Test Driven Development

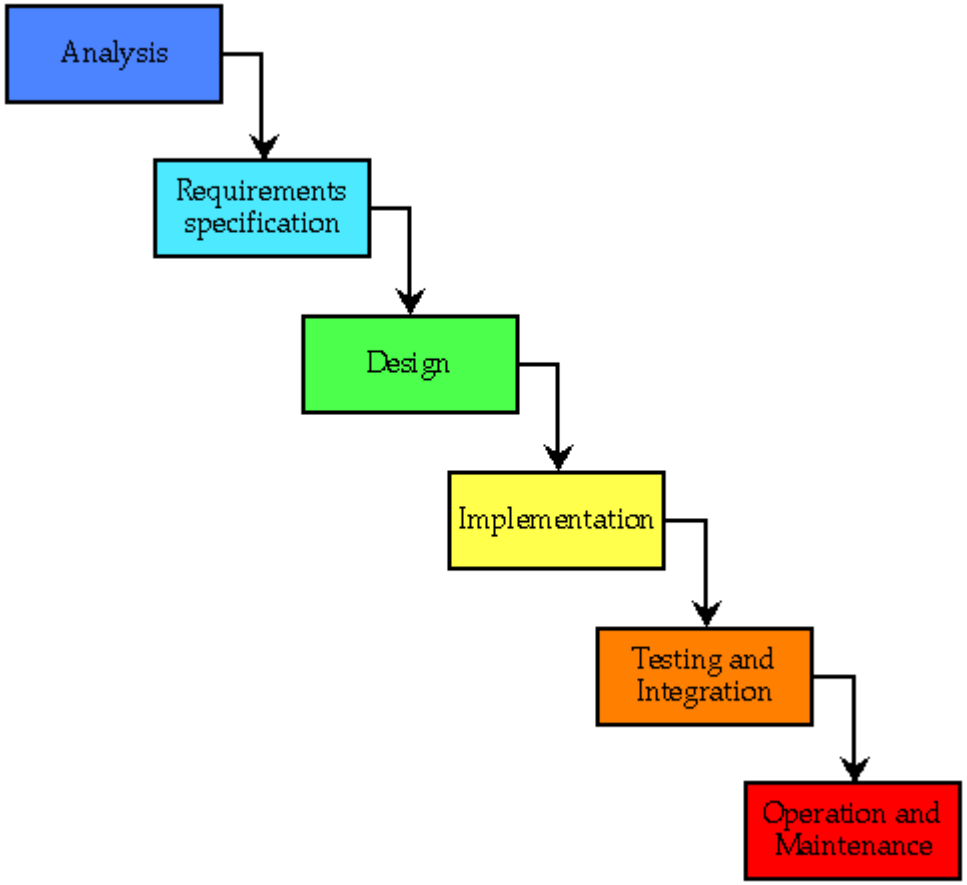
Satya Dodda
11/18/08

Agenda

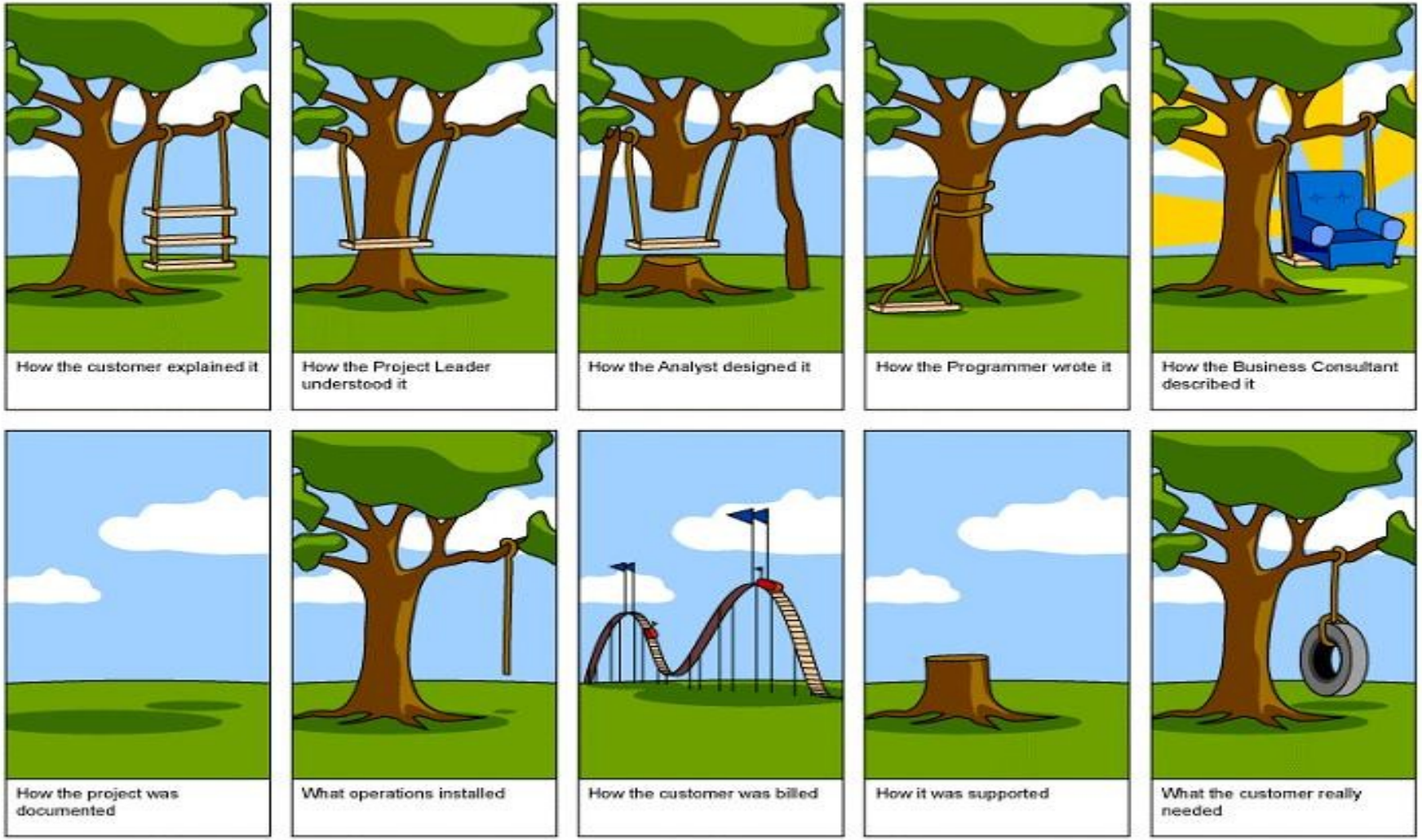
- Why Agile?
  - Waterfall Software Development
  - Agile Development
- Test Driven Development
  - Regular
  - Acceptance
- Key Learnings
  - New Testing Concepts
  - Best Practices
  - Test Tools

Why Agile?

