Milestone 4

Milestone 4 – due next week

- An interview with one of the staff (Yogitha, Kyle, HeeChang, Umakant)

- Sign up for a posted time slot – this is outside lecture and lab times

- During the interview:
  - You will be asked several questions about your project
  - Interviewer will take notes
  - You will be asked to provide a working demo of your software showing the interviewer two features/functions that actually work

- Grading – Just show up, answer the questions and do the demo

Software Testing

So what is the real cost of software errors?

- Your software project is delivered late
- Your software project is delivered over budget
- Your software fails to meet user requirements
- Your customers lose confidence
- Your “brand” is tarnished
- If software is your product, people won’t want to buy or use your software
- If you use software to sell your product, people will buy from your competitors
- Repairing errors late in the development process costs more

Software Testing

So how can we adequately test our software?

Software Testing

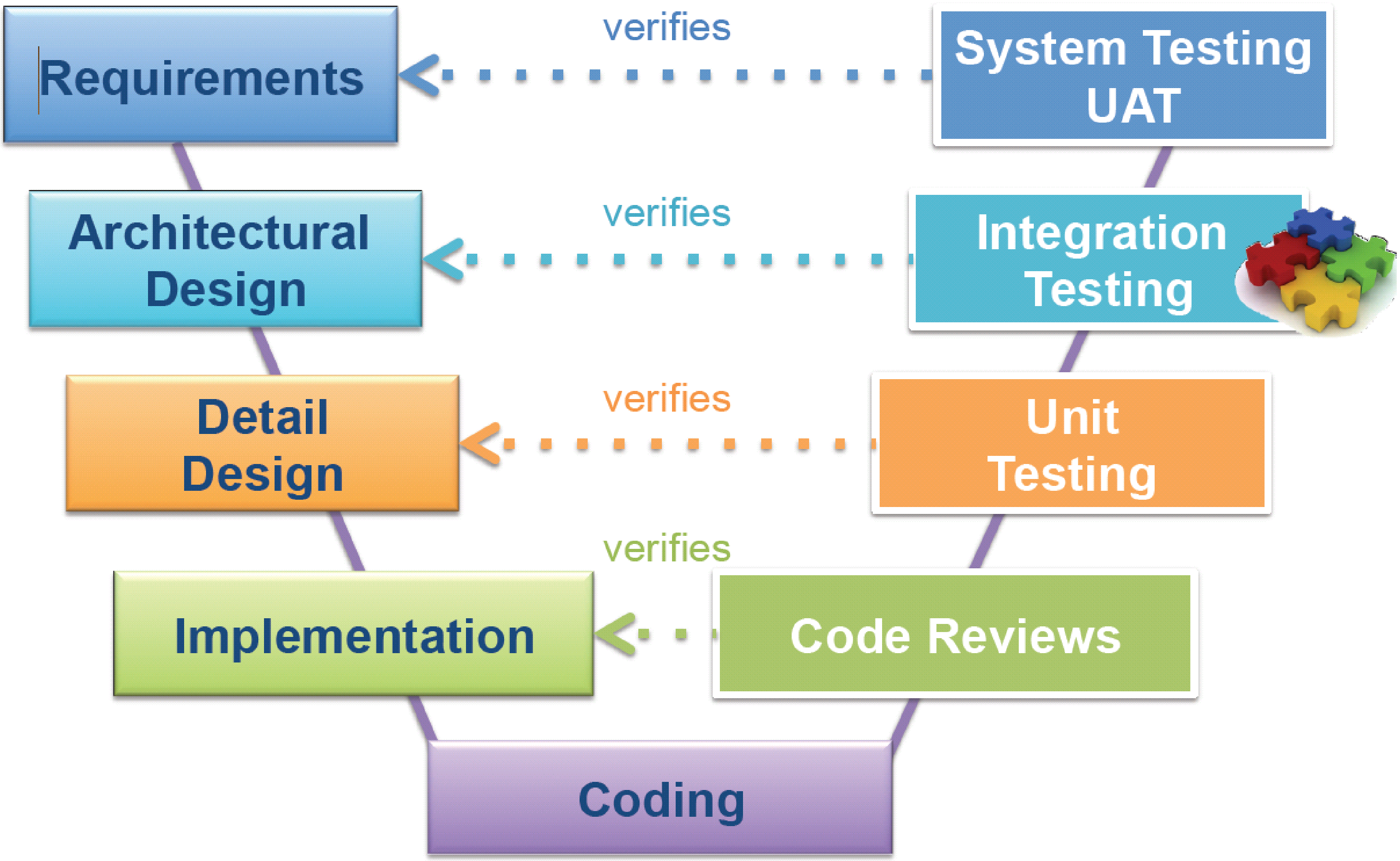
Different kinds of testing

- **Use Case Testing** – matching functionality to requirements
- **Load Testing** – how does it perform under stress?
- **Regression Testing** – How do I know my changes didn't break something that was working before?
- **User Acceptance Testing** - Does it do what they expect? Can they use it?
- **System Testing** – Combining many working components to see if they work together.
- **Unit Testing** – Does this piece of code do what it is written to do?
- **Integration Testing** – Can I add a module to existing, working software without breaking it?
- **Black Box** Testing – Testing its features without peering into its structures/code
- **White Box** Testing – Testing code/tructures without looking at the larger picture of how it meets functional requirements for the larger system.

