When Agile Developers Test Well

Achieving the full promise of Agility

Developers:
How Well Do You Test?

What would your testers say?

… the users ...?

Have you been trained in testing?

Are there organization standards for testing?

... team standards?

... enforced standards?

What % of your time is rework?

Importance of Good Testing by Developers

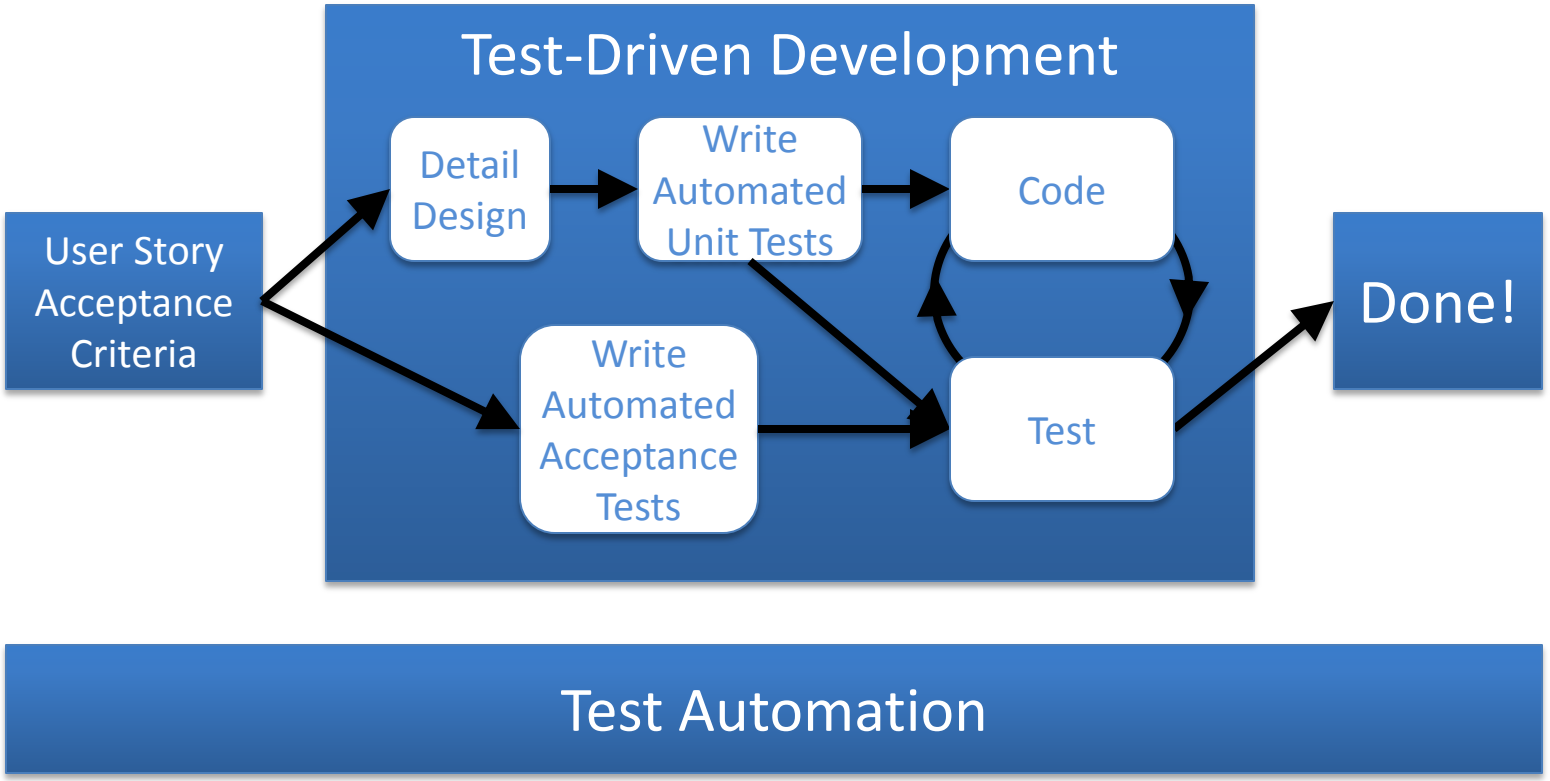
Agile Principle:

