Testing

As Paul Ehrlich (scientist, Nobel laureate) puts it –

"To err is human, but to really foul things up you need a computer."

Note: Modified by Stephanie Schwartz

Testing in the 21st Century

- Software defines behavior
  - network routers, finance, switching networks, other infrastructure
- Today's software market :
  - is much bigger
  - is more competitive
  - has more users
- Embedded Control Applications
  - airplanes, air traffic control
  - spaceships
  - watches
  - ovens
  - remote controllers
- Agile processes put increased pressure on testers
  - Programmers must unit test – with no training, education or tools !
  - Tests are key to functional requirements – but who builds those tests ?

Costly Software Failures

- NIST report, "The Economic Impacts of Inadequate Infrastructure for Software Testing" (2002)
  - Inadequate software testing costs the US alone between $22 and $59 billion annually
  - Better approaches could cut this amount in half
- Huge losses due to web application failures
  - Financial services : $6.5 million per hour (just in USA!)
  - Credit card sales applications : $2.4 million per hour (in USA)
- In Dec 2006, *Amazon.com's* BOGO offer turned into a double discount
- 2007 : Symantec says that most security vulnerabilities are due to faulty software

World-wide monetary loss due to poor software is staggering

Spectacular Software Failures

- **NASA's Mars Lander**: September 1999, crashed due to a units integration fault
- **Patriot Missile failure**: During the Gulf War, an American Patriot Missile battery in Saudi Arabia, failed to track and intercept an incoming Iraqi Scud missile. The Scud struck an American Army barracks, killing 28 soldiers and injuring around 100 other people. The General Accounting office reported on the cause of the failure. It turns out that the cause was an inaccurate calculation due to computer arithmetic errors.
- **Other Major failures**: Ariane 5 explosion, Mars Polar Lander, Intel's Pentium FDIV bug, Heathrow Terminal 5 Opening
- **Toyota brakes** : Dozens dead, thousands of crashes
- Poor testing of safety-critical software can cost *lives* :
  - **THERAC-25 radiation machine**: 3 dead

Ariane 5: exception-handling bug : forced self destruct on maiden flight (64-bit to 16-bit conversion: about 370 million $ lost)

We need our software to be dependable
Testing is *one* way to assess dependability

