

# Software Engineering

## Software Testing

Based on Software Engineering, 7<sup>th</sup> Edition by Ian Sommerville

Stan Kurkovsky

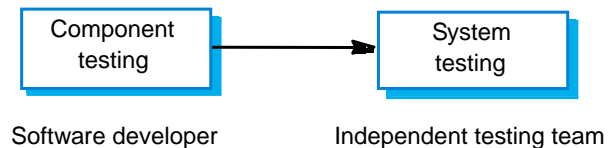
### Objectives

- To discuss the distinctions between validation testing and defect testing
- To describe the principles of system and component testing
- To describe strategies for generating system test cases
- To understand the essential characteristics of tool used for test automation

Stan Kurkovsky

## The testing process

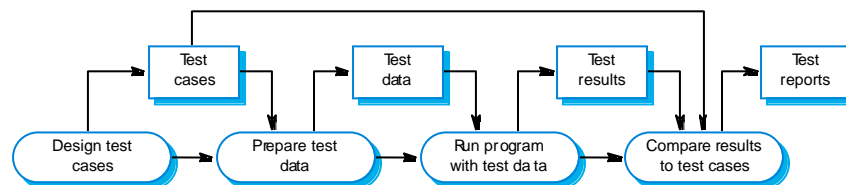
- Component testing
  - Testing of individual program components;
  - Usually the responsibility of the component developer (except sometimes for critical systems);
  - Tests are derived from the developer's experience.
- System testing
  - Testing of groups of components integrated to create a system or sub-system;
  - The responsibility of an independent testing team;
  - Tests are based on a system specification.



Stan Kurkovsky

## Testing process goals

- **Validation testing**
  - To demonstrate to the developer and the system customer that the software meets its requirements;
  - A successful test shows that the system operates as intended.
- **Defect testing**
  - To discover faults or defects in the software where its behavior is incorrect or not in conformance with its specification;
  - A successful test is a test that makes the system perform incorrectly and so exposes a defect in the system.



Stan Kurkovsky

## Testing policies

- Only exhaustive testing can show a program is free from defects. However, exhaustive testing is impossible,
- Testing policies define the approach to be used in selecting system tests:
  - All functions accessed through menus should be tested;
  - Combinations of functions accessed through the same menu should be tested;
  - Where user input is required, all functions must be tested with correct and incorrect input.

Stan Kurkovsky

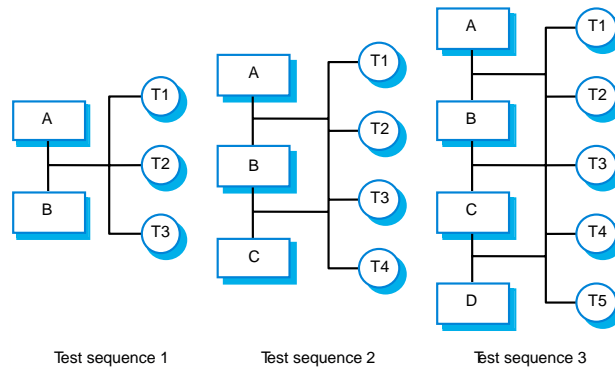
## System testing

- Involves integrating components to create a system or sub-system.
- May involve testing an increment to be delivered to the customer.
- Two phases:
  - **Integration testing** - the test team have access to the system source code. The system is tested as components are integrated.
  - **Release testing** - the test team test the complete system to be delivered as a black-box.

Stan Kurkovsky

## Integration testing

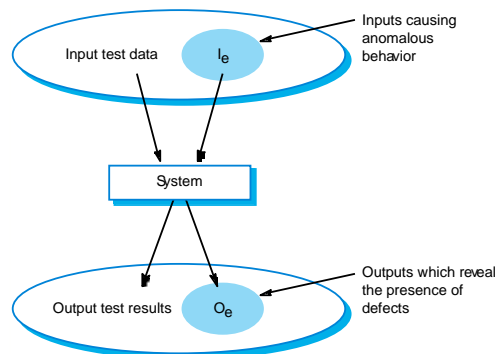
- Involves building a system from its components and testing it for problems that arise from component interactions.
- Top-down integration
  - Develop the skeleton of the system and populate it with components.
- Bottom-up integration
  - Integrate infrastructure components then add functional components.
- To simplify error localization, systems should be incrementally integrated.