Continuous Attention to Technical Excellence

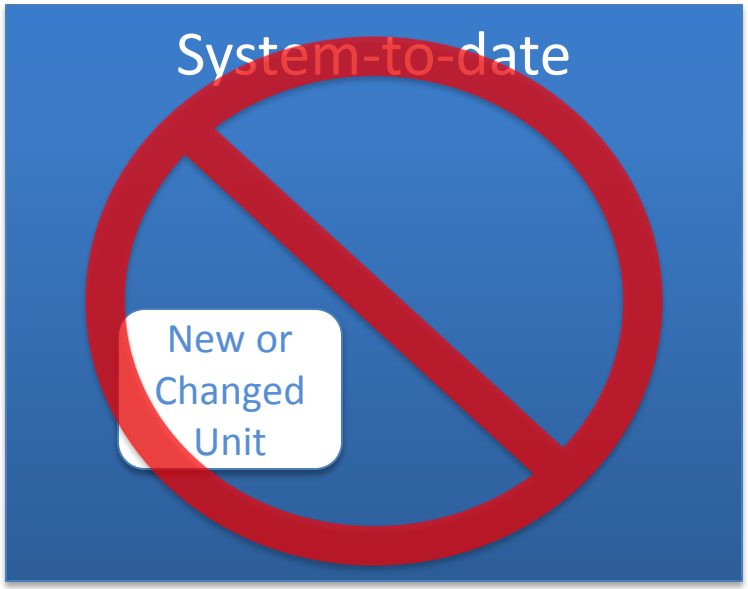
Agile Practices make:

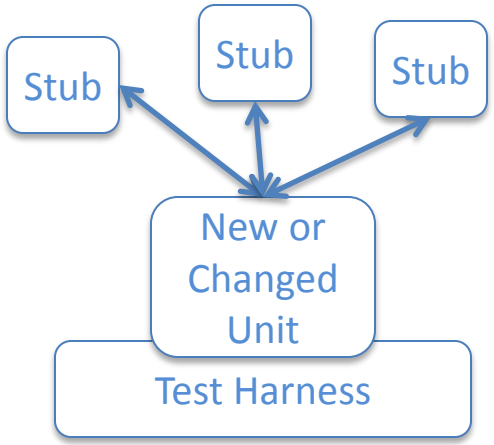
- Poor quality more visible
- Good testing more important