Waterfall Software Development



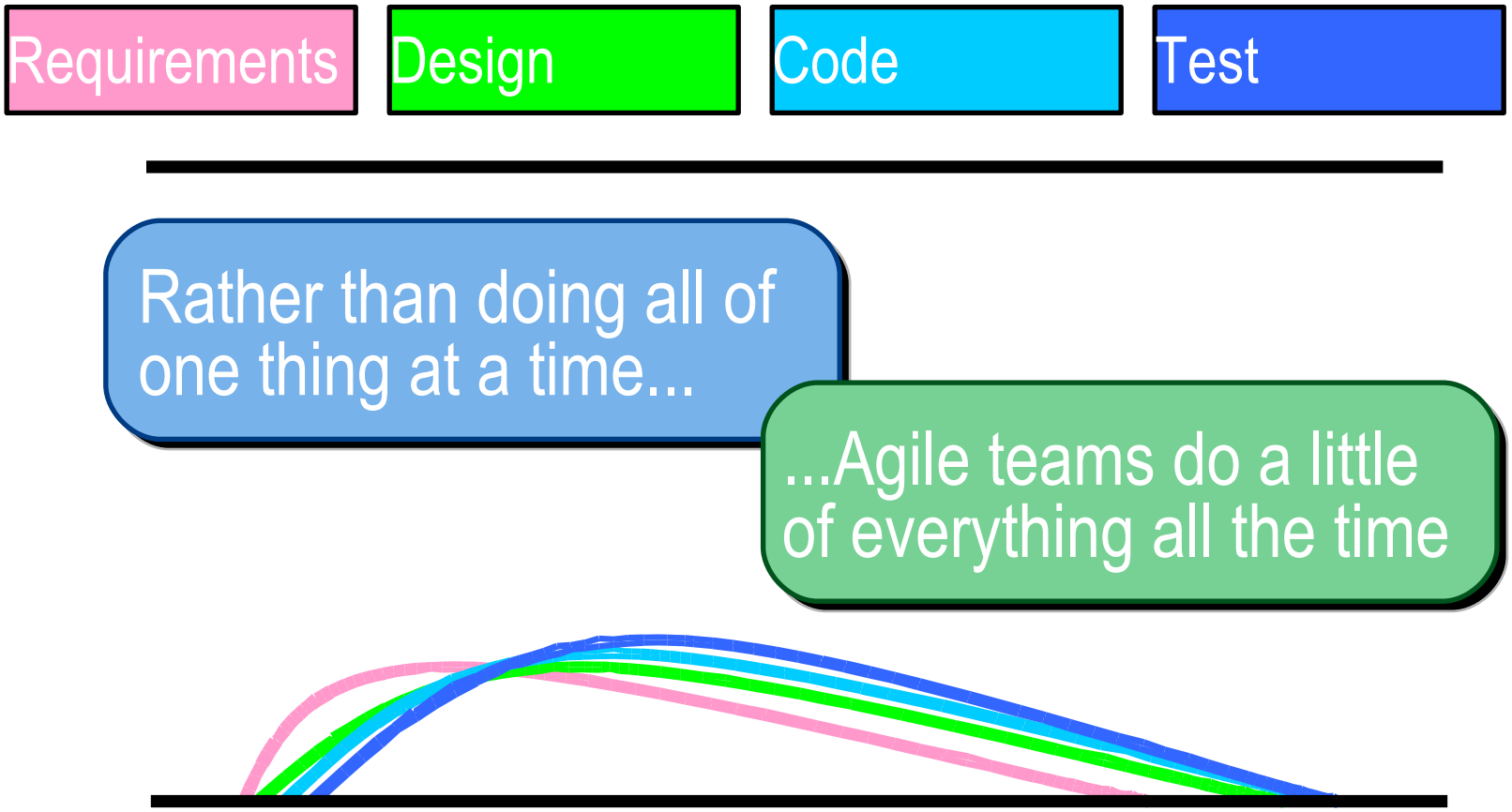
Issues


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Agile Development

- What is Agile?
  - An **iterative** and incremental (evolutionary) process approach to software development
  - Performed in a highly **collaborative** manner with '**just enough**' ceremony that produces high quality software which meets the **changing needs** of its stakeholders
  - Key Properties
    - Fail fast
    - Done
    - Don't blame requirements

Waterfall vs Agile



Agile Development Methods

- Test Driven Development (TDD)
- SCRUM
- XP
- Feature Driven Development
