Common System and Software Testing Pitfalls

Donald Firesmith

Software Solutions Conference 2015
November 16–18, 2015

Software Engineering Institute | Carnegie Mellon University

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM-0001886.

Agenda

Introduction

- Presentation Scope
- What is Testing?
- Why Test?
- Limitations and Challenges of Testing

Testing Pitfalls:

- Software vs. System Testing
- Pitfall Taxonomy and Ontology
- Addressing Testing Challenges
- Goals and Potential Uses
- Example Pitfall

Taxonomy of Common Testing Pitfalls (lists of pitfalls by category):

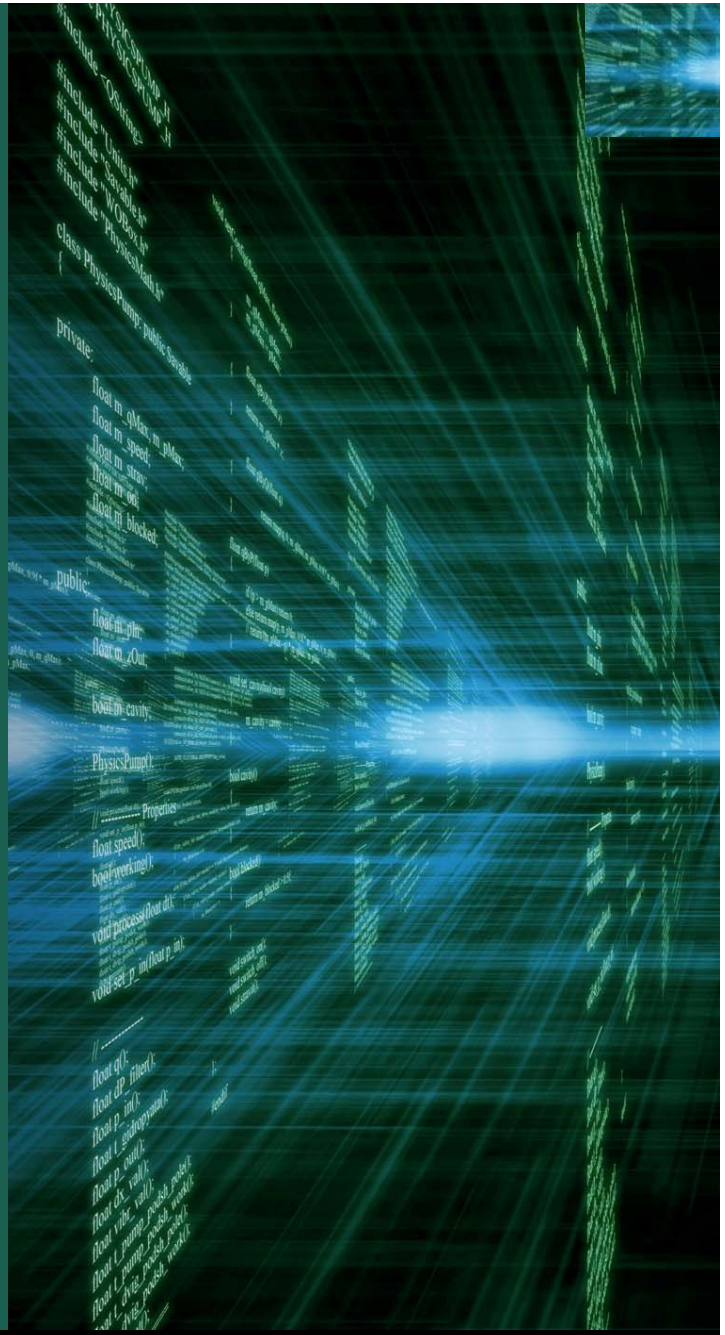
- General Pitfalls
- Test-Type-Specific Pitfalls

