Software Testing Basics

Adopted from:
Elaine Weyuker
AT&T Labs - Research

Dr. Panayotis Kikiras
INFS133

Most Common Software problems

- Incorrect calculation
- Incorrect data edits & ineffective data edits
- Incorrect matching and merging of data
- Data searches that yields incorrect results
- Incorrect processing of data relationship
- Incorrect coding / implementation of business rules
- Inadequate software performance

- Confusing or misleading data
- Software usability by end users & Obsolete Software
- Inconsistent processing
- Unreliable results or performance
- Inadequate support of business needs
- Incorrect or inadequate interfaces with other systems
- Inadequate performance and security controls
- Incorrect file handling

Objectives of testing

- Executing a program with the intent of finding an *error*.
- To check if the system meets the requirements and be executed successfully in the Intended environment.
- To check if the system is “ Fit for purpose”.
- To check if the system does what it is expected to do.

Objectives of testing

- A good test case is one that has a probability of finding an as yet undiscovered error.
- A successful test is one that uncovers a yet undiscovered error.
- A good test is not redundant.
- A good test should be “best of breed”.
- A good test should neither be too simple nor too complex.

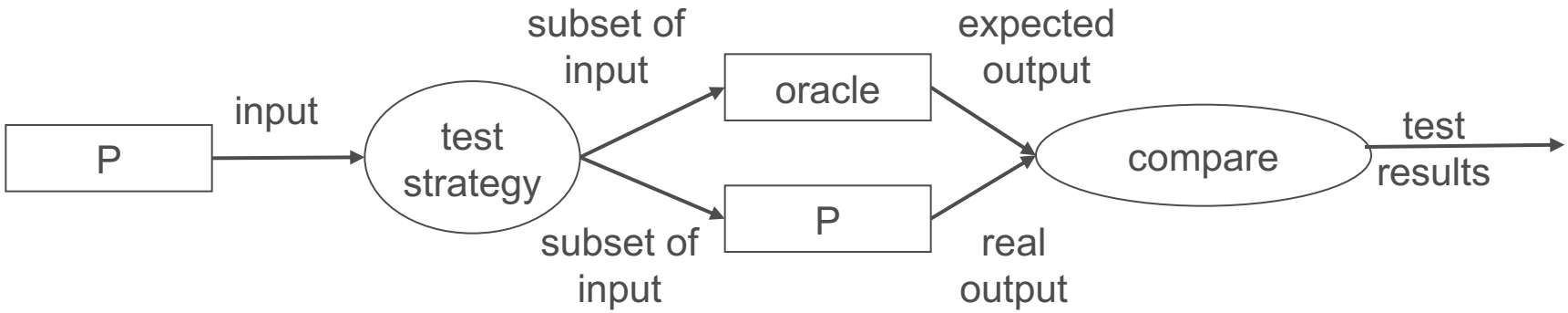
Objective of a Software Tester

- Find bugs as early as possible and make sure they get fixed.
- To understand the application well.
- Study the functionality in detail to find where the bugs are likely to occur.
- Study the code to ensure that each and every line of code is tested.
- Create test cases in such a way that testing is done to uncover the hidden bugs and also ensure that the software is usable and reliable

What is Software Testing?

Executing software in a simulated or real environment, using inputs selected *somehow*.

Testing process



Goals of Testing

- Detect faults
- Establish confidence in software
- Evaluate properties of software
  - Reliability
  - Performance
  - Memory Usage
  - Security
  - Usability

Software Testing Difficulties

Most of the software testing literature equates test case selection to software testing but that is just one difficult part. Other difficult issues include:

- Determining whether or not outputs are correct.
- Comparing resulting internal states to expected states.
- Determining whether adequate testing has been done.
- Determining what you can say about the software when testing is completed.
- Measuring performance characteristics.
- Comparing testing strategies.

Determining the Correctness of Outputs

We frequently accept outputs because they are *plausible* rather than correct.

It is difficult to determine whether outputs are correct because:

- We wrote the software to compute the answer.
- There is so much output that it is impossible to validate it all.
- There is no (visible) output.

Testing methodologies

Black box testing

White box testing

Incremental testing

Thread testing

- **Black box testing**
  - No knowledge of internal design or code required.
  - Tests are based on requirements and functionality
- **White box testing**
- Knowledge of the internal program design and code required.
- Tests are based on coverage of code statements,branches,paths,conditions.