- Adoptive Software Development
- Dynamic Systems Development Method
- Pragmatic Programming

Before test driving...

- Bleeding edge technologies
- No single shoe fits all
- People make difference not processes/tools
- Adopting to change is difficult and some concepts are radical
- Open source is providing automated tool support for some key concepts.

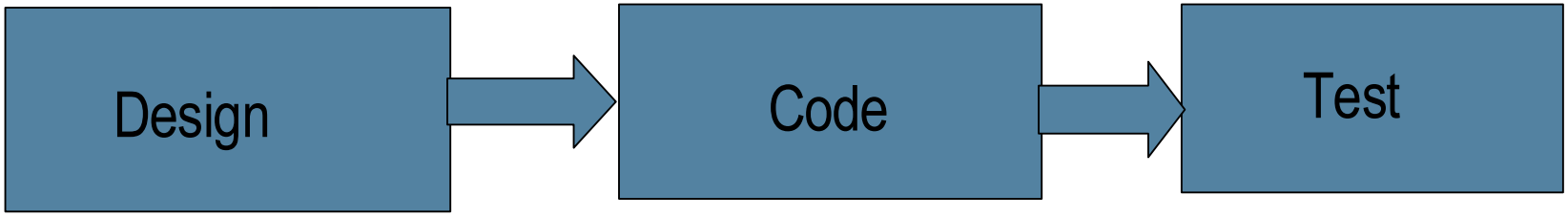
Test Driven Development

Test Driven Development

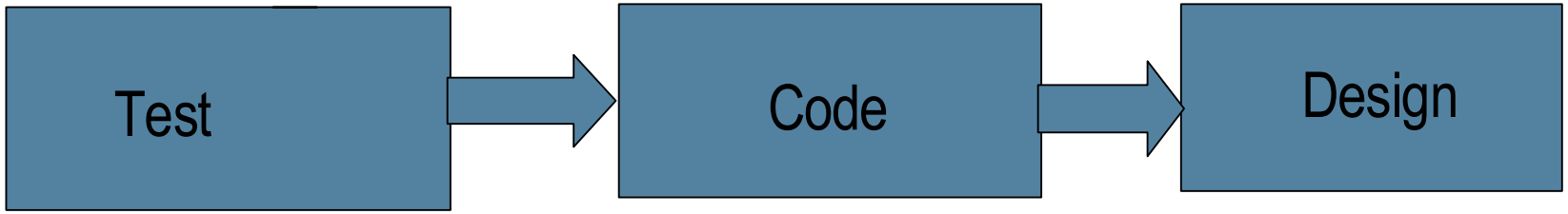
- (Regular) Test Driven Development (TDD)
  - How to build the thing right
  - Internal quality
    - Best design and easy maintenance

