CH09: Testing the System

- to ensure the system does what the customer wants it to do:
- * Principles of System Testing
- * Function Testing
- * Performance Testing
- * Reliability, Availability, and Maintainability

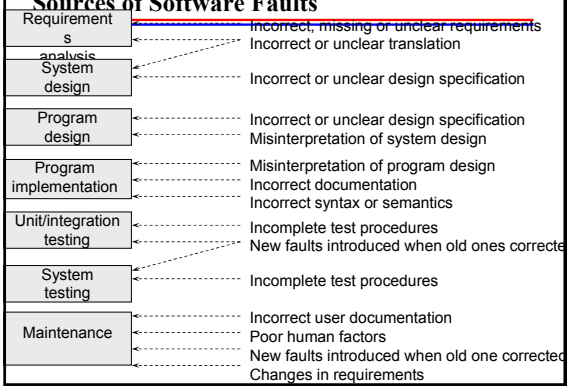
More System Tests

- * Acceptance Testing
- * Installation Testing
- * Automated System Testing
- * Test Documentation
- * Testing Safety-critical systems

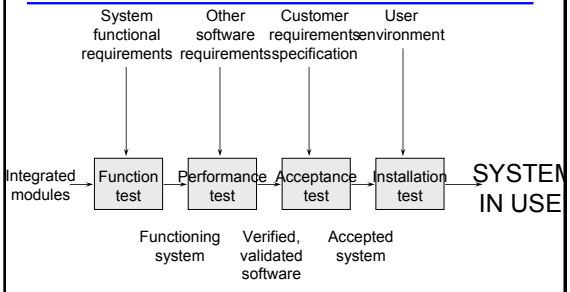
Principles of System Testing

- Sources of Software Faults
- System Testing Process
- Process Objectives
- Configuration Management
- Test Team

Sources of Software Faults



System Testing Process



Process Objectives

- A **function test** checks that the integrated system performs its functions as specified in the requirements.
- A **performance test** compares the integrated components with the nonfunctional system requirements,
  - **such as security, accuracy, speed, and reliability**

Verified vs. Validated System

- A **verified system** meets the requirement specifications (designer's view).
- A **validated system** meets the requirement definitions (customer's view).

Tests for Customers

- An **acceptance test** assures the system meet the customers' satisfaction.
- An **installation test** assures the system is installed correctly and working at the actual customer's hardware.

Configuration Management

- A **system configuration** is a collection of system components delivered to a particular customer or operating system.
- **Configuration management** helps us keep track on the difference system configurations.
  - **Also keep track of change: called** change control

Version and Release

- A configuration for a particular system is sometimes called a **version**.
- A new **release** of the software is an improved (?) system intended to replace the old one.
- A **regression test** is a test applied to a new version or release to verify that it still performs the same functions in the same manner as an older one.

Tracking the Changes: Deltas, Separate files, and Conditional compilation

- **Separate files**: Keep separate files for each different version or release.
- **Deltas**: Keep one main version. Other versions are stored as differences from the main version. The difference file is called a delta.
- **Conditional compilation**: One single code addresses all version. Use compiler to determine which statements apply to which versions.

Test Team

- Everybody needs to be involved.
- Developers usually do unit and integration testing.
- Testers (**testing group**) do functional and performance tests.
- Testers will ask analysts or front-end people to clarify requirements.
- Users, technical writers, ...

Function Testing

- Test what the system is supposed to do based on the system's functional requirements. Test should:
  - **have a high probability of detecting a fault**
  - **use a test team independent of the designers and programmers**
  - **know the expected actions and output**
  - **test both valid and invalid input**
  - **never modify the system just to make testing easier**
  - **have stopping criteria**

Performance Testing

- addresses nonfunctional requirements.
- Types include:
  - **Stress tests, Volume tests -- users and data**
  - **Configuration test -- software and hardware configurations**
  - **Compatibility test -- between systems**
  - **Regression tests -- between versions**
  - **Security test**

Lot more performance tests

- **Timing test -- respond time**
- **Environmental test -- perform at installation site**
- **Quality tests -- reliability, availability**
- **Recovery tests -- restart**
- **Maintenance tests -- diagnostic tools and tracing**
- **documentation tests -- e.g. follow the installation guide**
- **Human factors test, usability tests.**

Reliability, Availability, and Maintainability

- Definitions
- Failure Data
- Measuring Reliability, Availability, and Maintainability
- Reliability Stability and Growth
- Reliability Prediction
- Importance of the operational Environment

Definitions

- Reliability involves behavior over a period of time.
- Availability describes something at a given point in time.
- Maintainability describes what need to be done to keep the system functioning.

More formal definitions

- **Software** reliability **is the probability that a system will operate without failure under given condition for a give time interval.**
- **Software** availability **is the probability that a system is operating successfully according to specification at a given point in time.**
- **Software** maintainability **is the probability that a maintenance activity can be carried out within a stated time interval and using stated procedures and resources.**

Failure Data

- Reliability, availability, and maintainability are measured based on the working history of a complete system.
- Failure data must be kept to allow the measurements:
  - **time between each failure ("inter-failure time")**
  - **time for each maintenance**

MTTF, MTTR, MTBF

- Mean time to failure (MTTF) is the average value of inter-failure times.
- Mean time to repair (MTTR) is the average value of time taking to fix each failure.
- Mean time between failures (MTBF):
- MTBF = MTTF + MTTR

Measuring Reliability, Availability, and Maintainability

- Reliability = MTBF/(1+MTBF)
- Availability = MTBF/(MTBF + MTTR)
- Maintainability = 1/(1 + MTTR)

Reliability Stability and Growth

- tell us whether the software is improving
- If the inter-failure times stay the same,
- then we have reliability stability!!!
- If the inter-failure times increase,
- then we have reliability growth (getting better)!!!