Software Testing

Different Kinds of Testing Require
Different Testing Tools and Techniques at
Different Times in the
Software Development Life Cycle

Software Testing



Requirements ← verifies ← System Testing UAT

Architectural Design ← verifies ← Integration Testing

Detail Design ← verifies ← Unit Testing

Implementation ← verifies ← Code Reviews

Coding

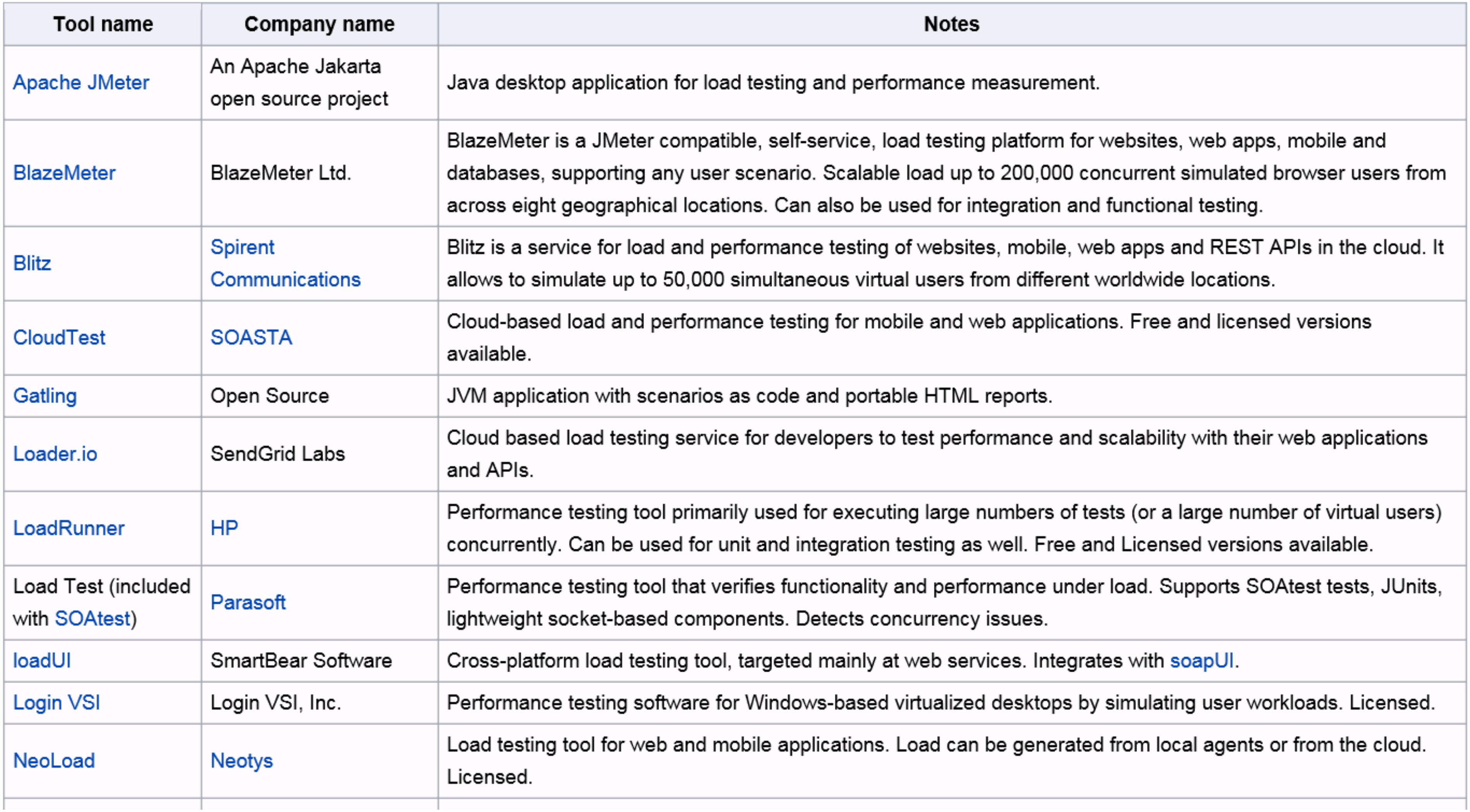
Software Testing

Performance/Load/Stress Testing

Objectives

- Put the system under stress by simulating increasing volumes
- May use simulation software to create traffic/volumes