- Acceptance Test Driven Development (ATDD)
  - How to build the right thing
  - External quality
    - Correct features and functionality

Traditional Development Cycle



Test Driven Development Cycle



The mantra: Only ever write code to fix a failing test

Test Phase

- More than just writing a test
  - > It is designing the interface or API
- Write just enough test code to have a failing test
  - > Rather than a large test covering lots of parts
- Focus what is needed now
  - > Rather than over engineering
- Atomic and isolated

Code Phase

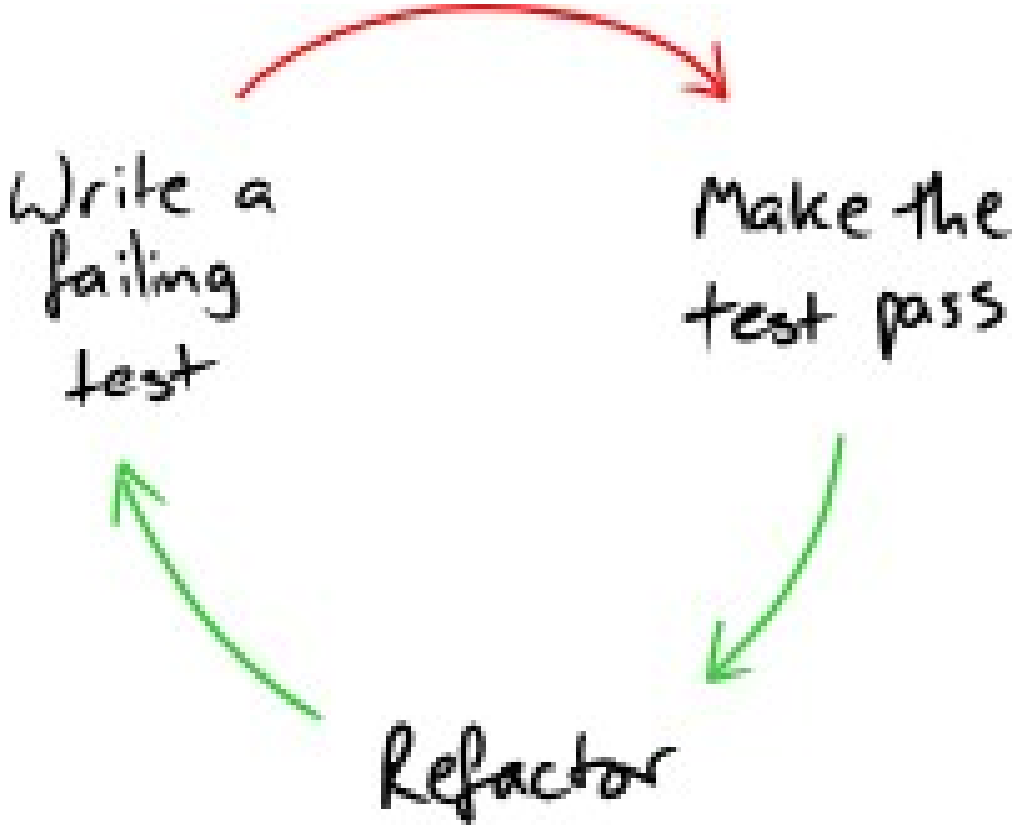
- Write just enough code to pass the test
- Meet requirements
- Show progress by making the test pass
- Testable code
- No more “90% done” syndrome

- Implementation may not be optimal

Refractor Phase

- A disciplined technique for restructuring an existing body of code, altering its internal structure without changing the external behavior
- Refactoring(verb) is about applying several small refactorings(noun)
- Refactorings include
  - Removing duplicate code
  - Applying OOP principles
  - Applying design patterns

Test Driven Development Process

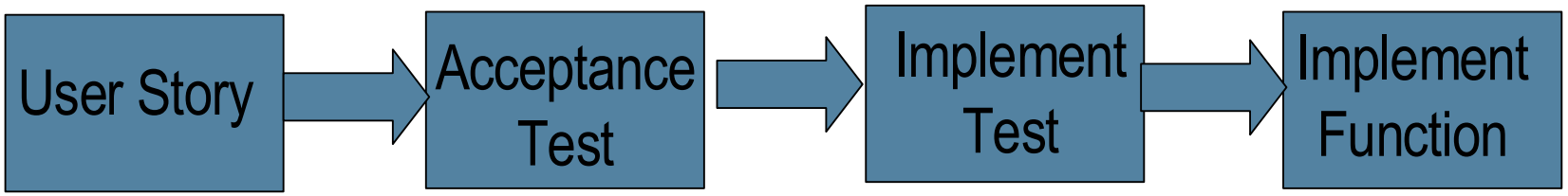


Test Code Refactor

or

Red Green Refactor

Acceptance Test Driven Development



User Stories

- An easy format to express requirements
  - Domain language
- States *who* does *what* and *why*
  - *Doctor sees patient history before starting conversation*
- Conveys what provides value to the customer, not how the system should provide that value
- Team collaboration
  - Customers, developers and testers

