Software Testing Preliminaries

Dr. Mark C. Paulk

SE 4367 – Software Testing, Verification, Validation, and Quality Assurance

Jonsson School of Engineering and Computer Science

Topics: Software Testing

Part I: Preliminaries

1. Software Testing

- **Humans, Errors, and Testing**
- **Software Quality**
- **Requirements, Behavior, and Correctness**
- **Correctness vs Reliability**
- ***Testing and Debugging***
- ***Test Metrics***
- ***Software and Hardware Testing***
- ***Testing and Verification***
- ***Defect Management***
- ***Test Generation Strategies***
- ***Static Testing***
- ***Model-Based Testing and Model Checking***
- ***Types of Testing***
- ***Saturation Effect***
- ***Principles of Testing***

The Purpose of Testing

What is the purpose of testing?

To demonstrate that the software works?

NO!

To find defects in the software?

YES!

There is always the possibility of latent defects…

Verification & Validation

Verification

"Are we building the product right".

The software should conform to its specification.

Validation

"Are we building the right product".

The software should do what the user really requires.

V&V in IEEE 729:1983

Verification

- **The process of determining whether or not the products of a given phase of the software development cycle fulfill the requirements established during the previous phase.**

Validation

- **The process of evaluating software at the end of the software development process to ensure compliance with software requirements.**

Three Mechanisms for V&V

Testing

- **"test" can be a generic word – see Weinberg's Perfect Software and Other Illusions About Testing**
  - **does (not) meet need**
  - **does (not) meet all requirements**
  - **(un)acceptable costs or constraints**
  - **grossly unreliable**
  - **poor performance**
- **by testing, we typically assume dynamic execution of a program**
  - **black box, white box, unit, integration, system, regression, etc., testing**

Formal methods

- formal specifications
- model / property checking
- proofs of correctness
  - Correctness attempts to establish the program is error-free; testing attempts to find if there are any errors in the program.

Peer reviews

- A review of a software work product, following defined procedures, by peers of the producers of the product for the purpose of identifying defects and improvements. (Software CMM)
  - managers are not peers…
  - customers are not peers…

Generic V&V Techniques

- Technical reviews, management reviews, joint reviews
- Symbolic execution, program proving, proof of correctness, formal methods
- Anomaly analysis, syntactical checks
- Functional testing, black-box testing, equivalence partitioning, boundary value analysis
- Structural testing, white-box testing, basis path testing, condition testing, data flow testing, loop testing
- Unit testing
- Regression testing, daily build and smoke test
- Integration testing
- Random testing, adaptive perturbation testing, mutation testing, be-bugging
- Operational profile testing, stress testing, performance testing
- System testing, acceptance testing
- Peer reviews, structured walkthroughs, inspections, active design reviews, pair programming, …

Correctness vs Testing

Correctness is established via mathematical proofs of programs.

Proofs are precise but subject to human fallibility.

"Beware of bugs in the above code; I have only proved it correct, not tried it."
- Donald Knuth, 1977

Correctness attempts to establish the program is error-free; testing attempts to find if there are any errors in the program.

"Program testing can be used to show the presence of bugs, but never to show their absence!"
- Edsger Dijkstra, "Notes on Structured Programming," 1970

Quality

1) **the degree to which a system, component, or process meets specified requirements.**
2) **ability of a product, service, system, component, or process to meet customer or user needs, expectations, or requirements.**
3) **the totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs.**
4) **conformity to user expectations, conformity to user requirements, customer satisfaction, reliability, and level of defects present.**
5) **the degree to which a set of inherent characteristics fulfils requirements.**
6) **the degree to which a system, component, or process meets customer or user needs or expectations.**

IEEE 24765: 2010 (Systems and Software Engineering – Vocabulary)

Also see R. Glass, "Defining Quality Intuitively," IEEE Software, May/June 1998.

Counting Defects

When programmers make mistakes, the defects injected are on the order of 10% of the SLOC.