Black Box - testing technique

- Incorrect or missing functions
- Interface errors
- Errors in data structures or external database access
- Performance errors
- Initialization and termination errors

Black box / Functional testing

- Based on requirements and functionality
- Not based on any knowledge of internal design or code
- Covers all combined parts of a system
- Tests are data driven

White box testing / Structural testing

- Based on knowledge of internal logic of an application's code
- Based on coverage of code statements, branches, paths, conditions
- Tests are logic driven

Functional testing

- Black box type testing geared to functional requirements of an application.
- Done by testers.

System testing

- Black box type testing that is based on overall requirements specifications; covering all combined parts of the system.

End-to-end testing

- Similar to system testing; involves testing of a complete application environment in a situation that mimics real-world use.

Sanity testing

- Initial effort to determine if a new software version is performing well enough to accept it for a major testing effort.

Regression testing

- Re-testing after fixes or modifications of the software or its environment.

Acceptance testing

- Final testing based on specifications of the end-user or customer

Load testing

- Testing an application under heavy loads.
- Eg. Testing of a web site under a range of loads to determine, when the system response time degraded or fails.

Stress Testing

- Testing under unusually heavy loads, heavy repetition of certain actions or inputs, input of large numerical values, large complex queries to a database etc.

- Term often used interchangeably with ‘load’ and ‘performance’ testing.

Performance testing

- Testing how well an application complies to performance requirements.

Install/uninstall testing

- Testing of full,partial or upgrade install/uninstall process.

Recovery testing

- Testing how well a system recovers from crashes, HW failures or other problems.

Compatibility testing

- Testing how well software performs in a particular HW/SW/OS/NW environment.

Exploratory testing / ad-hoc testing

- Informal SW test that is not based on formal test plans or test cases; testers will be learning the SW in totality as they test it.

Comparison testing

- Comparing SW strengths and weakness to competing products.

Alpha testing

- Testing done when development is nearing completion; minor design changes may still be made as a result of such testing.

Beta-testing

- Testing when development and testing are essentially completed and final bugs and problems need to be found before release.

Mutation testing

- To determining if a set of test data or test cases is useful, by deliberately introducing various bugs.
- Re-testing with the original test data/cases to determine if the bugs are detected.

White Box - testing technique

- All independent paths within a module have been exercised at least once
- Exercise all logical decisions on their *true* and *false* sides
- Execute all loops at their boundaries and within their operational bounds
- Exercise internal data structures to ensure their validity

Loop Testing

This white box technique focuses on the validity of loop constructs.

4 different classes of loops can be defined

- simple loops
- nested loops
- concatenated loops
- Unstructured loops

Other White Box Techniques

Statement Coverage – execute all statements at least once

Decision Coverage – execute each decision direction at least once

Condition Coverage – execute each decision with all possible outcomes at least once

Decision / Condition coverage – execute all possible combinations of condition outcomes in each decision.

Multiple condition Coverage – Invokes each point of entry at least once.