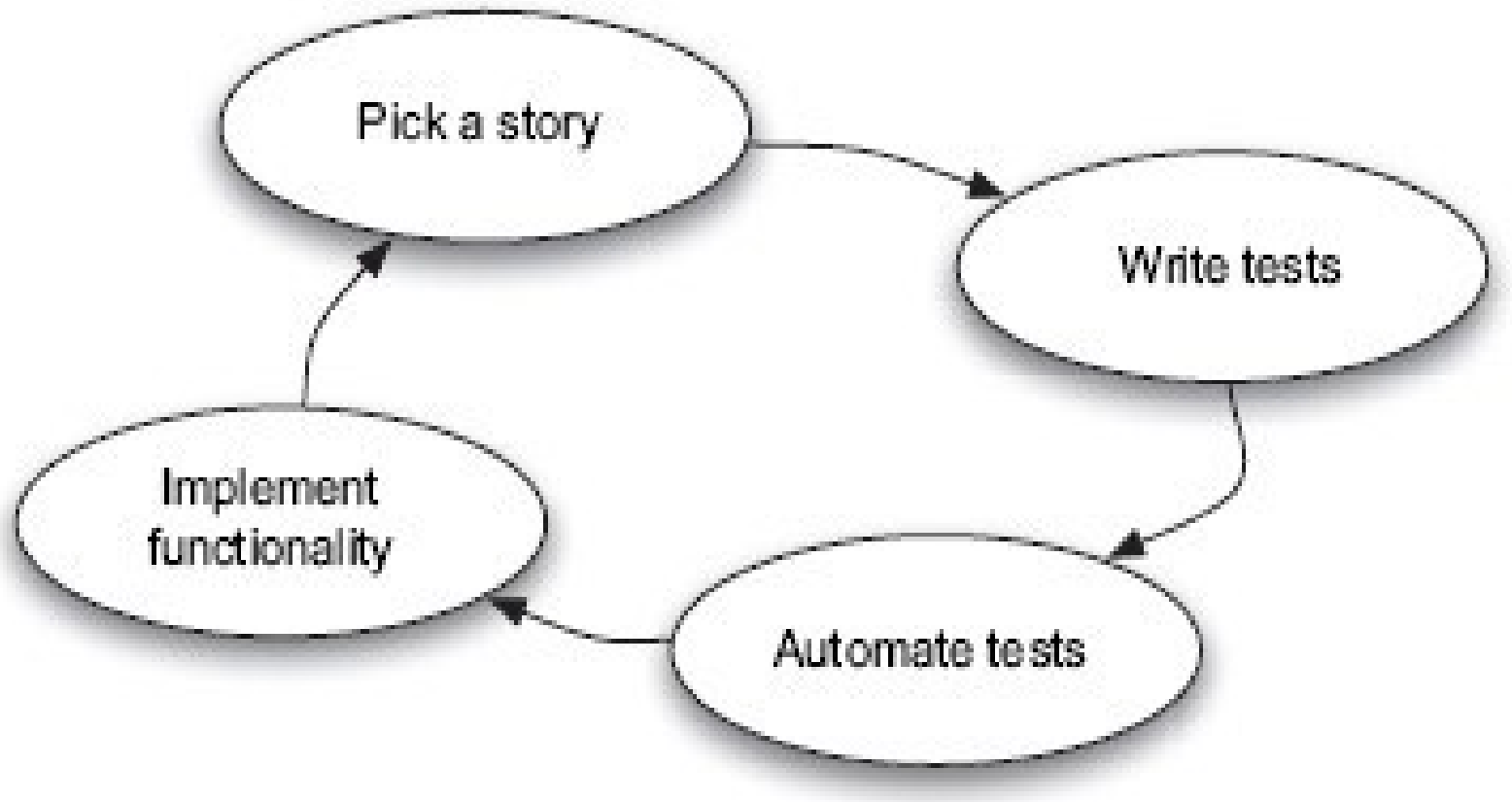
Acceptance Test Specifications

- Specifications for the desired behavior and functionality of a system
- Given a user story, describes how the system handles certain conditions and inputs
  - Show patient information for valid medical number
  - Show error message for invalid medical number
- Properties
  - Owned by customer
  - Written together with customer, developer, tester
  - Expressed in the language of the problem domain
  - Concise, precise, and unambiguous