Software Testing



Tool name	Company name	Notes
Apache JMeter	An Apache Jakarta open source project	Java desktop application for load testing and performance measurement.
BlazeMeter	BlazeMeter Ltd.	BlazeMeter is a JMeter compatible, self-service, load testing platform for websites, web apps, mobile and databases, supporting any user scenario. Scalable load up to 200,000 concurrent simulated browser users from across eight geographical locations. Can also be used for integration and functional testing.
Blitz	Spirent Communications	Blitz is a service for load and performance testing of websites, mobile, web apps and REST APIs in the cloud. It allows to simulate up to 50,000 simultaneous virtual users from different worldwide locations.
CloudTest	SOASTA	Cloud-based load and performance testing for mobile and web applications. Free and licensed versions available.
Gatling	Open Source	JVM application with scenarios as code and portable HTML reports.
Loader.io	SendGrid Labs	Cloud based load testing service for developers to test performance and scalability with their web applications and APIs.
LoadRunner	HP	Performance testing tool primarily used for executing large numbers of tests (or a large number of virtual users) concurrently. Can be used for unit and integration testing as well. Free and Licensed versions available.
Load Test (included with SOAtest)	Parasoft	Performance testing tool that verifies functionality and performance under load. Supports SOAtest tests, JUnits, lightweight socket-based components. Detects concurrency issues.
loadUI	SmartBear Software	Cross-platform load testing tool, targeted mainly at web services. Integrates with soapUI.
Login VSI	Login VSI, Inc.	Performance testing software for Windows-based virtualized desktops by simulating user workloads. Licensed.
NeoLoad	Neotys	Load testing tool for web and mobile applications. Load can be generated from local agents or from the cloud. Licensed.

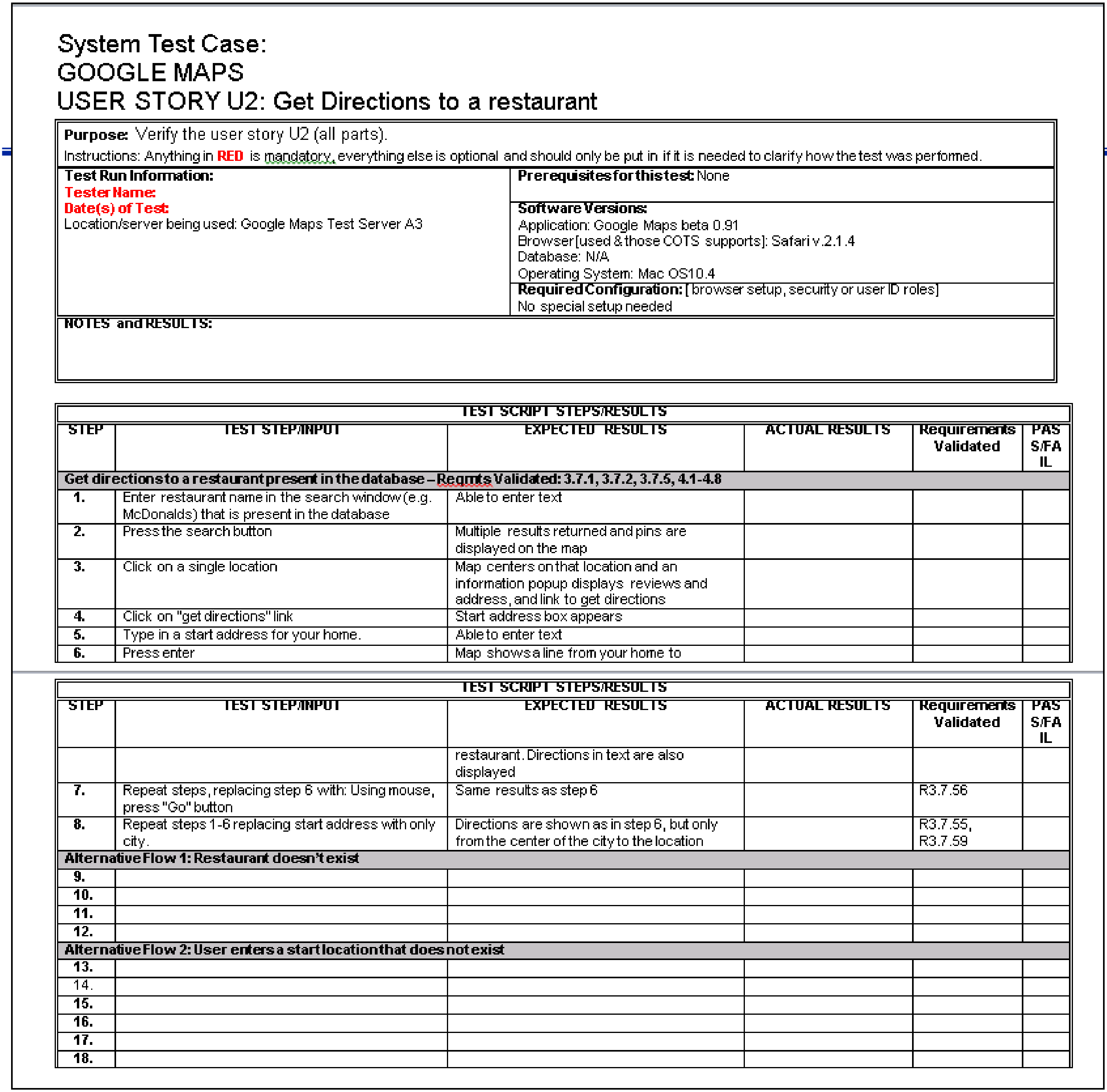
Software Testing

UAT – User Acceptance Testing

Objectives

- Confirm that the system meets requirements
- Identify, document and resolve any discrepancies
- Determine readiness for code/module/system deployment to production
- Can be a set of steps to ensure functionality and/or a series of subjective questions
- Each test is planned and documented according to requirements
- Each test execution is documented
- Any bugs are passed back to the developers for re-work
- Very time consuming, but very necessary

Testing



System Test Case:
GOOGLE MAPS
USER STORY U2: Get Directions to a restaurant

Purpose: Verify the user story U2 (all parts).