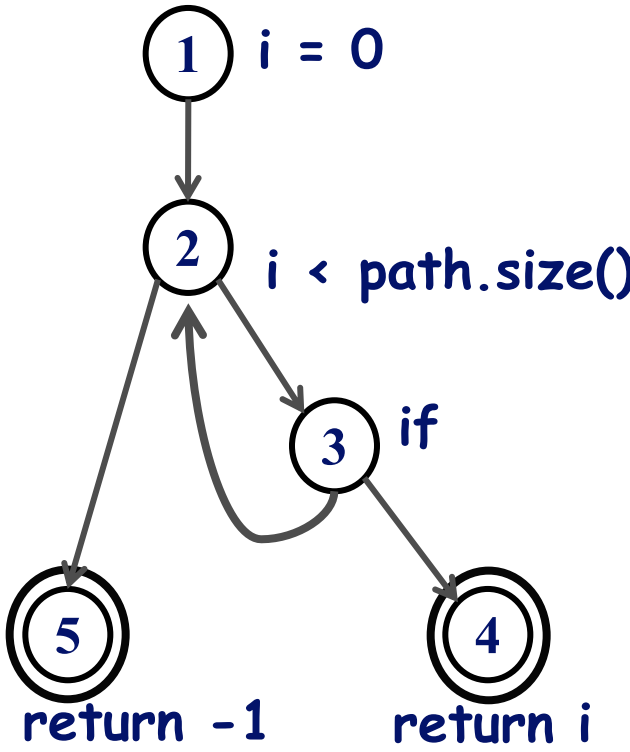
Statement Coverage – Examples

Small Illustrative Example

Software Artifact : Java Method

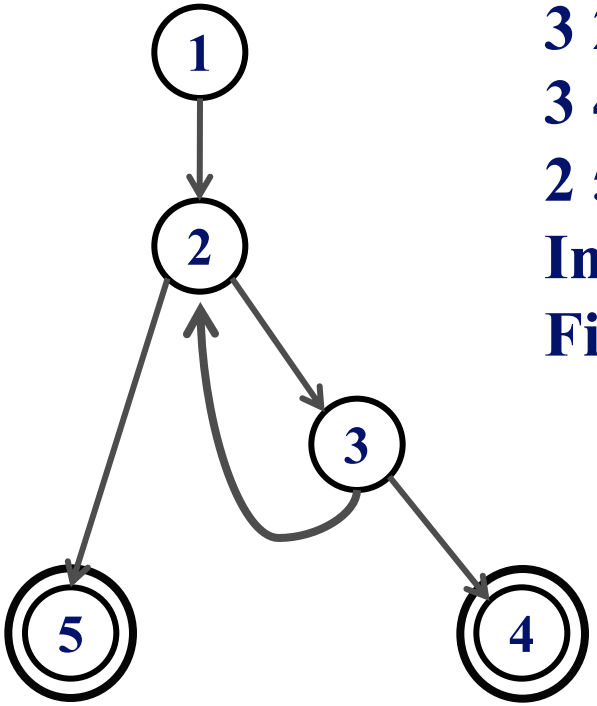
```
/**
 * Return index of node n at the
 * first position it appears,
 * -1 if it is not present
*/
public int indexOf (Node n)
{
  for (int i=0; i < path.size(); i++)
    if (path.get(i).equals(n))
      return i;
  return -1;
}
```

Control Flow Graph



Example (2)

Graph
Abstract version



Edges
1 2
2 3
3 2
3 4
2 5
Initial Node: 1
Final Nodes: 4, 5

6 requirements for Edge-Pair Coverage
1. [1, 2, 3]
2. [1, 2, 5]
3. [2, 3, 4]
4. [2, 3, 2]
5. [3, 2, 3]
6. [3, 2, 5]

Test Paths
[1, 2, 5]
[1, 2, 3, 2, 5]
[1, 2, 3, 2, 3, 4]

Find values …

Incremental Testing

- A disciplined method of testing the interfaces between unit-tested programs as well as between system components.
- Involves adding unit-testing program module or component one by one, and testing each result and combination.

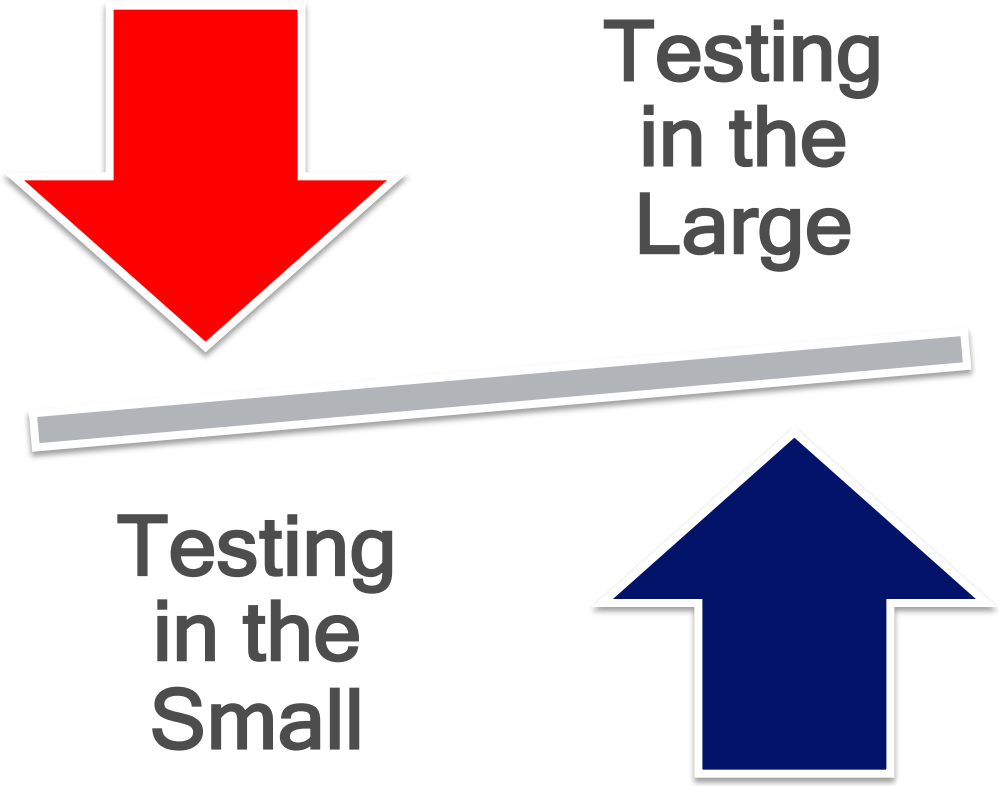
There are two types of incremental testing