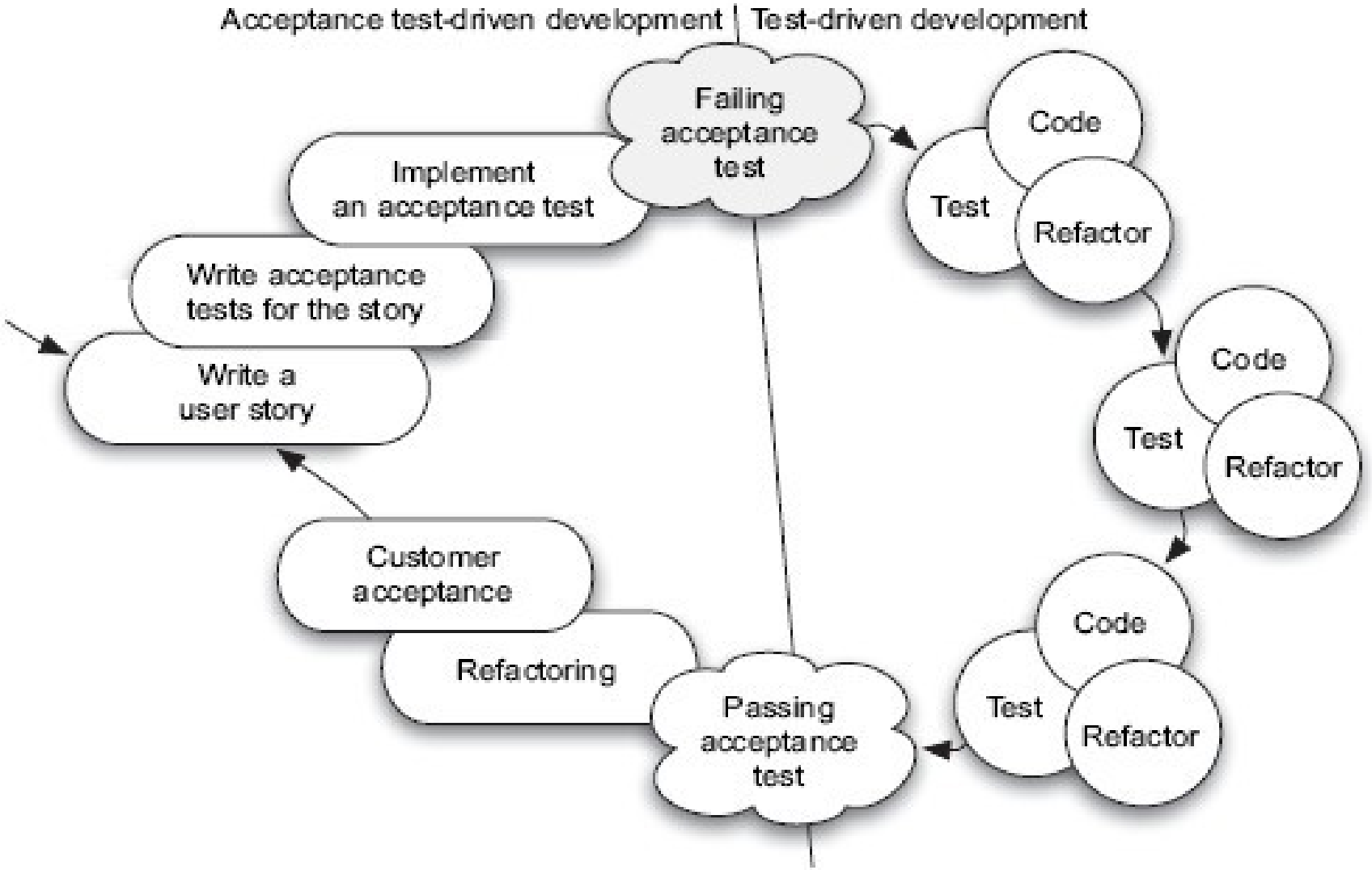
Implementation

- Tests
  - Similar to regular implementation of tests
  - Special emphasis
    - Can be implemented in a different language than the system under test
    - Declarative and tabular structure
    - Code chunks for columns/rows
- Code / functionality
  - Regular TDD
  - Test-code-refactor cycles

ATDD Process



Complete Picture



Key Learnings

Essential Testing Concepts

- Fixtures
- State and interaction based testing
- Test doubles
  - Stubs
  - Fakes
  - Mocks