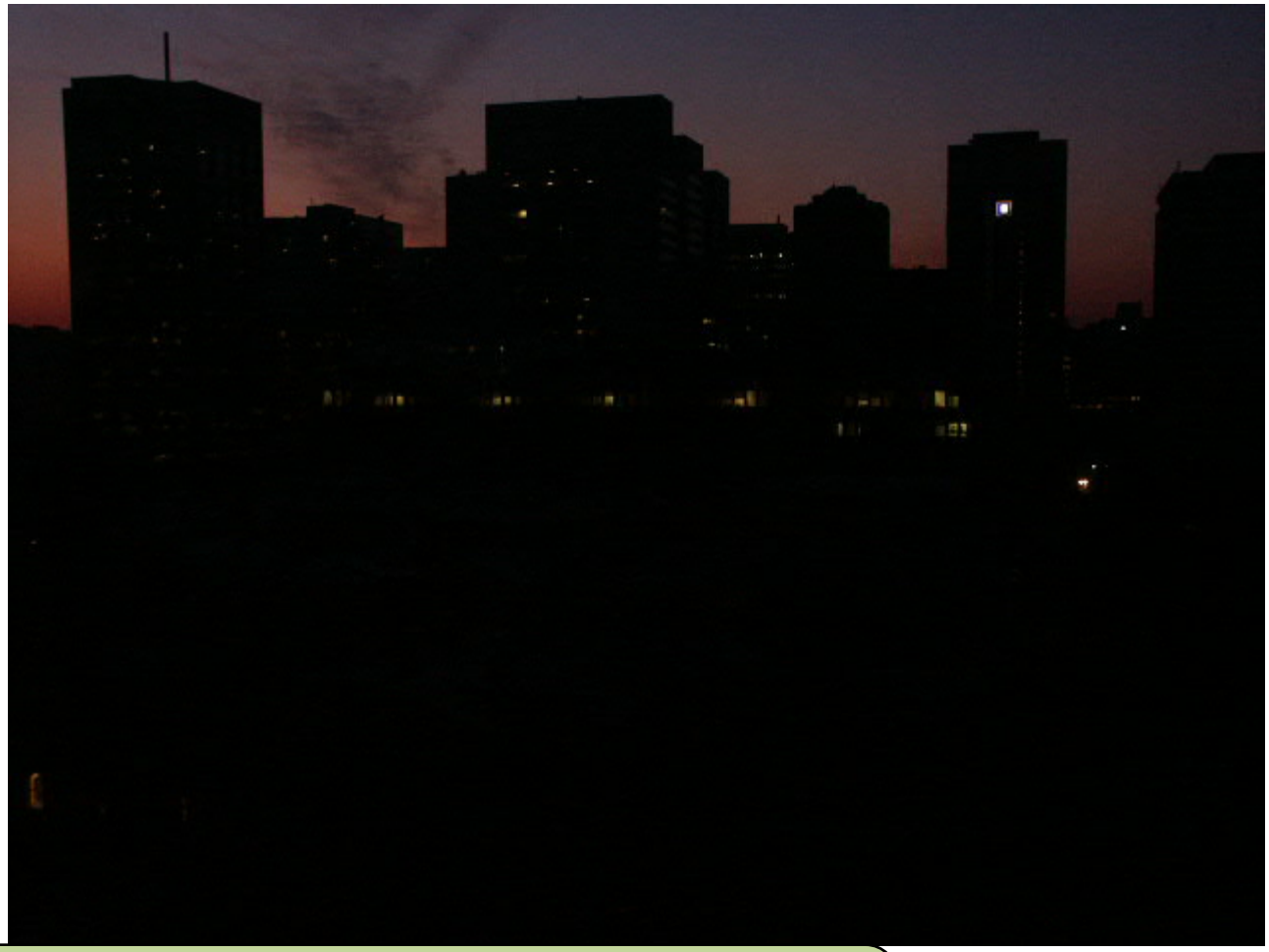
Northeast Blackout of 2003

508 generating units and 256 power plants shut down

Affected 10 million people in Ontario, Canada

Affected 40 million people in 8 US states

Financial losses of $6 Billion USD



The alarm system in the energy management system failed due to a software error and operators were not informed of the power overload in the system

Testing in the 21st Century

- More safety critical, real-time software
- Embedded software is ubiquitous … check your pockets
- Enterprise applications means bigger programs, more users
- Security is now all about software faults
  - Secure software is reliable software
- The web offers a huge deployment platform
  - Very competitive and very available to more users
  - Web apps are distributed
  - Web apps must be highly reliable

Planning for Quality

- Testing is an integral part of quality control in software development
- However, **quality cannot be "tested in"**
- Quality must be built into the entire development process

Verification and Validation

- Validation is oriented to **prevention** (Are we building the right product?)
- Verification is oriented to **detection** (Did we build the product right?)
- Both verification and validation are need to ensure high quality results
- Independent verification and validation (IV&V) is the "gold standard"
  - Verification and validation done by a unit not involved in the development
  - Expensive to implement, hence not done in all companies

Validation Techniques

- Reviews and meetings to evaluate documents, plans, requirements, specifications, and even code
  - Walkthroughs
  - Inspections (usually a more formalized walkthrough)
- Audit standards and processes

Verification Techniques

- Unit testing
- Integration testing
- System testing
- Regression testing
- Performance (load) testing
- Acceptance testing

Cost of Testing

Companies spend at least half of their development budget on testing,

- In the real-world, testing is the principle post-design activity
- Restricting early testing usually increases cost
- Extensive hardware-software integration requires more testing

Approach

If you don't know why you're conducting each test, it won't be very helpful

- Written test objectives and requirements must be documented
- What are your planned coverage levels?
- How much testing is enough?

When to Start

If you don’t start planning for each test when the functional requirements are formed, you’ll never know why you’re conducting the test

- What fact is each test trying to verify?

- Requirements definition teams need testers!

Cost of Not Testing

- Not testing is even more expensive
- Planning for testing after development is prohibitively expensive
- A test station for circuit boards costs half a million dollars …
- Software test tools cost less than $10,000 !!!

The Five Key Deliverables of Test Planning

Deliverable	Description
Test Approach	Explains the objectives and scope of the test; Documents entry/exit criteria and key dates
Test Scenarios	Provides high-level descriptions of functionality to be tested
Test Cases	Detailed procedure that fully tests a feature or an aspect of a feature.
Test Conditions and Expected Results	Describes all items and results that must be covered to fulfill each Test Scenarios
Test Cycle Control Sheet	Groups test scripts into logical categories (or cycles); documents when and by whom each cycle will be executed.
Test Scripts	Provides step-by-step instructions and detailed results for a test executor to follow during test execution

Software Application Testing

- A master test plan is developed during the analysis phase.
- During the design phase, unit, system and integration test plans are developed.
- The actual testing is done during implementation.
- Test plans provide improved communication among all parties involved in testing.