- **Top-down** – testing form the top of the module hierarchy and work down to the bottom. Modules are added in descending hierarchical order.

- **Bottom-up** – testing from the bottom of the hierarchy and works up to the top. Modules are added in ascending hierarchical order.

Testing Levels/ Techniques	White Box	Black Box	Incre-mental	Thread
Unit Testing	x			
Integration Testing	x		x	x
System Testing		x		
Acceptance Testing		x		

Stages of Testing



Testing in the Small

- Unit Testing
- Feature Testing
- Integration Testing

Unit Testing

Tests the smallest individually executable code units.

Usually done by programmers. Test cases might be selected based on code, specification, intuition, etc.

Tools:

- Test driver/harness
- Code coverage analyzer
- Automatic test case generator

Integration Testing

Tests interactions between two or more units or components. Usually done by programmers. Emphasizes interfaces.

Issues:

- In what order are units combined?
- How do you assure the compatibility and correctness of externally-supplied components?

Integration Testing

How are units integrated? What are the implications of this order?

- Top-down => need stubs; top-level tested repeatedly.
- Bottom-up => need drivers; bottom-levels tested repeatedly.
- Critical units first => stubs & drivers needed; critical units tested repeatedly.

Integration Testing

Potential Problems:

- Inadequate unit testing.
- Inadequate planning & organization for integration testing.
- Inadequate documentation and testing of externally-supplied components.

Major Testing Types

- Stress / Load Testing
- Performance Testing
- Recovery Testing
- Conversion Testing
- Usability Testing
- Configuration Testing

Stress / Load Test

- Evaluates a system or component at or beyond

  the limits of its specified requirements.

- Determines the load under which it fails and

  how.

Performance Test

- Evaluate the compliance of a system or component with specified performance requirements.

- Often performed using an automated test tool to simulate large number of users.

Recovery Test

Confirms that the system recovers from expected or unexpected events without loss of data or functionality.

Eg.

- Shortage of disk space
- Unexpected loss of communication
- Power out conditions

Conversion Test

– Testing of code that is used to convert data from existing systems for use in the newly replaced systems

Usability Test

– Testing the system for the users to learn and use the product.

Configuration Test

- Examines an application's requirements for pre-existing software, initial states and configuration in order to maintain proper functionality.

SOFTWARE TESTING LIFECYCLE - PHASES

- Requirements study
- Test Case Design and Development
- Test Execution
- Test Closure
- Test Process Analysis

Requirements study

- Testing Cycle starts with the study of client's requirements.
- Understanding of the requirements is very essential for testing the product.

Analysis & Planning

- Test objective and coverage
- Overall schedule
- Standards and Methodologies
- Resources required, including necessary training
- Roles and responsibilities of the team members
- Tools used

Test Case Design and Development

- Component Identification
- Test Specification Design
- Test Specification Review

Test Execution

- Code Review
- Test execution and evaluation
- Performance and simulation

Test Closure

- Test summary report
- Project Documentation

Test Process Analysis

Analysis done on the reports and improving the application's performance by implementing new technology and additional features.

TEST PLAN

Objectives

- To create a set of testing tasks.
- Assign resources to each testing task.
- Estimate completion time for each testing task.
- Document testing standards.