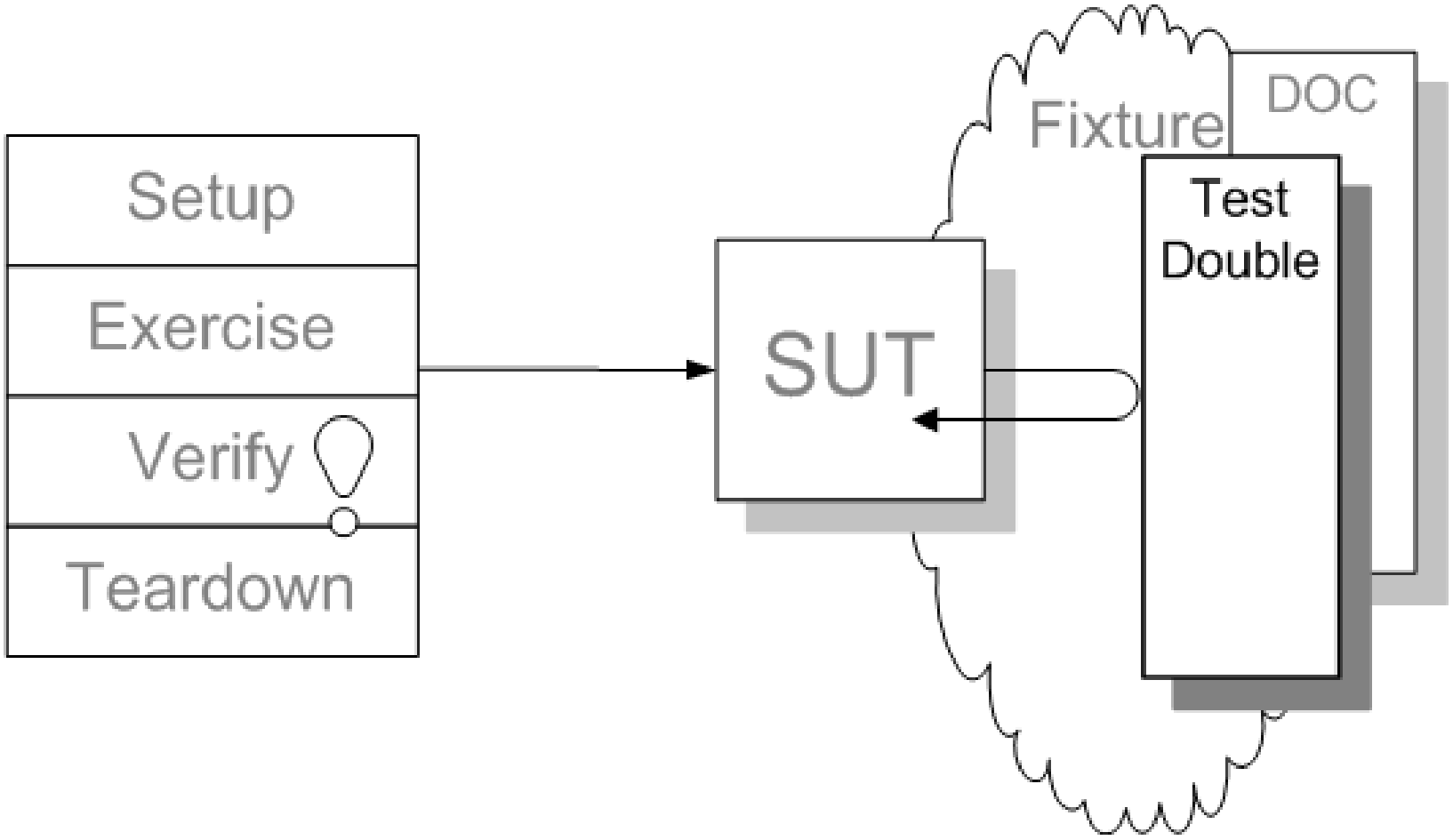
Fixtures

- A shared starting state from which all test cases of a test class begin their execution
  - Like “setup”
- Good design
- Lack duplication
- Allow focused test cases
- Modern test harnesses have great support
  - Junit's setup() Php's setup()

State and interaction based testing

- State based testing
  - Using state to determine pass or fail
  - Example: Adding a patient to a database results in non empty database
  - Traditional testing
- Interaction based testing
  - Using a sequence of method calls with appropriate parameters
  - Example: IsInRole() and AddPatient() methods are called with proper parameters
  - Agile testing

Test Doubles



Test Doubles

- Stubs
  - Typically return hard coded meaning-less values
- Fakes
  - Returns different values based on input
- Mocks
  - Records a sequence of method calls. Used to test complex SUTs like servlet container or EJB containers
  - Examples: EasyMock, jMock, and rMock.