Good Developer Testing: What it Looks Like



True Unit Testing





OO Design & Unit Testing

Good Object Oriented Design …

- Encapsulation
- Information Hiding
- Loose coupling
- Separation of concerns

Using an OO Language ≠ Good OO Design

Enables True Unit Testing

- A simple test harness
- A few simple stubs

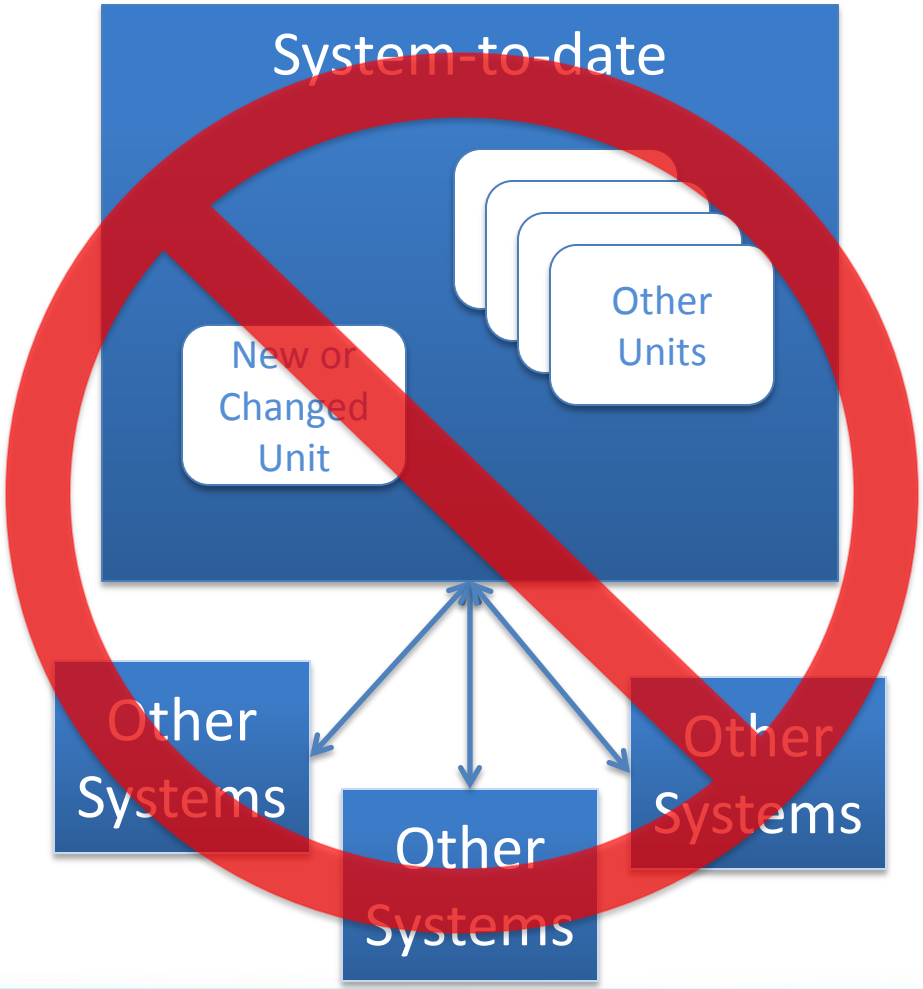
Automating Unit Testing

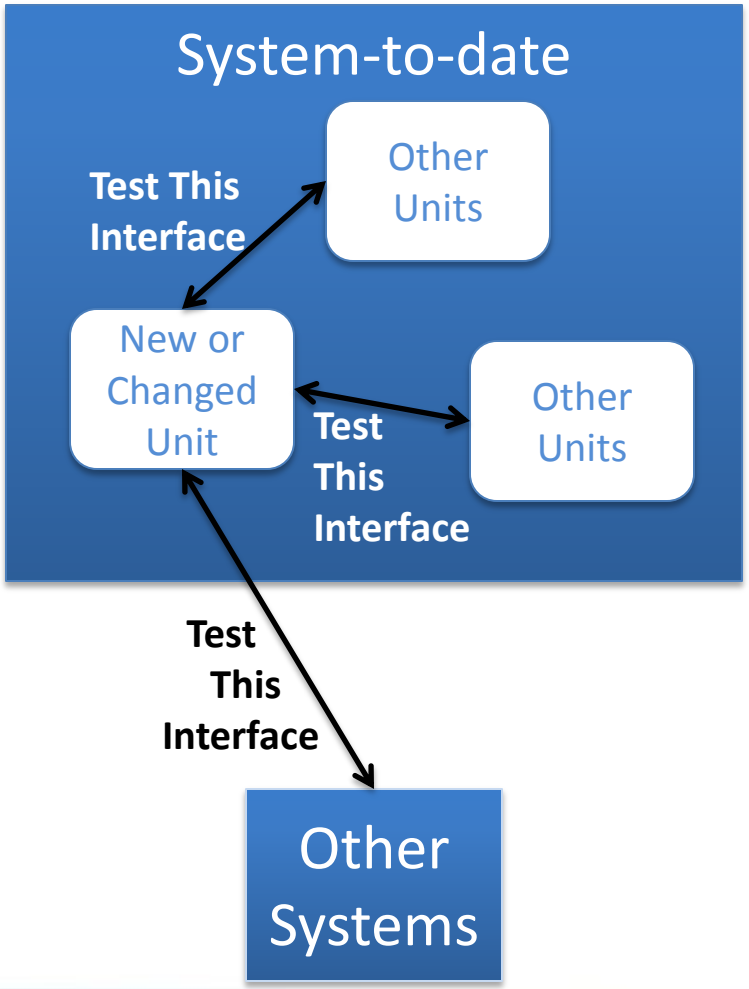
- IDE has built in test automation?

  (e.g. JUnit or NUnit?)

  USE IT!
- Using an OO Language?
  - Write Test classes
- Not OO?
  - Write test functions

