Quality Center.
- Agile teams have the fewest tests here. They often use GUI tests to create a "smoke test" suite.
- Smoke tests validate the basic functionality of the system. They exercise the system from end to end. They are not exhaustive tests, but are capable of detecting major problems.
- Smoke tests are typically run as part of the nightly build Continuous Integration cycle.

Build Artifact Repository

- Advanced Agile Teams put build automation in place so that Testers and Product Owners can pull any build at any time

- A "Test this Build" capability empowers Testers and POs to be completely in control of their testing environment, eliminating delays and a common source of frustration.

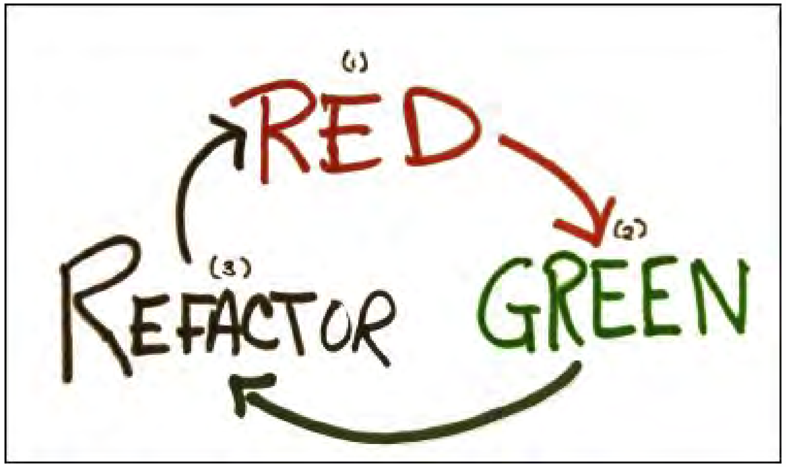
- A "Build Artifact Repository" can be as simple as a folder with zip files. There are also commercial tools and many Continuous Integration solutions include this capability (e.g., Maven).



write tests and run test suite after changing logic
retrieve dependencies
Developer
Tester
communicate
continuously commit changes to VCS and merge changes from others
test requirements and prepare feedback
Artifact Repository
Production Environment Clone
Version Control System (VCS)
poll VCS for changes
Continuous Integration (CI) Server
deploy release
publish release

Jacob Orshalik
solutionsfit.com

Test Driven Development



- Test Driven Development (TDD) and Acceptance Test Driven Development (ATDD) are technical practices that many Agile teams adopt.
- TDD and ATDD are specification and design approaches as opposed to the traditional validation function of testing. They are useful *emergent design* approaches for producing higher quality, more maintainable code.
- In TDD, a programmer writes a test before they write any production code. The test fails and the programmer writes just enough code to make it pass. This cycle is repeated on timelines of every few minutes.

Summary and Next Steps

Summary

- Automate for shorter feedback loops…
- Leverage and Productivity…
- …And a lower cost of change
- Continuous Integration + Unit Testing pays huge dividends
- Use Cohn's Testing Pyramid to Guide Automation Investment Decisions

Next Steps – Click Footer Links

- View the "Leading an Agile Transition" presentation to see how to get started.
- Explore the Implementing Agile Teams, Agile Test & Engineering Practices, and Rally JumpStart Service Offerings
- Request a free 30-day Rally Enterprise Edition trial.