Instructions: Anything in **RED** is mandatory, everything else is optional and should only be put in if it is needed to clarify how the test was performed.

Test Run Information: **Tester Name:** **Date(s) of Test:** Location/server being used: Google Maps Test Server A3	**Prerequisites for this test:** None
	Software Versions: Application: Google Maps beta 0.91 Browser [used & those COTS supports]: Safari v.2.1.4 Database: N/A Operating System: Mac OS10.4
	Required Configuration: [browser setup, security or user ID roles] No special setup needed

NOTES and RESULTS:

TEST SCRIPT STEPS/RESULTS

STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	Requirements Validated	PASS/FAIL
Get directions to a restaurant present in the database – Reqmts Validated: 3.7.1, 3.7.2, 3.7.5, 4.1-4.8					
1.	Enter restaurant name in the search window (e.g. McDonalds) that is present in the database	Able to enter text			
2.	Press the search button	Multiple results returned and pins are displayed on the map			
3.	Click on a single location	Map centers on that location and an information popup displays reviews and address, and link to get directions			
4.	Click on "get directions" link	Start address box appears			
5.	Type in a start address for your home.	Able to enter text			
6.	Press enter	Map shows a line from your home to restaurant. Directions in text are also displayed			
7.	Repeat steps, replacing step 6 with: Using mouse, press "Go" button	Same results as step 6		R3.7.56	
8.	Repeat steps 1-6 replacing start address with only city.	Directions are shown as in step 6, but only from the center of the city to the location		R3.7.55, R3.7.59	
Alternative Flow 1: Restaurant doesn't exist					
9.					
10.					
11.					
12.					
Alternative Flow 2: User enters a start location that does not exist					
13.					
14.					
15.					
16.					
17.					
18.					

Software Testing

The Ideal User Acceptance Tester:

Background

- Understands the user requirements

Skills

- Good Communicator
- Understands the System
- Technical OR Non-Technical
- Attention to detail

Availability

- Fully Dedicated to Conducting the Tests, Documenting Results

UAT – Deliverables

Test plan - Outlines the testing strategy:

- Acceptance Testing
- Entry and Fail Criteria
- Test Execution Team
- Test Script Developer

Test cases - Guide the team to effectively test the application

Test Log - records all the test cases executed and the actual results

User Sign Off - indicates that the customers are satisfied with the product.

Software Testing

Unit Testing

- Ensure that every line of code does exactly what it is supposed to do.
- Change anything? You must do ***regression testing*** to make sure you didn't break anything.
- A "unit" = the smallest possible unit of code behavior that can be tested in isolation.
- Test the code for
  - Handling expected inputs properly
  - Handling unexpected inputs properly
  - Graceful error handling
  - Edge case: handling the extremes
    - 20 bytes in a 20-byte field
    - 21 bytes in a 20-byte field
    - Character data in a numeric field
    - Divide by zero

Why Automate Unit Testing?

- **Speeds up Unit Testing**
- **Enables Speedy, Reliable Regression Testing**
- **Ensures that Function meets Design**

Software Testing

How to Automate Unit Testing?

Use a “Testing Framework” to develop some testing software that will do this:

For a given action in your code:

- **Determine the expected value** (the value which should be produced if the software is working correctly)
- **Determine the actual value** (the value which the software is actually computing)
- **Compare the two**:
  - If they agree, the test passes
  - If they disagree, the test fails

Automated Unit Testing Frameworks:

- JUnit Tutorial: http://clarkware.com/articles/JUnitPrimer.html
- PyUnit: http://wiki.python.org/moin/PyUnit
- Python’s unittest https://docs.python.org/3/library/unittest.html
- PhpUnit: https://phpunit.de/
- Unit testing with C#: http://www.csunit.org/tutorials/tutorial7/
- Unit testing in Objective-C and Xcode:

http://developer.apple.com/mac/articles/tools/unittestingwithxcode3.html

- Unit testing for Ruby:

http://www.ruby-doc.org/stdlib/libdoc/test/unit/rdoc/classes/Test/Unit.html

Software Testing

Refactoring:

Code refactoring is the process of restructuring existing computer code without changing its external behavior.