True Integration Testing



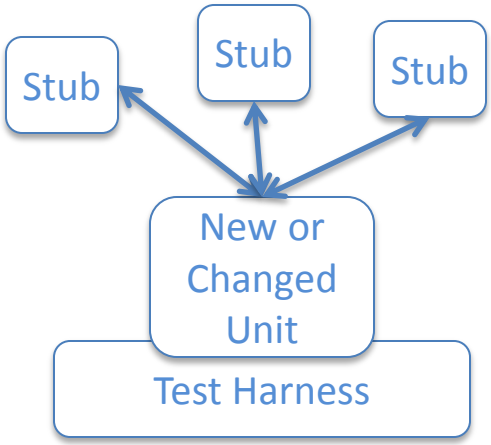


Testing Interfaces

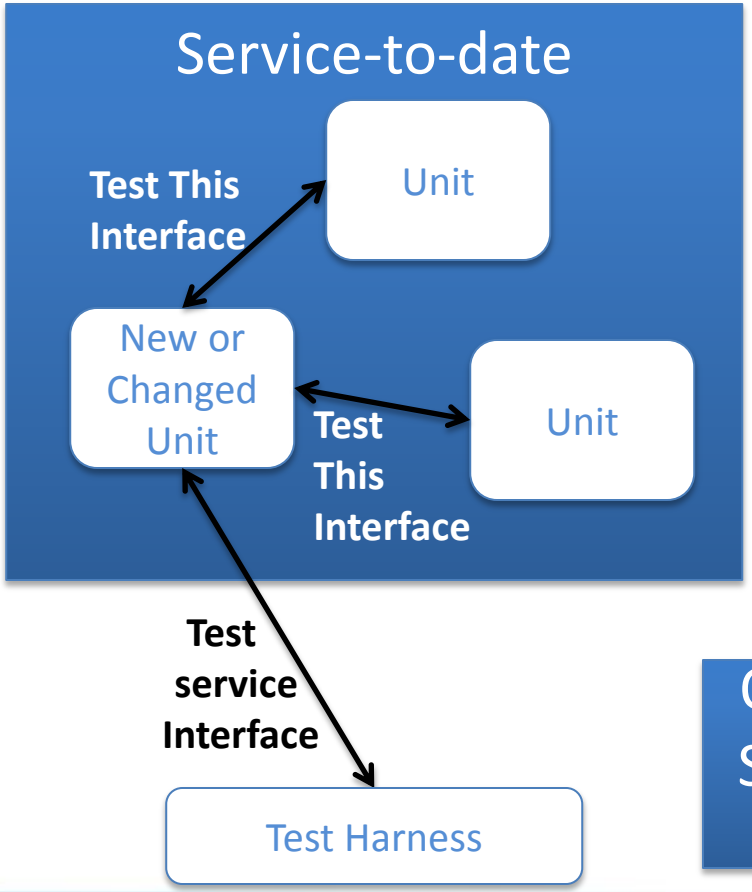
- Interfaces among Units within the System
  - Test all potential system states
  - Test all possible parameter values (including error values)
- Interfaces to other Systems
  - Test all potential states (this and other system)
  - Test all possible parameter values
  - Test all possible error scenarios
- Exposed APIs
  - How will the API be abused?

Testing Services

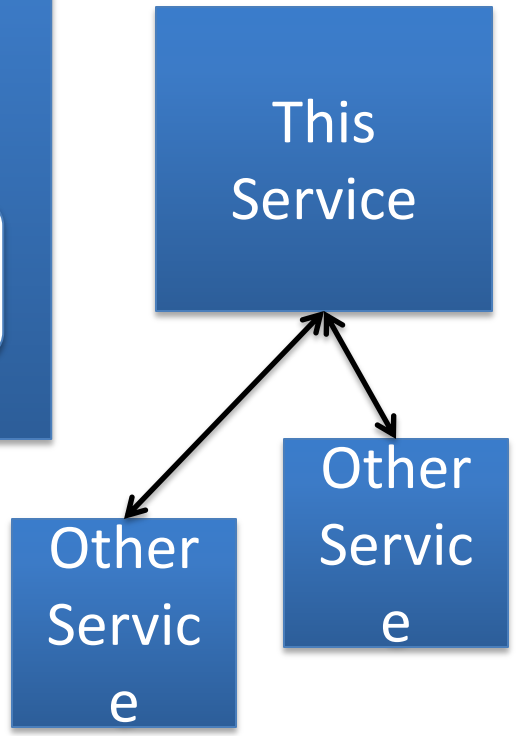
1. Unit Test within the service



2. Integration Test within the service



3. Test the service in context



Integration Testing Strategies

Incremental Strategy:

1. Lowest-level interfaces first
   (among Units within)
2. Work up through layers of interfaces
3. Highest-order interfaces last
   (e.g. API's, other systems, devices)

Why?

- Narrow focus in each test
- Build on validated interfaces

Functional Testing

Yes, this is the developer's job!

Functional Testing:

- Unit Test: Each Unit
- Integration Tests: Each combination
- Full system: Each function involved

"Done" means:

- Full confidence it works 100%
- Additional testing will find no issues

Regression Testing

Changed 1 line of code – I broke something!

