CSC 405 Lecture Notes Week 3

Overview of Software Testing Concepts

I. Goals for 405 lecture and lab in weeks 3 and 4.

A. In lecture, cover conceptual and theoretical details of software testing.

B. In lab, come up with testing framework(s) and tools to use for testing our projects.

II. A bit of testing motivation.

A. Last decade has seen highly significant shift in the industry mindset on testing.

B. A number of good studies provide evidence that testing can be very cost effective.

Motivation, cont'd

C. Emphasis placed on testing in industrial settings is likely to increase in coming years.

D. Increasingly, software test engineers

- ***get paid well***
- ***boss the developers around***

Motivation, cont’d

E. My all-time favorite testing-related failures:

1. *NASA deep space network 2-day crash*

2. *massive northeast power blackout*

F. The same cause for both -- ***what was it?***

III. Revised Goals for Week 3

A. Friday Lecture:

1. 1st half: *finish testing concepts overview*
2. 2nd half: *work on GIT repo setup;*
   - *Eric leads GIT installation efforts;*
   - *Cedric & Gene hammer out structure*

Revised Week 3 Goals, cont’d

B. **Friday Lab:**

1. *re-visit requirements spec base-lining*
2. *work with Julie to refine OCU requirements*

Continuing where we left off Wednesday ...

IV. **Review of testing terminology.**

-- *sitck "test" or "testing" in front of or after each:*

1. unit
2. module
3. integration
4. system
5. acceptance
6. black box
7. white box
8. design
9. plan
10. top-down
11. bottom-up
12. case
13. oracle
14. stub
15. driver
16. regression
17. coverage
18. subsumption
19. automation
20. mutation
21. harness
22. framework
23. suite

V. **Unit Testing**

A. Done at the level of function, aka method.

B. Provide inputs, expected outputs.

C. Check the actual outputs meet expected.

VI. Module Testing

A. Done at level of class, aka, module.

B. Define test fixtures.

C. Define unit-by-unit test execution.

D. Consider inter-function communication.

VII. **Integration Testing**

A. Done at level of package, aka, namespace.

B. Integrate multiple module tests.

C. Defined external data source test fixtures.

VIII. System Testing

A. Done at level of sub-systems, aka separate launch points

B. Super-integrate previously tested packages

IX. **Acceptance**

A. Done at level of HCI/API.

B. Provide inputs at external interface, not at code level.

X. Black Box Testing

A. Tests based on external specification.

B. Code is not used to generate tests.

XI. **White box**

A. Tests based on internal implementation.

B. Code paths used to generate tests.

XII. **Testing Design**

A. Organize all of the different levels of testing.

B. Define critical paths.

XIII. Test Plan

A. The framework-independent documentation of a testing level.

B. Function comment for unit test plan.

C. Class comment for module test plan.

D. Package comment for system test plan.

XIV. **Top-down Testing**

A. Top-level components tested first.

B. "Stubs" written for lower-level methods.

XV. **Bottom-up Testing**

A. Lower-level components tested first.

B. Function "drivers" written for upper-level meth-
ods.

XVI. **Test Case**

A. One input/output pair in a test plan.

XVII. **Testing Oracle**

A. The entity that determines the expected output.

B. The entity that validates the actual and expected output are equal.

XVIII. **Testing Stub**

A. A place holder for an unimplemented software component.

B. Provides "canned" data for other components being tested

XIX. **Test Driver**

A. Executes components being tested with upper-level components are not yet implemented.

XX. **Regression Testing**

A. Record results of step phase *n*.

B. Compare same-unit results with test phase *n-1*, expecting no differences.

XXI. Test Coverage

A. Ensure that test cover all white box execution paths.

XXII. Test Subsumption

A. When the results of one test of test case fully cover another case or test.

B. Allows redundant tests to be removed from a suite.

XXIII. **Test Automation**

A. Computational support for any and all aspects of testing.

B. Most typically automated are results recording, regression differencing, and coverage

XXIV. Mutation Testing

A. Systematic changes to code being tested and re-execution of tests.

B. Goal is to uncover test weaknesses.

XXV. Testing Harness

A. System-level test driver

XXVI. **Testing Framework**

A. Organizational structure of the tests and there execution.

B. Different frameworks support different testing styles.