Refactoring improves nonfunctional attributes of the software:

- Make it more Readable
- Make it Concise: Smallest possible size
- Reduces Complexity
- Improves Maintainability

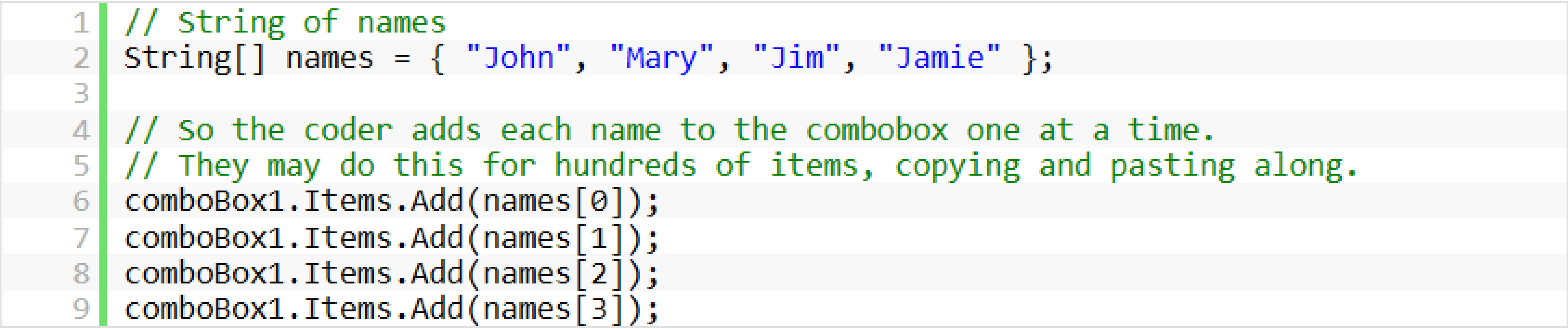
Software Testing

Refactoring:

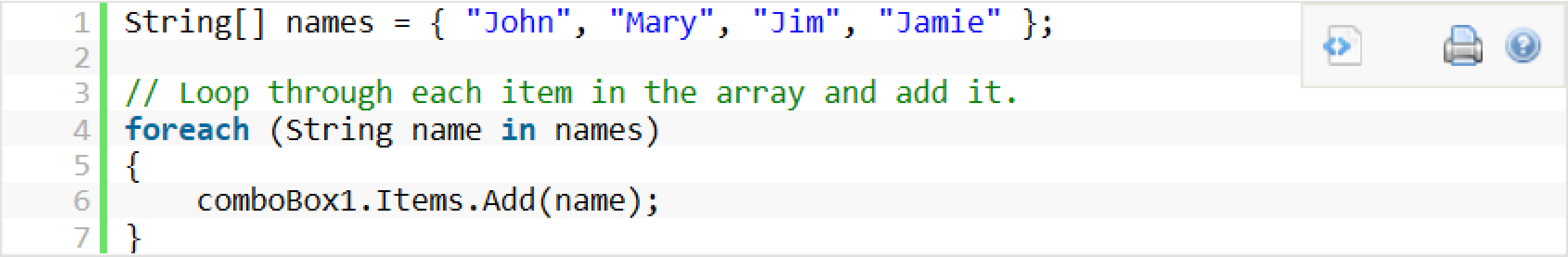
You must ensure that you have solid, repeatable, automated unit tests to ensure that refactoring does NOT impact code functionality AT ALL.

Refactoring Tips

Tip 1 – Look for multiple lines virtually doing the same thing



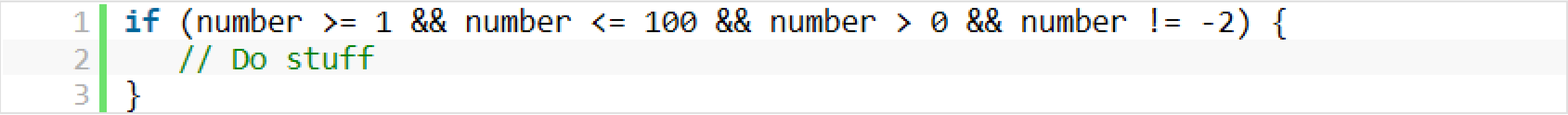
```
// String of names
String[] names = { "John", "Mary", "Jim", "Jamie" };

// So the coder adds each name to the combobox one at a time.
// They may do this for hundreds of items, copying and pasting along.
comboBox1.Items.Add(names[0]);
comboBox1.Items.Add(names[1]);
comboBox1.Items.Add(names[2]);
comboBox1.Items.Add(names[3]);
```