Different Types of Testing Techniques

- Static or dynamic techniques
  - Static testing means that the code being tested is not executed.
  - Dynamic testing involves execution of the code.
- Test is automated or manual
  - Automated means computer conducts the test.
  - Manual means that people complete the test.

Different Types of Tests

- **Inspection**: a testing technique in which participants examine program code for predictable language-specific errors
- **Walkthrough**: a peer group review of any product created during the systems development process, including code
- **Desk checking**: a testing technique in which the program code is sequentially executed manually by the reviewer

Different Types of Tests (Cont.)

- **Unit testing**: each module is tested alone in an attempt to discover any errors in its code
- **Integration testing**: the process of bringing together all of the modules that a program comprises for testing purposes
  - Modules are typically integrated in a top-down incremental fashion.

Different Types of Tests (Cont.)

- **Regression testing:** verifies that software which was previously developed and tested still performs the same way after it was changed or interfaced with other software. Changes may include software enhancements, patches, or configuration changes
- **System testing**: the bringing together of all of the programs that a system comprises for testing purposes
  - Programs are typically integrated in a top-down, incremental fashion.

Different Types of Tests (Cont.)

- **Performance testing:** determine how a system performs in terms of responsiveness and stability under a particular workload
- **Acceptance Testing**: testing by the end-users

The Testing Process

- The purpose of testing is to confirm that the system satisfies the requirements.
- Good testing techniques should reveal errors
- Testing must be planned.
- Test case is a specific scenario of transactions, queries or navigation paths.

The Testing Process (Cont.)

- Test cases represent either:
  - Typical system use
  - Critical system use, or
  - Abnormal system use.
- Test cases and results should be thoroughly documented so they can be repeated for each revision of an application.

Automated Testing

- Improves testing quality
- Reduce testing time up to 80%
- Functions:
  - Create recorded data entry and user action scripts
  - Compare test results between test cases
  - Simulate high-volume for stress-testing

Testing Strategy

- Testing should begin “in the small” – finding and correcting errors is much easier if done near the source
- Move from the “small” to the “large”
- Typical sequence
  - Unit testing
  - Integration testing
  - System testing
  - Acceptance Testing
- Performance and Regression testing are done at various points when appropriate

Software Testing Objectives

- Test suites are created and evaluated with the goal of providing as much coverage as is feasible/practical
- Test cases should be carefully chosen to:
  - Test for the presence of all required functionality
  - Have high probability of finding as yet undiscovered errors
  - Test areas of the program where errors would have the most serious consequences
  - Test for probable attempted misuse (deliberate or unintentional) of the program
- It is impossible to test everything in a software system

Exercise

Consider a program that contains 3 *if … then …else* statements

1. How many different program states could possible arise from just these statements during execution of the program?
2. Answer the same question for a program with 20 *if … then … else* statements.
3. What are the implications of these results for testing?

2^3 = 8

2^20 = 1048576

Test Scenario & Case Design

- We should design test plans that have the highest likelihood of finding the most errors with limited time and effort.
- A systematic approach should be adopted to minimize wasted effort and ensure coverage as complete as possible given the testing resources available.

Developing Test Scenarios

- A Test Scenario is a business requirement to be tested
- A Scenario is any functionality that can be tested.
- For a Flight Reservation Application a few scenarios would be
- 1) Check the Login Functionality
- 2) Check that a New Order can be created
- 3) Check that an existing Order can be opened
- 4) Check that a user can FAX an order
- 5) Check that the information displayed in the HELP section is correct

Test Cases con't

- Consider the **Test scenario** Check Login Functionality
- There many possible **test cases** like
  - Check response on entering valid Agent Name & Password ,
  - Check response on entering invalid Agent Name & Password ,
  - Check response when Agent Name is Empty & Login Button is pressed, and many more
- Test scenarios are rather vague and cover a wide range of possibilities. Testing is all about being very specific.
- Hence we need Test Cases
- Consider the test case , **Check response on entering valid Agent Name and password**.
  - this test case needs input values Agent Name & Password
  - **Identifying test data can be time-consuming and may some times require creating test data afresh**.
  - For this reason it needs to be documented