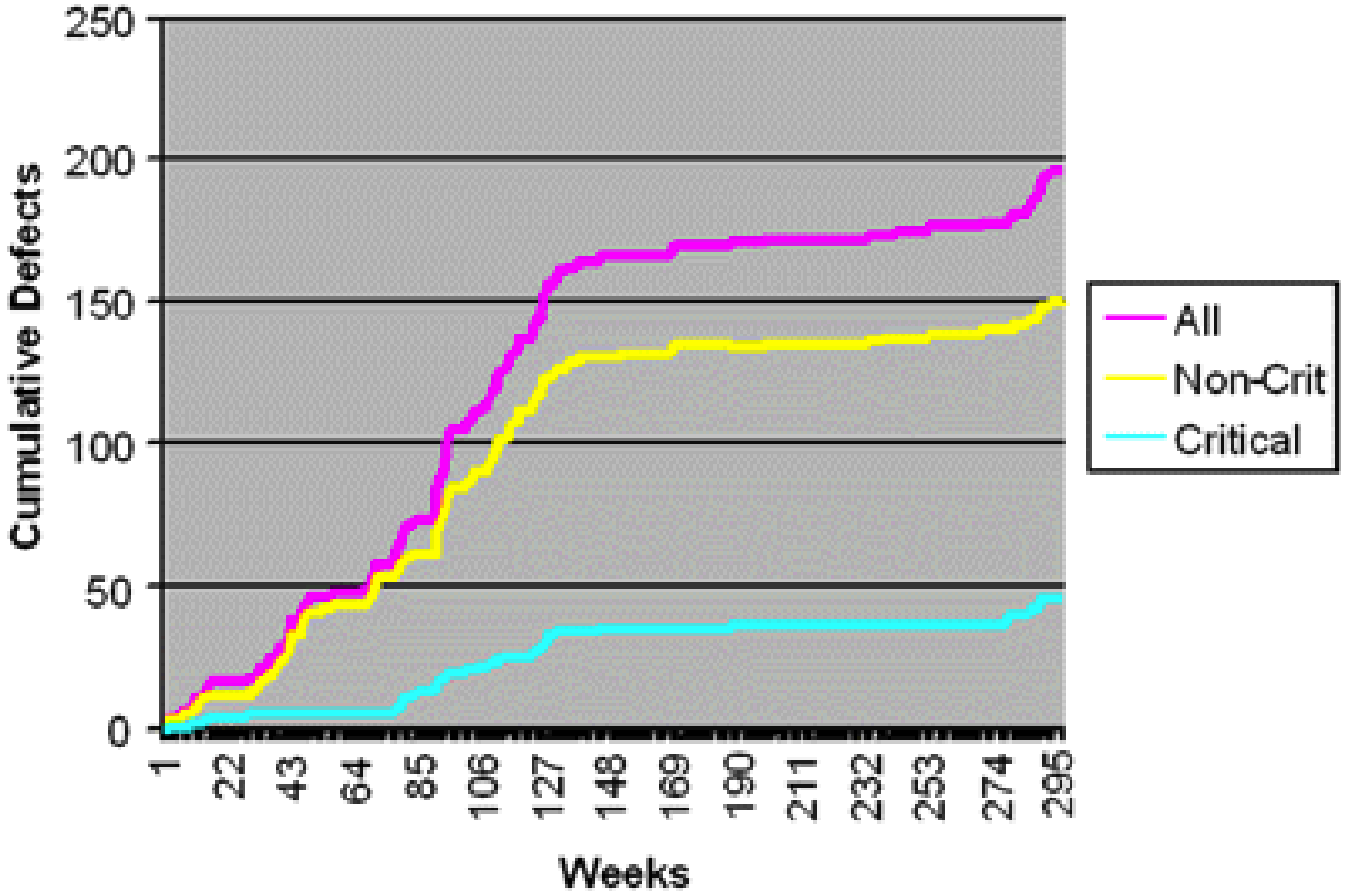
- **over 100 defects / KSLOC from PSP**

The "normal" development process will remove ~50% of defects before they are ever counted.

Defects will be distributed among requirements, design, code, ...

- **there will be ripple effects from defects injected earlier in the process**

Galileo System Test Defects



Galileo Testing

Mission: Jupiter Orbit
Launched: October 1989
Size of software: 22 KSLOC

Only one critical defect was found in the first 10 weeks of system testing.

- **five critical defects were found in the first 76 weeks of testing**
- **after nearly six years of testing, three critical defects were found in eight weeks**

Testing shows the presence of bugs, not the absence of bugs…

Subtle Distinctions

A person makes a mistake

Which becomes a fault (bug, defect)

Resulting at execution in a failure

Which is wrong with some degree of error

From the IEEE definition of fault tolerance…

Testing

Test: an activity in which a system or component is executed under specified conditions, the results are observed or recorded, and an evaluation is made of some aspect of the system or component.