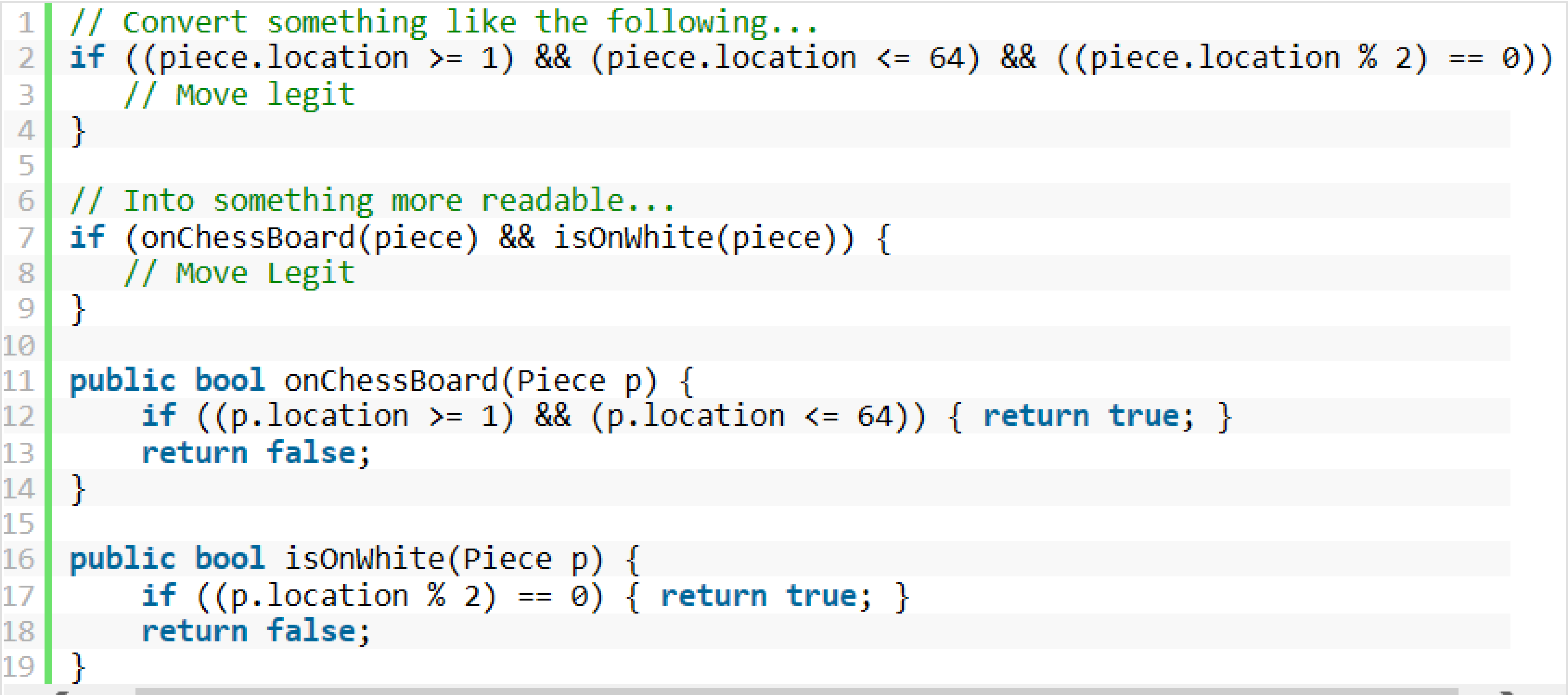
```
String[] names = { "John", "Mary", "Jim", "Jamie" };

// Loop through each item in the array and add it.
foreach (String name in names)
{
    comboBox1.Items.Add(name);
}
```

Refactoring Tips & Examples

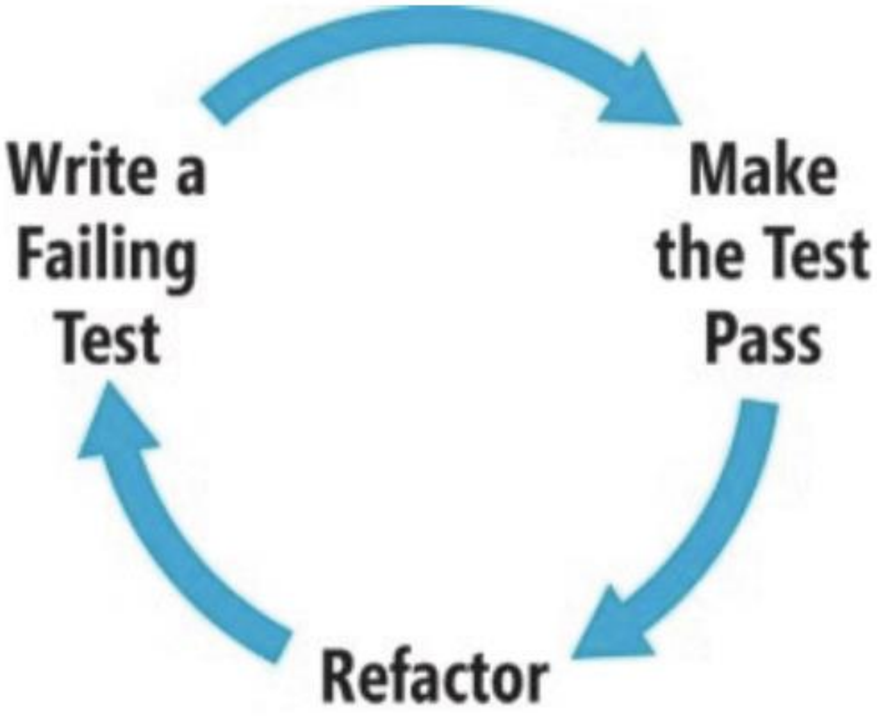
Tip 2 – Cut down complex conditionals



```
if (number >= 1 && number <= 100 && number > 0 && number != -2) {
    // Do stuff
}
```



```
// Convert something like the following...
if ((piece.location >= 1) && (piece.location <= 64) && ((piece.location % 2) == 0))
    // Move legit
}

// Into something more readable...
if (onChessBoard(piece) && isOnWhite(piece)) {
    // Move Legit
}

public bool onChessBoard(Piece p) {
    if ((p.location >= 1) && (p.location <= 64)) { return true; }
    return false;
}

public bool isOnWhite(Piece p) {
    if ((p.location % 2) == 0) { return true; }
    return false;
}
```