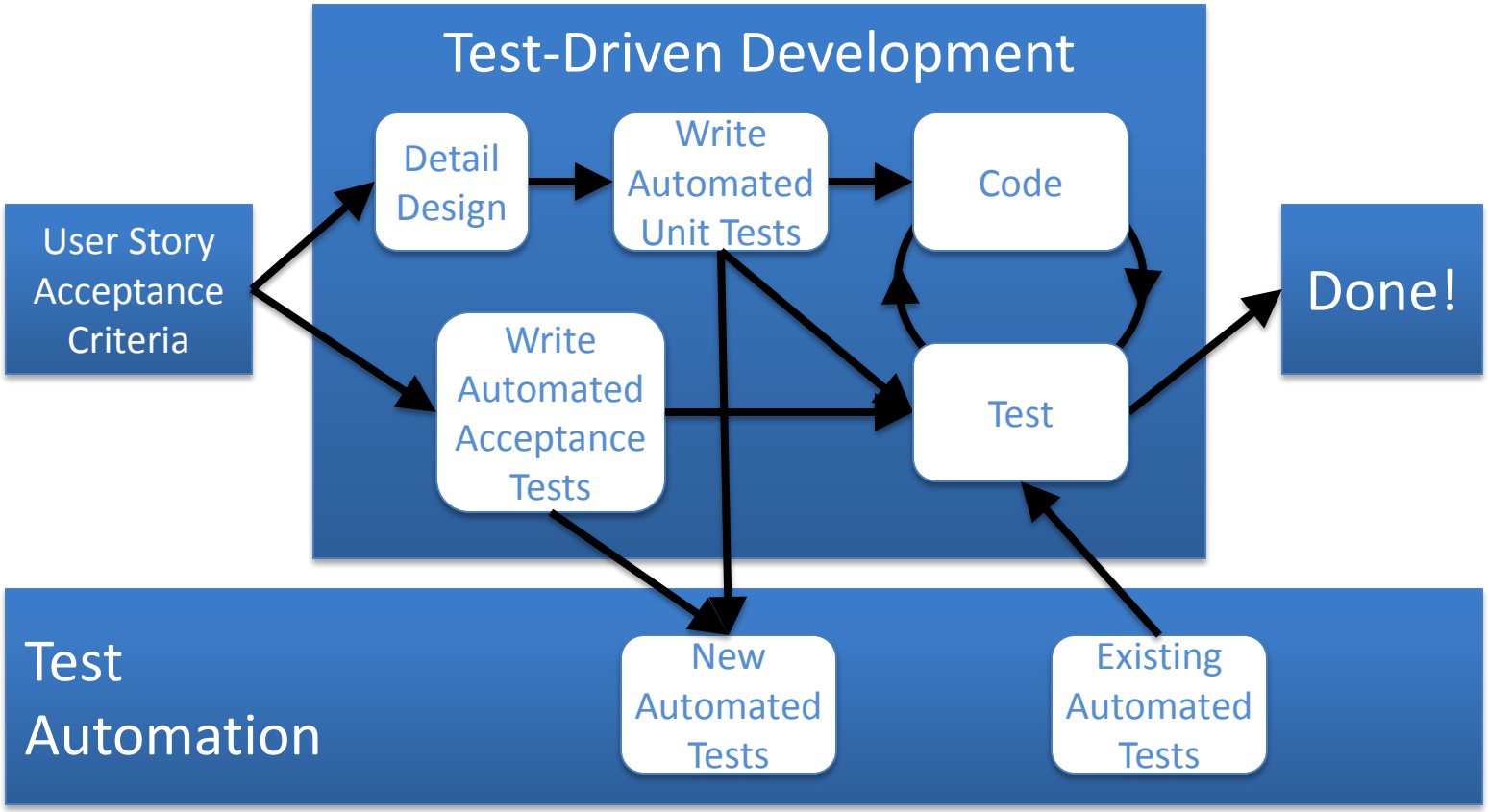
Regression Testing:

- Unit Test: Each Unit that was touched
- Integration Test: Each interface of each Unit touched
- Full system: Each function that may be affected

"Done" means:

- Full confidence there are no regressions
- Additional testing will find no issues

Regression Testing as an Integral Part of Coding



Testing and Developer Productivity

Do I have time for all this testing?

- How much of your time is rework?
- Testing is an Investment
- The Payback is Significantly reduced rework

Better testing = Higher productivity

QUESTIONS?