- **to conduct an activity as in (1).**
- **a set of one or more test cases and procedures.**

Testing: activity in which a system or component is executed under specified conditions, the results are observed or recorded, and an evaluation is made of some aspect of the system or component.

IEEE Std 24765: 2010

Input Domain

The set of all possible inputs to a program P is known as the input domain, or input space, of P.

- **includes both valid and invalid inputs**

Testing a program on all possible inputs is known as exhaustive testing.

A program is considered correct if it behaves as expected on each element of its input domain.

Test Case

A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.

- **IEEE 24765**

A pair consisting of test data to be input to the program and the expected output.

- **Mathur, pg 18**
- **do we usually write the expected output down as part of our test case?**

Test set – a collection of test cases.

Test Case (IEEE 29119-1:2013)

Set of test case preconditions, inputs (including actions, where applicable), and expected results, developed to drive the execution of a test item to meet test objectives, including correct implementation, error identification, checking quality, and other valued information

- **Note 1 to entry: A test case is the lowest level of test input (i.e. test cases are not made up of test cases) for the test subprocess for which it is intended.**
- **Note 2 to entry: Test case preconditions include test environment, existing data (e.g. databases), software under test, hardware, etc.**
- **Note 3 to entry: Inputs are the data information used to drive test execution.**
- **Note 4 to entry: Expected results include success criteria, failures to check for, etc.**

Test Set and Test Procedure

Test set *(IEEE 29119-1:2013)*

- **set of one or more test cases with a common constraint on their execution**
  - **EXAMPLE: A specific test environment, specialized domain knowledge or specific purpose.**

Test procedure *(IEEE 29119-1:2013)*

- **sequence of test cases in execution order, and any associated actions that may be required to set up the initial preconditions and any wrap up activities post execution**
  - **Note 1 to entry: Test procedures include detailed instructions for how to run a set of one or more test cases selected to be run consecutively, including set up of common preconditions, and providing input and evaluating the actual result for each included test case.**

Functional (Black-Box) Testing

Functional testing

- **testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions**
- **testing conducted to evaluate the compliance of a system or component with specified functional requirements (IEEE 24765)**
  - **Synonymous with: specification-based, closed-box testing**

Specification-based testing

- **testing in which the principal test basis is the external inputs and outputs of the test item, commonly based on a specification, rather than its implementation in source code or executable software *(IEEE 29119-1:2013)***

Black-Box Measures

Have you covered all the requirements with at least one test case?

- **prerequisite: a requirements "specification" with clearly stated and labeled requirements**
- **how would "user stories" (from agile methods) fit with this prerequisite?**

Measure – percentage of requirements covered

Have you covered all combinations of (independent) requirements?

Other black-box testing measures:

- **percentage of equivalence classes covered**
- **percentage of boundary values covered**

Structural (White-Box) Testing

Structural testing – Testing that takes into account the internal mechanism of a system or component.

- **aka white-box testing, glass-box testing**

NOTE Types include branch testing, path testing, statement testing.

- **Branch testing – Testing designed to execute each outcome of each decision point in a computer program.**
- **Path testing – Testing designed to execute all or selected paths through a computer program.**
- **Statement testing – Testing designed to execute each statement of a computer program.**

IEEE 24765

White-Box Testing Measures

Have you covered every statement in the program with at least one test case?

- **don't need to include syntactical markers, e.g., else, {, }, end**

Have you covered both branches of every decision in the program with at least one test case?

- **decisions include loops, ifs, switches**

Have you covered both branches of every condition in the program with at least one test case?

- **multiple conditions in compound predicates: AND, OR, …**

Measure – percentage of statements, decisions, conditions covered

Test Coverage Example

```
1)   // Program P1
2)   integer x, y;
3)   input (x, y);
4)   if (x > 0 && y > 0)
5)   {
6)      y = y / x;
7)   }
8)   else
9)   {
10)     y = x ** 2;
11)  }
12)  output (x, y);
13)  end;
```

What is the statement coverage for t_1 = (1, 2)?

- **do not count syntactical markers, including comments, {, }, else, end**

The decision coverage?

The condition coverage?

```
1)    // Program P1
2)    integer x, y;
3)    input (x, y);
4)    if (x > 0 && y > 0)
5)    {
6)       y = y / x;
7)    }
8)    else
9)    {
10)      y = x ** 2;
11)   }
12)   output (x, y);
13)   end;
```

Statement coverage for t_1 = (1, 2)?