TDD

Software Testing

- Traditionally (i.e. “waterfall”) large systems are designed and coded up front, then tested by QA teams when coding is done.

Test Driven Development “TDD”

- TDD is a developer process of writing unit-tests first, then writing code to pass the tests.
- Tests are executable requirements/specifications
- End-result is a complete system (working code) with corresponding, automated unit tests
  - Assist with regression testing
  - Enable refactoring
- NO CODE is ever written without a test being created first

Software Testing

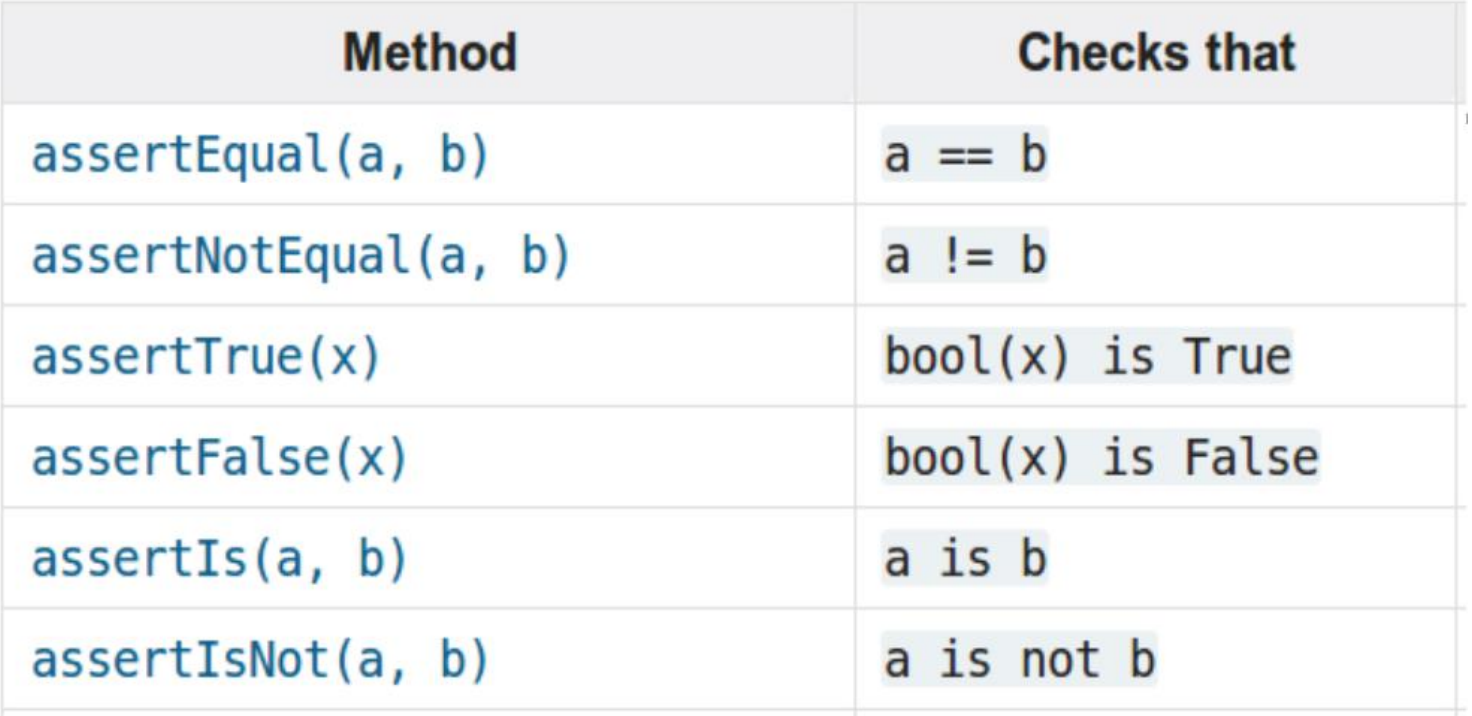
TDD

- *First* write the test, *then* do the design/implementation
- Part of agile approaches like XP (Extreme Programming)
- Supported by testing framework tools (like Junit, PyUnit)
- TDD Is more than a mere testing technique; it incorporates much of the detail design work
- Useful for many code behaviors, but not really for GUI or Database functionality
- It is not a replacement for UA, System, Performance testing