- A document that describes the
  - scope
  - approach
  - resources
  - schedule
- …of intended test activities.
- Identifies the
  - test items
  - features to be tested
  - testing tasks
  - task allotment
  - risks requiring contingency planning.

Purpose of preparing a Test Plan

- Validate the acceptability of a software product.

- Help the people outside the test group to understand ‘why’ and ‘how’ of product validation.

- A Test Plan should be
  - thorough enough (Overall coverage of test to be conducted)
  - useful and understandable by the people inside and outside the test group.

Scope

- The areas to be tested by the QA team.
- Specify the areas which are out of scope (screens, database, mainframe processes etc).

Test Approach

- Details on how the testing is to be performed.
- Any specific strategy is to be followed for testing (including configuration management).

Entry Criteria

Various steps to be performed before the start of a test i.e. Pre-requisites.

E.g.

- Timely environment set up
- Starting the web server/app server
- Successful implementation of the latest build etc.

Resources

List of the people involved in the project and their designation etc.

Tasks/Responsibilities

Tasks to be performed and responsibilities assigned to the various team members.

Exit Criteria

Contains tasks like

- Bringing down the system / server
- Restoring system to pre-test environment
- Database refresh etc.

Schedule / Milestones

Deals with the final delivery date and the various milestones dates.

Hardware / Software Requirements

- Details of PC's / servers required to install the application or perform the testing
- Specific software to get the application running or to connect to the database etc.

Risks & Mitigation Plans

- List out the possible risks during testing
- Mitigation plans to implement incase the risk actually turns into a reality.

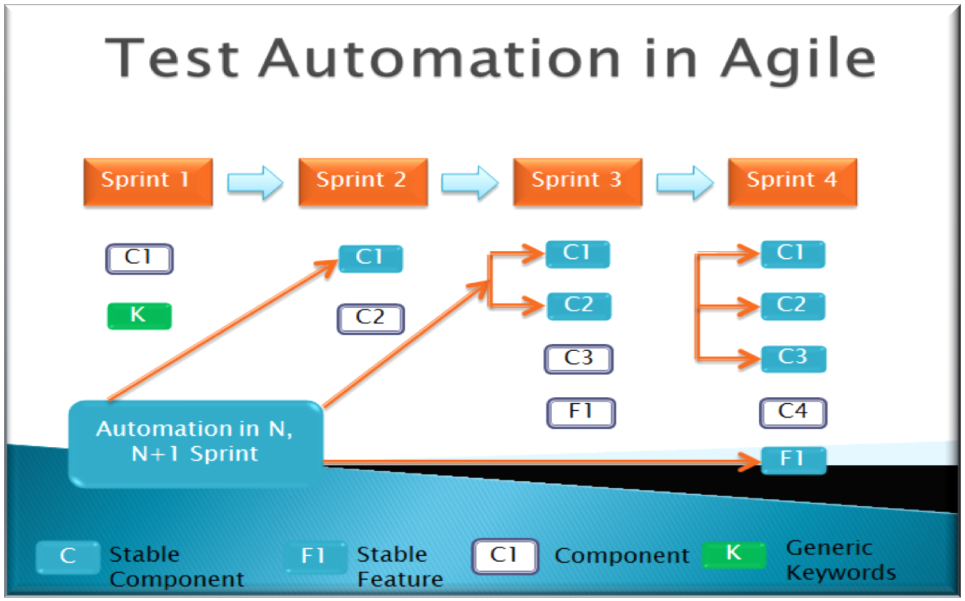
Tools to be used

- List the testing tools or utilities
- Eg.WinRunner, LoadRunner, Test Director, Rational Robot, QTP.

Deliverables

- Various deliverables due to the client at various points of time i.e. Daily / weekly / start of the project end of the project etc.
- These include test plans, test procedures, test metric, status reports, test scripts etc.

Testing in Agile



Automation is required to determine the stability of module developed in each Sprint as depicted in Figure above.

Furthermore, with automation ready, module developed in Sprint 1 can be tested up to 'n' sprints without incurring any cost in terms of manual testing.

All the defects that are found are fixed on priority. Automation can be carried at different phases in agile. i.e. Unit, Integration, System, Regression.