Developing Test Cases

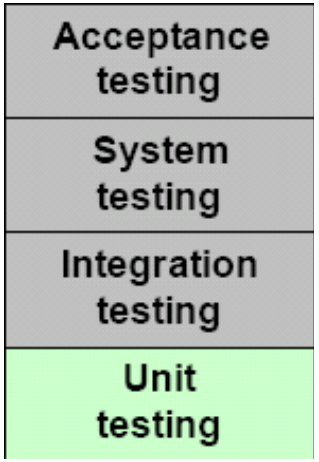
- Test case forms have the following sections:
  - Test Case ID
  - Category/Objective of Test
  - Description
  - System Version

White Box Testing (Glass Box Testing)

- Uses the form of the code itself to construct test cases
- Designed to uncover errors in program logic
- Isn't *directly* driven by requirements, although it addresses the important *implicit requirement* for a robust and reliable system
- Done mostly at the unit test level

Testing Phases in Detail

Unit Testing (White box testing)

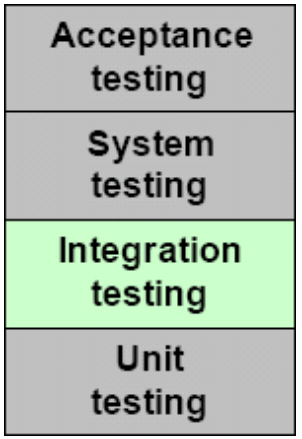


- The most ‘micro’ scale of Testing
- A unit = smallest testable software component
  - Objects and methods
  - Procedures / functions
- Requires detailed knowledge of the internal program design and code.
- The units are tested in isolation.
- Ensures the component is working according to the detailed design/build specifications of the module.

Unit Testing (cont'd)

- Unit testing focuses on **verifying small unit(s) or modules** of the software design
- Unit testing should include test cases for:
  - Interface with other units
  - Local data structures
  - Independent paths through the code
  - Error handling paths
  - Boundary conditions

Integration Testing



- Testing of more than one (tested) unit together to determine if they function correctly.
- Focus on interfaces
  - Communication between units
- It is done using the integration test design prepared during the architecture design phase.
- Helps assembling incrementally a whole system, ensuring the correct ‘flow’ of data from the first through the final component.
- Done by developers/designers and testers in collaboration
- Also called Interface Testing or Assembly Testing.

Integration Testing (cont'd)

- Integration testing is a systematic technique for constructing the larger program structure from smaller units, while testing to **uncover errors in the interfaces between units**
- Integration testing often uses one of the following pairs of techniques:
  - Writing driver modules
  - Assembling from the bottom up

  or
  - Stubbing lower level units
  - Assembling from the top down

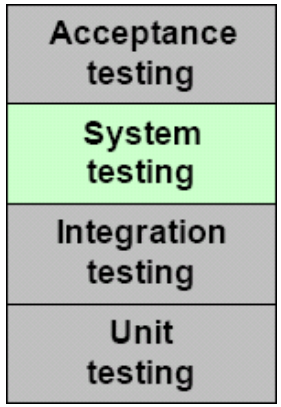
Stub Testing

- **Stub testing**: a technique used in testing modules, especially where modules are written and tested in a top-down fashion, where a few lines of code are used to substitute for subordinate modules

Regression Testing

- Regression testing focuses on testing the larger program into which a new or modified unit or module has been inserted.
- Regression testing usually involves re-executing some subset of the test plan that was used to test the larger system before its modification
- Regression testing can also involve employing new test cases designed to test the impact of the modifications on the larger system
- Regression testing focuses on **verifying that modifications have no unintended impact** on the larger program

System Testing



- Testing the system as a whole - Black-box type testing that is based on overall requirements specifications; covers all combined parts of a system.

- Most software development is done as a part of a larger system. System testing focuses on **testing the larger system once it is assembled**
- Systems testing is usually a series of tests designed to exercise the completed system and all its components within the "production" environment

System Testing (cont'd)

- Ensures that system meets all functional and business requirements.

- Focus
  - Verifying that specifications are met
  - Validating that the system can be used for the intended purpose

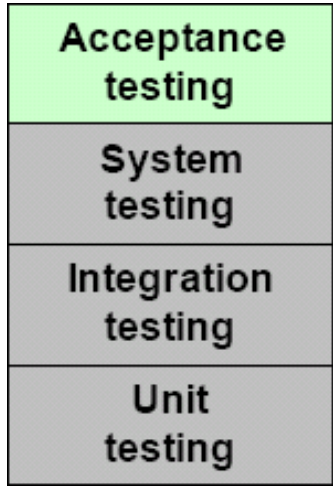
- The system test design is derived from the system design documents and is used in this phase.