- **Domain: {2, 3, 4, 6, 10, 12}**
- **t_1 traverses 2, 3, 4, 6, 12**
- **Statement Coverage = 5 / 6**

Decision coverage for t_1?

- **Decision: 4) if (x > 0 && y > 0)**
- **t_1 traverses the true branch**
- **Decision Coverage = 0 / 1**

Condition coverage for t_1?

- **Conditions: 4) x > 0; 4) y > 0**
- **t_1 traverses the true branch of the two conditions**
- **Condition Coverage = 0 / 2**

Levels of Testing

Unit testing

Integration testing

Product (system) testing

Acceptance testing

Alpha release

Beta release

Unit vs Integration Testing

Unit testing

- **testing of individual routines and modules by the developer or an independent tester**
- **a test of individual programs or modules in order to ensure that there are no analysis or programming errors**

Integration testing

- **testing in which software components, hardware components, or both are combined and tested to evaluate the interaction among them**

(ISO/IEC/IEEE 24765)

System vs Acceptance Testing

System testing

- **testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements**

Acceptance testing

- **testing conducted to determine whether a system satisfies its acceptance criteria and to enable the customer to determine whether to accept the system**
- **formal testing conducted to enable a user, customer, or other authorized entity to determine whether to accept a system or component.**

(ISO/IEC/IEEE 24765)

Regression Testing

Selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements.

Testing required to determine that a change to a system component has not adversely affected functionality, reliability or performance and has not introduced additional defects.

IEEE 24765

Peer Reviews

A review of a software work product, following defined procedures, by peers of the producers of the product for the purpose of identifying defects and improvements.

- **Software CMM**

Review of work products performed by peers during development of the work products to identify defects for removal.

- **IEEE 24765**

Walkthroughs

Structured walkthrough – A systematic examination of the requirements, design, or implementation of a system, or any part of it, by qualified personnel.

IEEE 24765

See Weinberg's The Psychology of Computer Programming for a discussion of walkthroughs and egoless programming.

Inspections

1) **A visual examination of a software product to detect and identify software anomalies, including errors and deviations from standards and specifications.**
2) **A static analysis technique that relies on visual examination of development products to detect errors, violations of development standards, and other problems**
3) **Technique examining or measuring to verify whether an activity, component, product, result, or service conforms to specified requirements.**

NOTE Inspections are peer examinations led by impartial facilitators who are trained in inspection techniques. Determination of remedial or investigative action for an anomaly is a mandatory element of a software inspection, although the solution should not be determined in the inspection meeting. Types include code inspection; design inspection

IEEE 24765

Formal Methods

A technique for expressing requirements in a manner that allows the requirements to be studied mathematically.

Formal methods allow sets of requirements to be examined for completeness, consistency, and equivalency to another requirement set.

Formal methods result in formal specifications.

EIA 731.1: 2002 (Systems Engineering Capability Model)

- **definition focused on formal specifications but not proofs of correctness**

Quality Assurance

All the planned and systematic activities implemented within the quality system, and demonstrated as needed, to provide adequate confidence that an entity will fulfill requirements for quality. (ISO 12207)

Note 1 - There are both internal and external purposes for quality assurance: a) Internal quality assurance: within an organization, quality assurance provides confidence to management; b) External quality assurance: in contractual situations, quality assurance provides confidence to the customer or others.

Note 2 - Some quality control and quality assurance actions are interrelated.

Note 3 - Unless requirements for quality fully reflect the needs of the user, quality assurance may not provide adequate confidence.

Developer and Tester as Two Roles

Distinct but complementary roles

- **developers have a conflict of interest in testing**

A developer may assume the role of tester at some points in the software process

Some testers may be "black hats"

- **an independent testing group**
- **distinct testing skills**
- **make no assumptions about the software**
- **actively try to break the software**

Problem Reports

Problem report number
Program
Release
Version
Report type (coding error, design issue, …)
Severity (fatal, serious, minor, …)
Attachments
Problem summary
Can you reproduce the problem?
Problem and how to reproduce it
Suggested fix (optional)
Reported by
Date
Comments
Status
Priority
Resolution
Resolution version
Resolved by (date)
Resolution tested by (date)
Treat as deferred (y/n)

IEEE 1044, "IEEE Standard Classification for Software Anomalies," 1993.

Summary – Things to Remember

Purpose of testing

Testing vs correctness

Fault tolerance terminology – subtle distinctions

Black box vs white box

Kinds of testing – unit, integration, system, acceptance, regression

Conflict of interest for developers

Questions and Answers