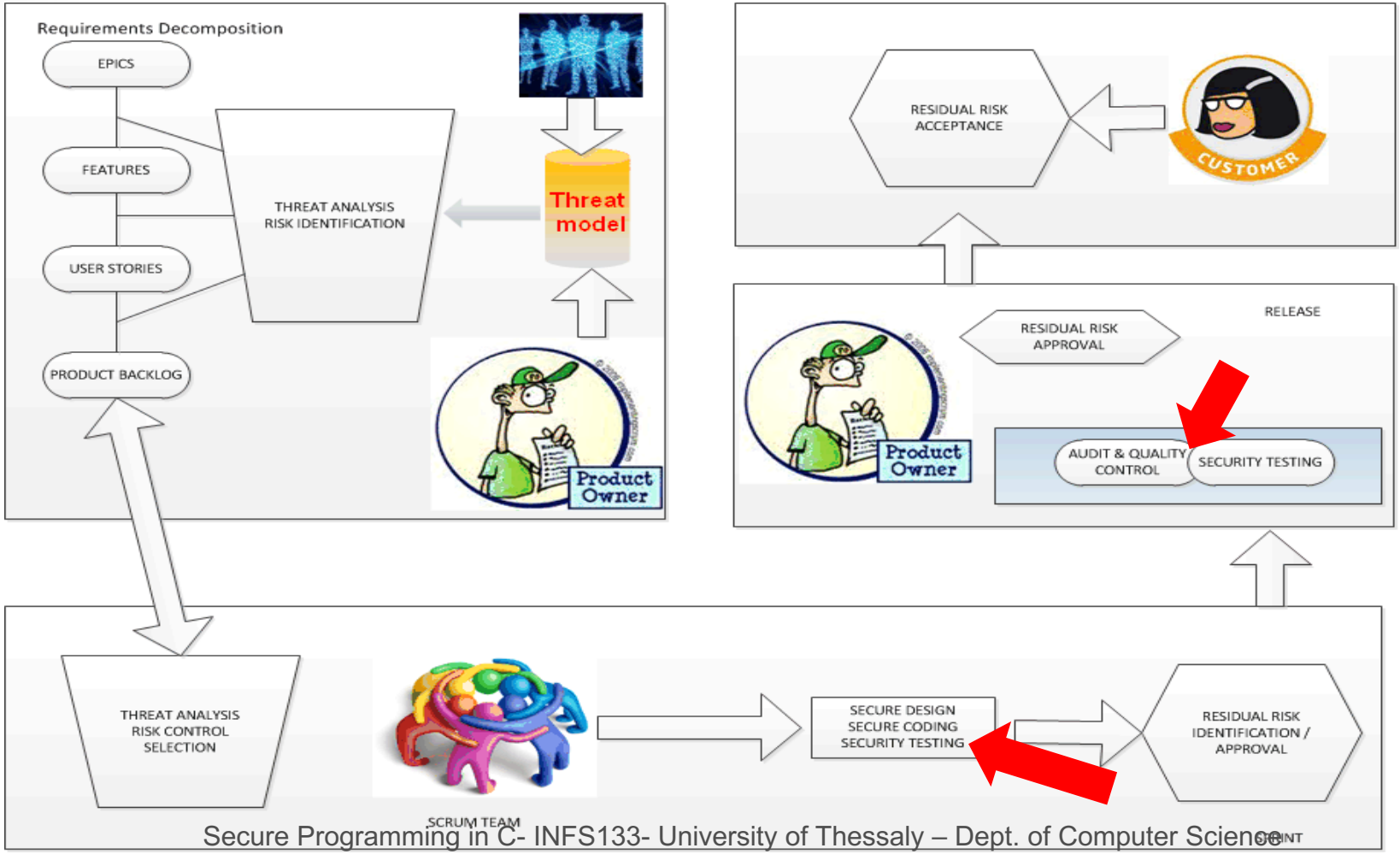
Prioritizing Test Cases

Once a test suite has been selected, it is often desirable to prioritize test cases based on some criterion. That way, since the time available for testing is limited and therefore all tests can't be run, at least the "most important" ones can be.

Bases for Test Prioritization

- Most frequently executed inputs.
- Most critical functions.
- Most critical individual inputs.
- (Additional) statement or branch coverage.
- (Additional) Function coverage.
- Fault-exposing potential.

Securing SCRUM
- Testing



Summary

- Do test as early as possible
- Testing is a continuous process
- Design with testability in mind
- Test activities must be carefully planned, controlled and documented.
- No single reliability model performs best consistently


Software Testing
The Fine Art Of
Screwing Stuff Up