- It can involve a number of specialized types of tests to check performance, stress, documentation etc. Sometimes testing is automated using testing tools.

System Testing(Cont.)

- Types of System Tests:
  - *Recovery testing* — forces software (or environment) to fail in order to verify that recovery is properly performed. how does the software recover from a series of possible exceptional events?
  - *Security testing* — verifies that protection mechanisms built into the system will protect it from improper penetration
  - *Stress testing/Load testing* — tries to break the system. Will system perform under real life loads. How does the software perform when given abnormally large volume demands?
  - *Performance testing* — determines how the system performs on the range of possible environments in which it may be used
  - Migration testing – ensure software can be moved from older infrastructure to current system infrastructure without any problems
  - Regression testing – make sure none of the changes made over the course of development have caused new bugs. Also makes sure no old bugs appear from addition of new software modules over time.
  - Usability testing – focuses on the user's ease to use application, flexibility in handling controls and ability to meet objectives.

Acceptance Testing



- The process whereby actual users test a completed information system, the end result of which is the users' acceptance of it

- To determine whether a system satisfies its acceptance criteria and business requirements or not.
- Similar to System testing in that the whole system is checked, but the important difference is the change in focus.

- Done by real business users.

- It enables the customer to determine whether to accept the system or not.

- Also called as Beta Testing, Application Testing or End User Testing.

- Approach
  - Should be performed in real or simulated operating environment .
  - Customer should be able to perform any test based on their business processes.
  - Final Customer sign-off.

Acceptance Testing (cont'd)

- Where appropriate customers should be involved in earlier testing phases as well
- **Acceptance criteria** should be established as part of the requirements
- Acceptance testing **verifies that the system satisfies the requirements**
- Developers should anticipate *implied requirements* for quality, performance, and usability as well

Acceptance Testing by Users (Cont'd)

- **Alpha testing**: user testing of a completed information system using simulated data
- **Beta testing**: user testing of a completed information system using real data in the real user environment

Black Box Testing (Behavioral Testing)

- Focuses on the functional requirements
- Test cases are formed to test the program's behavior against various input scenarios, without regard to the internal logic of the code itself
- Used in integration, system, and acceptance testing
- Black box testing is *not an alternative* to white box testing
- White box and black box testing are complementary

Black Box Testing (cont'd)

- Black box tests suites are designed to answer the following questions
  - How can functionality be verified?
  - How can system performance be tested?
  - How can system usability be tested?
  - What classes of input will make good test cases?
  - How does the program behave at the boundaries of normal input classes?
  - What combinations of data and/or actions should be considered?
  - How does the system behave against "bad" data?

Capacity Planning

- In 1997, Oxford Health Plans posted a $120 million loss to its books. The company's unexpected growth was its undoing because the system, which was originally planned to support the company's 217,000 members, had to meet the needs of a membership that exceeded 1.5 million.
- System users found that processing a new-member sign-up took 15 minutes instead of the proposed 6 seconds. Also, the computer problems left Oxford unable to send out bills to many of its customer accounts and rendered it unable to track payments to hundreds of doctors and hospitals.
- In less than a year, uncollected payments from customers tripled to more than $400 million and the payments owed to caregivers amounted to more than $650 million.
- Mistakes in infrastructure planning cost far more than the cost of hardware, software, and network equipment alone.

Security Testing

- You may be aware that there are professional security firms that organizations can hire to break into their own networks to test security. BABank (pseudonym) was about to launch a new online banking application, so it hired such a firm to test its security before the launch. The bank's system failed the security test – badly.
- The security team began by mapping the bank's network. It used network security analysis software to test password security, and dialing software to test for dial-in phone numbers. This process found many accounts with default passwords (i.e. passwords set by the manufacturer that are supposed to be changed when the systems are first set up).
- The team then tricked several high-profile users into revealing their passwords to gain access to several high-privilege accounts. Once into these computers, the team used password-cracking software to find passwords on these computers and ultimately gain the administrator passwords on several servers.
- At this point, the team transferred $1000 into their test account. They could have transferred much more, but the security point was made.

Testing System Documentation and Help

- Testing plans should include a test of the system documentation
- Similarly help functions should be tested as well
- Errors in documentation and help facilities are extremely frustrating to users