Software Testing and Strategies

Instructor: Dr. Jerry Gao
Software Testing and Strategies

- Verification and Validation
- Software Testing Organization
- Software Testing and Testing Process
- Testing Strategies
- Test Planning and Documentation
- Special Testing Issues for Application Systems
Verification and Validation

Software testing is one element of a broader topic that is often referred to as Verification and Validation (V&V).

Verification --> refers to the set of activities that ensure that software correctly implements a specific function.

Validation --> refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements.

Boehm [BOE81]:

Verification: “Are we building the product right?”
Validation: “Are we building the right product?”

The definition of V&V encompasses many of SQA activities, including:

- formal technical reviews
- quality and configuration audits
- performance monitoring
- different types of software testing
- feasibility study and simulation
Software Testing Organization

**Testing objective:** uncover the errors (defects) in the software, including errors in:
- requirements from requirement analysis
- design documented in design specifications
- coding (implementation)
- system resources and system environment
- hardware problems and their interfaces to software

Software testing can be considered to be (psychologically) destructive.

**Who are involving software testing?**
- developers
- testers (test engineers) in ITG
- SQA group

**Software testing organization:**
- Individual tester in a development team
- Independent test group (ITG)
Software Testing Process

- Unit test
- Integration test
- Validation test
- System test

V&V Targets
- Code & Implementation
- Software Design
- Requirements
- System engineering
**Unit Test (Component Level Test)**

Unit testing: Individual components are tested independently to ensure their quality. The focus is to uncover errors in design and implementation, including:
- data structure in a component
- program logic and program structure in a component
- component interface
- functions and operations of a component

Unit testers: developers of the components.

---

**White-box**

- Interface
- Internal logic, data, structure
- Input
- Output

**Black-box**

- Interface
- Operations and Functions with I/O
- Input
- Output
- Operation
Integration Test

Integration test: A group of dependent components are tested together to ensure their the quality of their integration unit.

The focus is to uncover errors in:
- Design and construction of software architecture
- Integrated functions or operations at sub-system level
- Interfaces and interactions between them
- Resource integration and/or environment integration

Integration testers: either developers and/or test engineers.
Integration Testing Strategies

Approaches:  

a) non-incremental integration  
b) incremental integration

Non-incremental integration:

- Big Band - combine (or integrate) all parts at once.

Advantages:  

simple

Disadvantages:  

- hard to debugging, not easy to isolate errors  
- not easy to validate test results  
- impossible to form an integrated system

Incremental integration:

integrate the system step-by-step (or piece by piece)  
in an well-designed order. Three important methods:  
a) Top-down  
b) bottom-up  
c) Sandwich  
- uses a top-down for upper-level modules  
and bottom-up for low-level modules
**Top-down Integration**

_Idea:_ Modules are integrated by moving downward through the control structure.

_Modules subordinate to the main control module are incorporated into the system in either a depth-first or breadth-first manner._

 Integration process (five steps):
1. _the main control module is used as a test driver, and the stubs are substituted for all modules directly subordinate to the main control module._
2. _subordinate stubs are replaced one at a time with actual modules._
3. _tests are conducted as each module is integrated._
4. _On completion of each set of tests, another stub is replaced with the real module._
5. _regression testing may be conducted._

_Pros and cons top-down integration:_
- stub construction cost
- major control function can be tested early.
**Bottom-Up Integration**

**Idea:** Modules at the lowest levels are integrated at first, then by moving upward through the control structure.

**Integration process (five steps):**
1. Low-level modules are combined into clusters that perform a specific software sub-function.
2. A driver is written to coordinate test case input and output.
3. Test cluster is tested.
4. Drivers are removed and clusters are combined moving upward in the program structure.

**Pros and cons of bottom-up integration:**
- no stubs cost
- need test drivers
- no controllable system until the last step
Validation Test

Validation test: The integrated software is tested based on requirements to ensure that we have a right product.

The focus is to uncover errors in:
- System input/output
- System functions and information data
- System interfaces with external parts
- User interfaces
- System behavior and performance

Validation testers: test engineers in ITG or SQA people.
System Test

System test: The system software is tested as a whole. It verifies all elements mesh properly to make sure that all system functions and performance are achieved in the target environment.

The focus areas are:
- System functions and performance
- System reliability and recoverability (recovery test)
- System installation (installation test)
- System behavior in the special conditions (stress and load test)
- System user operations (acceptance test/alpha test)
- Hardware and software integration and collaboration
- Integration of external software and the system

System testers: test engineers in ITG or SQA people.

When a system is to be marketed as a software product, a testing process called beta testing is often used.
System Testing

Recovery testing - a system test that forces the software to fail in various ways and verifies that recovery is properly performed.

Security testing - to verify that protection mechanism built into a system will in fact protect it from improper penetration.

Stress testing - is designed to confront programs with abnormal conditions. - quantity, frequency, or volume.

Performance testing - is designed to test run-time performance of software within the context of an integrated system.

Installation testing - is designed to test the installation procedure and its supported software.
Test Planning

Test planning is concerned with setting out standards for the testing process rather than describing product tests.

Test plan consists of the following:
- standards for the testing process
- required resources (hardware, software and engineers)
- testing schedules (testing tasks and milestones)
- test items (what should be tested)
- test recording procedures
  (test results must be systematically recorded)
- constraints

Test planning is a task of a test manager.
A test plan is its output of the planning. (Figure 17.11)
Test Issues in Real World

Software testing is very expensive.

How to achieve test automation?????

Test criteria, test coverage, adequate testing.

Other software testing:

GUI Testing

Object-Oriented Software Testing

Component Testing and Component-based Software Testing

Domain-specific Feature Testing

Testing Web-based Systems