Best Practices

- Remove duplicate tests
- Do not skip refactoring
- Happy path first
- Use Patterns
  - Parameterized, Self Shunt, etc.,
- Choose composition over inheritance
- Avoid static and singleton
- Isolate dependencies

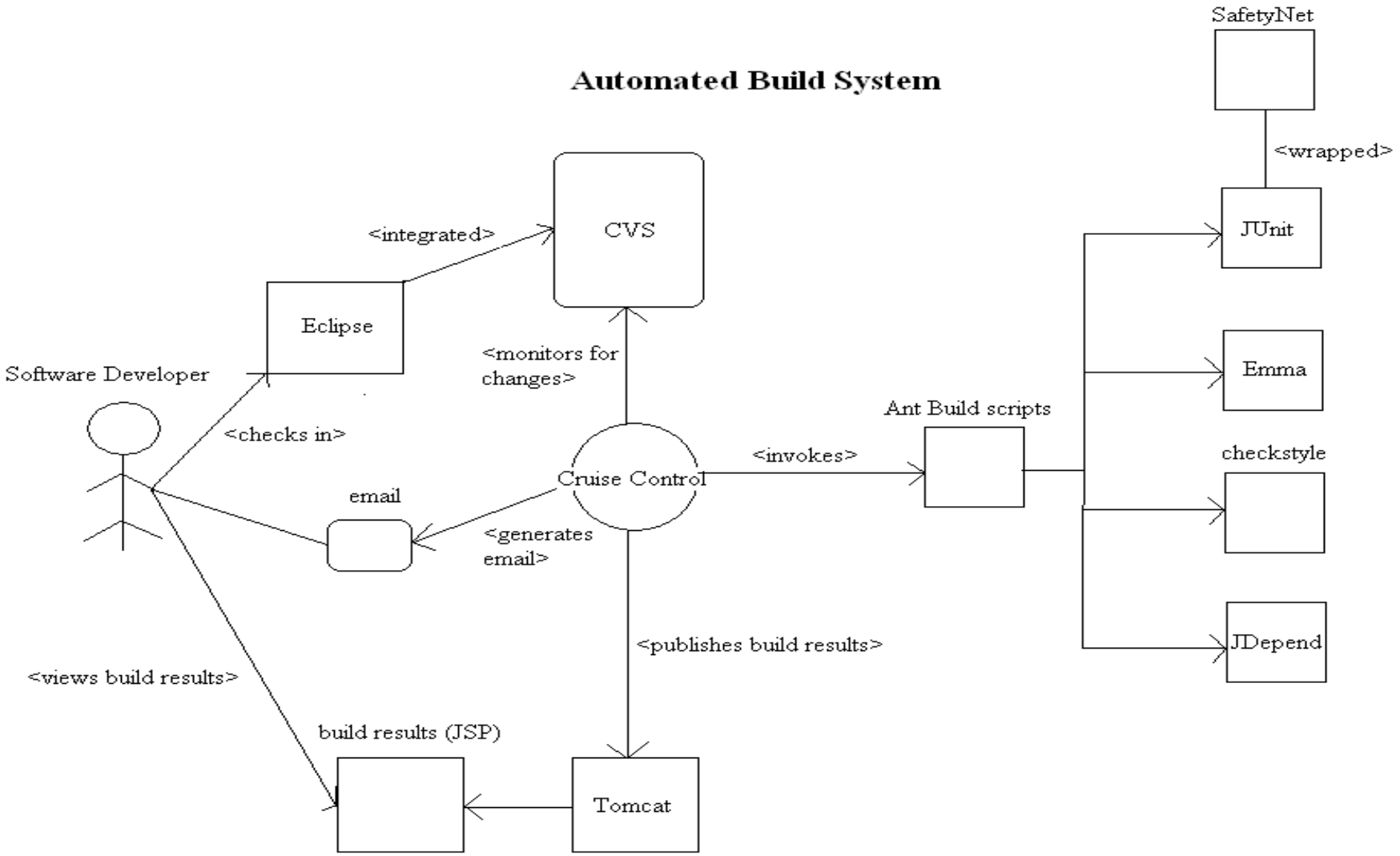
TDD Test Tools

- Unit testing
  - XUnit
  - JUnit
- Continuous integration and builds
  - Cruise-Control
  - Hudson
  - AntHill

ATDD Test Tools

- Table based test frameworks
  - Fit – Framework for integrated tests
    - Smart parsing for simple HTML tables
  - FitNesse – Fit in a wiki
    - A combination of Fit and wiki
  - Selinium
    - Controlling browser through HTML tables
- Text based test frameworks
  - Exactor
  - TextTest

Thank you!


Automated Build System
SafetyNet
<wrapped>
JUnit
Emma
checkstyle
JDepend
CVS
<integrated>
Eclipse
Software Developer
<checks in>
<monitors for changes>
Ant Build scripts
<invokes>
Cruise Control
email
<generates email>
<publishes build results>
<views build results>
build results (JSP)
Tomcat