Stan Kurkovsky

## Release testing

- The process of testing a release of a system that will be distributed to customers.
- Primary goal is to increase the supplier's confidence that the system meets its requirements.
- Release testing is usually black-box or functional testing
  - Based on the system specification only;
  - Testers do not have knowledge of the system implementation.
- Black box testing



Stan Kurkovsky

## Testing guidelines

- Testing guidelines are hints for the testing team to help them choose tests that will reveal defects in the system
  - Choose inputs that force the system to generate all error messages;
  - Design inputs that cause buffers to overflow;
  - Repeat the same input or input series several times;
  - Force invalid outputs to be generated;
  - Force computation results to be too large or too small.
- Use cases
  - Use cases can be a basis for deriving the tests for a system. They help identify operations to be tested and help design the required test cases.
  - From an associated sequence diagram, the inputs and outputs to be created for the tests can be identified.

Stan Kurkovsky

## Performance testing

- Part of release testing may involve testing the emergent properties of a system, such as performance and reliability.
- Performance tests usually involve planning a series of tests where the load is steadily increased until the system performance becomes unacceptable.

### Stress testing

- Exercises the system beyond its maximum design load. Stressing the system often causes defects to come to light.
- Stressing the system test failure behavior. Systems should not fail catastrophically. Stress testing checks for unacceptable loss of service or data.
- Stress testing is particularly relevant to distributed systems that can exhibit severe degradation as a network becomes overloaded.

Stan Kurkovsky

## Component testing

- Component or unit testing is the process of testing individual components in isolation.
- It is a defect testing process.
- Components may be:
  - Individual functions or methods within an object;
  - Object classes with several attributes and methods;
  - Composite components with defined interfaces used to access their functionality.

## Object class testing

- Complete test coverage of a class involves
  - Testing all operations associated with an object;
  - Setting and interrogating all object attributes;
  - Exercising the object in all possible states.
- Inheritance makes it more difficult to design object class tests as the information to be tested is not localized.

Stan Kurkovsky

## Interface testing

- Objectives are to detect faults due to interface errors or invalid assumptions about interfaces.
- Particularly important for object-oriented development as objects are defined by their interfaces.

### Interface types

- Parameter interfaces
  - Data passed from one procedure to another.
- Shared memory interfaces
  - Block of memory is shared between procedures or functions.
- Procedural interfaces
  - Sub-system encapsulates a set of procedures to be called by other sub-systems.
- Message passing interfaces
  - Sub-systems request services from other sub-systems.

### Interface errors

- Interface misuse
  - A calling component calls another component and makes an error in its use of its interface e.g. parameters in the wrong order.
- Interface misunderstanding
  - A calling component embeds assumptions about the behavior of the called component which are incorrect.
- Timing errors
  - The called and the calling component operate at different speeds and out-of-date information is accessed.

Stan Kurkovsky

## Test case design

- Involves designing the test cases (inputs and outputs) used to test the system.
- The goal of test case design is to create a set of tests that are effective in validation and defect testing.
- Design approaches:
  - Requirements-based testing;
  - Partition testing;
  - Structural testing.

Stan Kurkovsky

## Requirements based testing

- A general principle of requirements engineering is that requirements should be testable.
- Requirements-based testing is a validation testing technique where you consider each requirement and derive a set of tests for that requirement.

Stan Kurkovsky

## Partition testing

- Input data and output results often fall into different classes where all members of a class are related.
- Each of these classes is an **equivalence partition** or domain where the program behaves in an equivalent way for each class member.
- Test cases should be chosen from each partition.

Stan Kurkovsky

## Structural testing

- Sometime called white-box testing.
- Derivation of test cases according to program structure. Knowledge of the program is used to identify additional test cases.
- Objective is to exercise all program statements (not all path combinations).

Stan Kurkovsky

## Path testing

- The objective of path testing is to ensure that the set of test cases is such that each path through the program is executed at least once.
- The starting point for path testing is a program flow graph that shows nodes representing program decisions and arcs representing the flow of control.
- Statements with conditions are therefore nodes in the flow graph.

Stan Kurkovsky

## Test automation

- Testing is an expensive process phase. Testing workbenches provide a range of tools to reduce the time required and total testing costs.
- Systems such as JUnit support the automatic execution of tests.
- Most testing workbenches are open systems because testing needs are organization-specific.
- They are sometimes difficult to integrate with closed design and analysis workbenches.

Stan Kurkovsky

## Key points

- Testing can show the presence of faults in a system; it cannot prove there are no remaining faults.
- Component developers are responsible for component testing; system testing is the responsibility of a separate team.
- Integration testing is testing increments of the system; release testing involves testing a system to be released to a customer.
- Use experience and guidelines to design test cases in defect testing.
- Interface testing is designed to discover defects in the interfaces of composite components.
- Equivalence partitioning is a way of discovering test cases - all cases in a partition should behave in the same way.
- Structural analysis relies on analyzing a program and deriving tests from this analysis.
- Test automation reduces testing costs by supporting the test process with a range of software tools.