The Need for Test Automation

- We have examined a variety of ways to generate test cases, and in this lecture we will examine efficient ways to manage the testing process.
- Testing is the last stage of software development, and hence suffers the greatest impact from time and cost blowouts.
- Testing is essential, and when done properly may consume as many resources as the other activities put together.
- While the rigor of the test process should not be compromised, planning and automation can help manage the time and cost burden.

Software processes

- All modern software processes are iterative.
- The early a defect is discovered, the easier it is to fix
- Test early and test often!
- Tests need to repeated each iteration, and should be documented and tracked.
- The more often a test is repeated, the greater the saving from test automation.
- Large systems may accumulate many different tests cases, and complex test scenarios. It is not uncommon for full testing to take days.

The V-Model

Level of Detail

- Requirements
- Specifications
- functional requirements
- Software requirements
- Customer requirements specification

High

Problem with V-Model: Client’s Perception is the same as the Developer’s Perception

Acceptance Testing

System Testing

Integration Testing

Object Design

Unit Testing

Design

Analysis

System Time

Project Time

Automating unit tests

- Unit tests test individual functions within a single. They have the least dependencies, and the most stable environment.
- Unit tests are the easiest and most cost effective tests to automate.
- Unit test cases can be derived using the white and black box methods we have examined.
- Automating unit tests can be done through an ad-hoc test harness (as in the assignment), but it is better to use a test manager that can be integrated into an IDE.
- We will consider JUnit.
Example: JUnit

- JUnit is a java unit testing environment written by Kent Beck.
- It is intended to support Test Driven Development.
- The test script is a java file annotated with @Before, @After and @Test.
- It uses statements assertEquals, assertTrue, and a few others to ensure the unit of functionality behaves as specified.
- It can be integrated into the development environment to give developers instant feedback on the implementation.

JUnit: setup, test, teardown

```java
import org.junit.*;
import static org.junit.Assert.*;

class SampleTest {
    private java.util.List<String> emptyList;

    /**
     * Sets up the test fixture.
     * (Called before every test case method.)
     */
    @Before
    public void setUp() {
        emptyList = new java.util.ArrayList<String>();
    }

    /**
     * Tears down the test fixture.
     * (Called after every test case method.)
     */
    @After
    public void tearDown() {
        emptyList = null;
    }

    /**
     * Tests a method.
     * (Called once before every test method.)
     */
    @Test
    public void testSomeBehavior() {
        assertEquals("List should be empty", 0, emptyList.size());
    }

    /**
     * Tests an exception.
     * (Called once after every test method.)
     */
    @Test(expected=IndexOutOfBoundsException.class)
    public void testForException() {
        Object o = emptyList.get(0);
    }
}
```

Running JUnit

- JUnit may be used in an integrated development environment like Eclipse.
- JUnit may be run from the command line java org.junit.runner.JUnitCore Test
- JUnit may be integrated into ant build files - ant use an xml format to specify compilation, testing, documentation, and building of software.

Integration Testing

- Once the individual components have passed the unit tests, it is necessary to ensure that the components and modules work together.
- "Big bang testing" waits until all components are ready and then tests everything at once. This is not recommended (why?)
- double tests test 2 components at a time, triple tests test 3 and so on.
- Test drivers and test stubs can allow for incremental testing of components.

System Testing

System testing contains several subtypes:
- Functional testing – tests functional requirements
- Performance testing – tests non-functional requirements
- Pilot testing – tests common functionality among end users
- Acceptance testing – performed by customer against acceptance criteria
- Installation testing is performed with in the target environment
The last three are not suitable for automation.
Functional Testing

- The functional requirements can be extracted from the RAD, and a set of test cases may be developed using Heumann’s Use Case method.
- The cases can be determined early, and tests scripts can be prepared. The tests will probably involve interaction with GUI so GUI scripting, or record and playback may be required.
- Sikuli is an example of a GUI scripting language.
- Automation and execution of these tests can be very time consuming.

Performance testing

- Performance testing is done against the SDD to ensure that some non-functional requirements are met.
- Give some non-functional requirement tests that can be automated, and some which should not be automated.

More Aspects of Testing

- Test Plans – A test plan needs to be maintained throughout the software project to organize tests
- Test Manuals – A test manual records each test case, its expected returns, its preconditions and its dependencies
- Test Prioritization – Often projects accumulate more test cases than they have time to run. Test Case Prioritization is an important activity to select the cases most likely to reveal a defect.
- Regression Testing – Every iteration, we must ensure that new modifications have not caused a passed test to now fail.
- Test quality Metrics – It is important to assess how effective a set of test cases are.

Testing Automation Products

There are many commercial products that provide scripting languages for automated test cases, organize test cases and present and track the outcomes of tests.

Some of the products are:
- HP QuickTest – part of the HP Quality Center Suit. Uses VB scripting.
- IBM Rational Functional Tester – Part of IBM rational environment.
- SilkTest – For web applications. Has record and playback features