Software Testing

TDD

- Assertions: a method that allows verification of ACTUAL results versus EXPECTED results



Method	Checks that
`assertEqual(a, b)`	`a == b`
`assertNotEqual(a, b)`	`a != b`
`assertTrue(x)`	`bool(x) is True`
`assertFalse(x)`	`bool(x) is False`
`assertIs(a, b)`	`a is b`
`assertIsNot(a, b)`	`a is not b`

Software Testing

TDD Step-By-Step

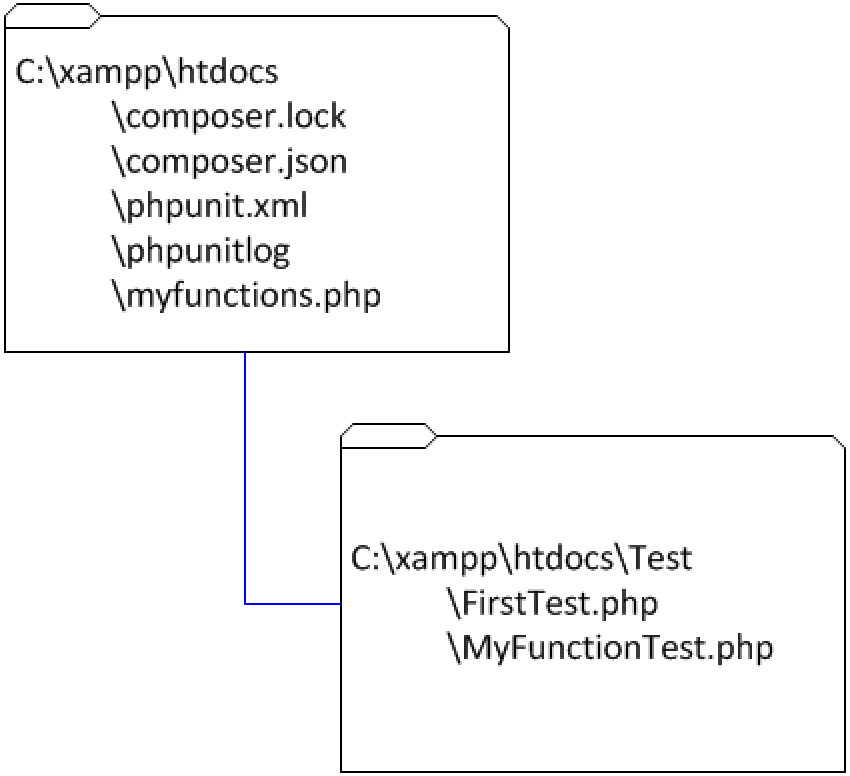
1. Write a single test
2. Run that test, system fails *(because the actual code is not written yet)*
3. Write a “stub” of the code function
4. Run that test, system fails *(because the stub doesn’t do anything)*
5. Fill in the stub with code to make the test pass
6. Run the test again, and verify that the code runs properly
7. Refactor the code as needed to improve its design
8. Run the test again to ensure clean regression test

Let’s Look at an Example:

- Using “phpunit” in my XAMPP environment on my PC
  (Apache, MySQL, Php)
- I used Composer to Install phpunit from https://phpunit.de/
- It runs from within my c:\xampp\htdocs folder
  (from where apache executes programs for the localhost web server)
- I tell phpunit what is my expected result, and it compares that to the actual result of executing my code
- Uses an “assertion” to measure the outcome
  - `assertEquals(expected result, actual result);`

Software Testing

Here’s my setup:



```
C:\xampp\htdocs
        \composer.lock
        \composer.json
        \phpunit.xml
        \phpunitlog
        \myfunctions.php
```

```
C:\xampp\htdocs\Test
        \FirstTest.php
        \MyFunctionTest.php
```

Software Testing

C:\xampp\htdocs

- \composer.lock – composer config info
- \composer.json – composer metadata
- \phpunit.xml – phpunit config info
- \phpunitlog – we write the output of the phpunit tests here
- \myfunctions.php – a php program defining functions used in testing

C:\xampp\htdocs\Test

- \FirstTest.php – simple boolean true/false
- \MyFunctionTest.php – simple math A + B

Software Testing

myfunctions.php

```
<?php
    function my_addition($arg1, $arg2){
        return $arg1 + $arg2;
    }
?>
```

FirstTest.php

```
<?php # FirstTest.php

class FirstTest extends \Phpunit_Framework_TestCase
         {
                 public function testUselessness()
                          {
                                  $this->assertTrue(true);
                          }
         }

?>
```

Software Testing

myfunctiontest.php

```
<?php

class MyProceduralTest extends \Phpunit_Framework_TestCase {

    /*
     * Testing the addition function
     */

    public function testAddition(){
         include('my_functions.php');
         $result = my_addition(1,1);
         $this->assertEquals(2, $result);
    }
}
?>
```

Software Testing

1. Edit the code to enter my expected result
2. Invoke phpunit, with option to write results to log file

For this week’s lab, you will do basically the same thing, but using a different framework – “Python3UnitTest” -- and running in your lab VM environment.