Reliability Prediction

- Use historical information about failures to build a predictive models of reliability.
- = assuming the change in system behavior is the same by fixing one fault as by fixing another
- - fixing one fault might introduce a lot more fault (predictive models can not take account of this problem)

Importance of the operational Environment

- Capture operational profile that describes likely user input over time.
- e.g. % of create, % of delete, or % of modify used by a user
- (statistical testing) Test cases are created to test them according to the %.
  - **Testing concentrates on the parts of the system most likely to be used**
  - **Reliability from these test gives the reliability as seen by the user.**

Acceptance Testing

- Now the customer leads testing and defines the cases to be tested.
- **Benchmark test** uses a set of test cases that represent typical conditions of usage.
- **Pilot test** installs the system on an experimental basis and rely on the everyday working of the system to test all functions.

Alpha and Beta tests

- In house test is called an **alpha test**.
- The customer's pilot test is called **beta test**.
- **Parallel testing**: the new system operates in parallel with the previous version. (Something to fall back to in case the new one does not work!)

Installation Testing

- Final round of testing!
  - **involves installing the system at customer's sites.**
- After installation, run regression test (most important test cases) to insure the software is working "in the field".
- When the customer is satisfied with the results, testing is complete and the system is formally delivered.
- Done for now!!!

Automated System Testing

- **Simulator** presents to a system all characteristics of a device or system without actually having the device or system available.
- Sometimes a device simulator is more helpful than the device itself.
- Record and Play-back testing tools to simulate users.

Test Documentation

- Test Plans (covered earier)
- Test Analysis Report
- Problem Report Forms

Test Analysis Report

- When a test has been administered, we analyze the results to determine if the function or performance tested meets the requirements. Anaysis Report:
  - **documets the results of tests**
  - **if a failure occurs, it provides information needed to duplicate the fauilure and to locate and fix the source of the problem.**
  - **provides info to determine if the probject is complete**
  - **establishes confidnece in the system's performance**

Problem Report Forms

- capture data about faults and failures in problem report forms
- **discrepancy report form** (it is called **MR** (modification request) in Lucent)
  - **is a problem report that describes occurrences of problems where actual system behaviors or attributes do not match with what we expect.**
- **fault report form** explains how a fault was found and fixed, often in response to filing a discrepancy report form.

Discrepancy report form

DISCREPANCY REPORT FORM

DRF Number:_____ **Tester name:**_____
Date: _____ **Time:** _____
Test Number: _____
Script step executed when failure occurred: _____
Description of failure: _____

Activities before occurrence of failure:

Expected results:

Requirements affected:

Effect of failure on test:

Effect of failure on system:

Severity level:
(LOW) 1 2 3 4 5 (HIGH)

FAULT REPORT **S.P0204.6.10.3016**

ORIGINATOR: Joe Bloggs

BRIEF TITLE: Exception 1 in dps_c.c line 620 raised by NAS

FULL DESCRIPTION Started NAS endurance and allowed it to run for a few minutes. Disabled the active NAS link (emulator switched to standby link), then re-enabled the disabled link and CDIS exceptioned as above. (I think the re-enabling is a red herring.) (during database load)

ASSIGNED FOR EVALUATION TO: **DATE:**

CATEGORISATION: 0 1 2 3 Design Spec Docn
SEND COPIES FOR INFORMATION TO:
EVALUATOR: **DATE:** 8/7/92

CONFIGURATION ID	ASSIGNED TO	PART
dpo_s.c		

COMMENTS: dpo_s.c appears to try to use an invalid CID, instead of rejecting the message. AWJ

ITEMS CHANGED

CONFIGURATION ID	IMPLEMENTOR/DATE	REVIEWER/DATE	BUILD/ISSUE NUM	INTEGRATOR/DATE
dpo_s.c v.10	AWJ 8/7/92	MAR 8/7/92	6.120	RA 8-7-92

COMMENTS:

CLOSED

FAULT CONTROLLER: **DATE:** 9/7/92

Testing Safety-critical systems

- **Safety-critical systems**: failure of the system can harm or kill people!
- Ultrahigh reliability system: has at most one failure in 10^9 hours!
  - **i.e. the system can fail at most once in over 100,000 years of operation.**
- Problem: if a program has worked failure-free for x hours, there is about a 50:50 chance that it will survive the next x hours before failing!

Methods to help understand and assure reliability

- Design Diversity
- Software Safety Cases
- Cleanroom

Design Diversity

- built software using same requirements specifications,
- but in several independent different designs to form several independent systems.
- Each system runs in parallel and
- a voting scheme coordinates actions when one system's results differ from the others'
- e.g. U.S. space shuttle

Software Safety Cases

- assign failure rates or constraints to each component of the system
- then estimate the failure rates or constraints of the entire system
- e.g. use fault-tree analysis

Cleanroom

- address two principles:
  - **to certify the software with respect to the specifications**
  - **to produce zero-fault or near-zero-fault software**
- use formal proofs to certified with respect to specifications
  - **Not unit testing**
- use statistical usage testing to determine the expected MTTF and other quality measure.