Conclusion

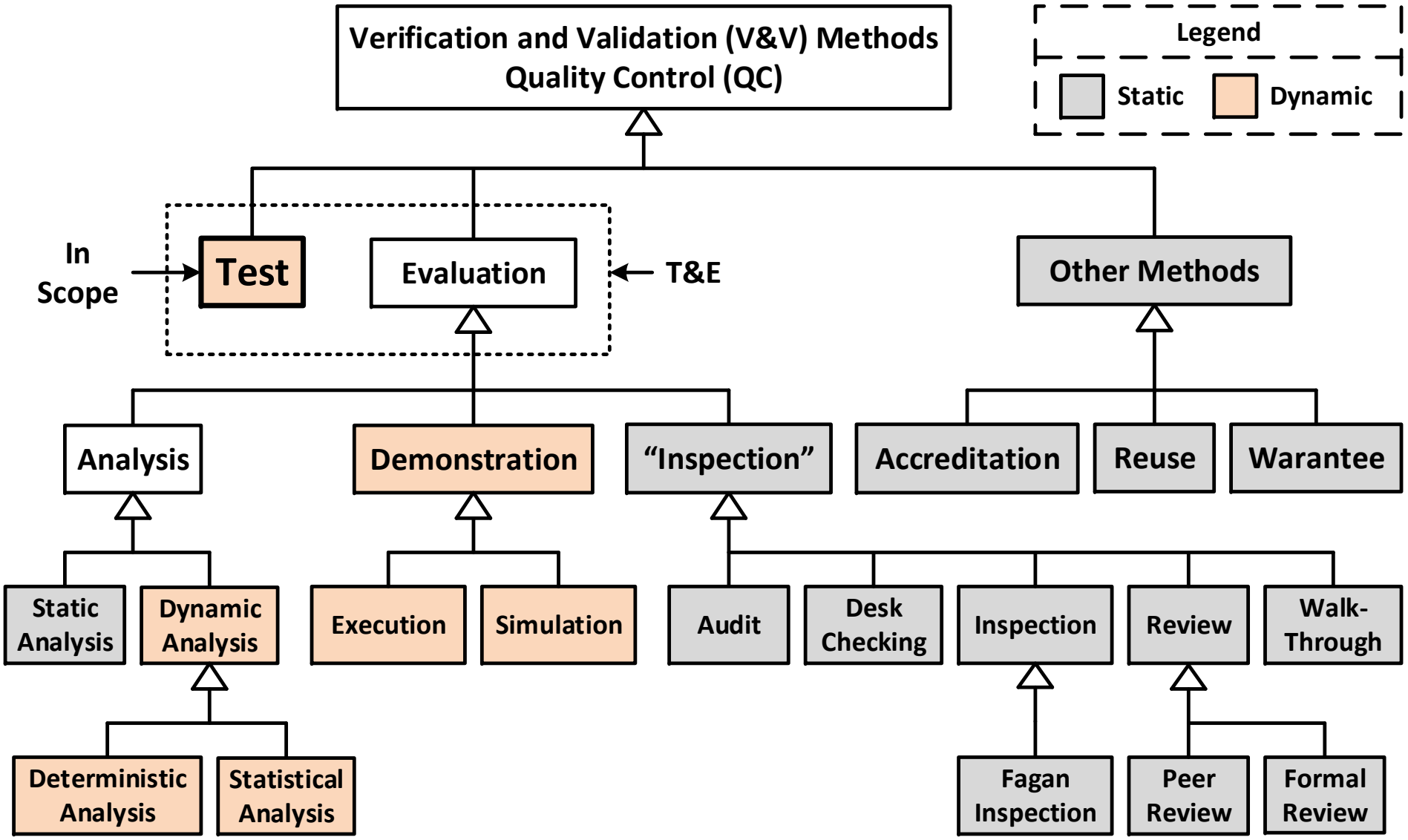
- Remaining Limitations and Questions
- Future Work

A Taxonomy of System & Software Testing Types

Introduction



Presentation Scope



What is Testing?

Testing

The ***execution*** of an Object Under Test (OUT) under specific ***preconditions*** with specific ***stimuli*** so that its ***actual behavior*** can be compared with its ***expected or required behavior***

- *Preconditions:* pretest mode, states, stored data, or external conditions
- *Stimuli:*
  - Calls, commands, and messages (control flows)
  - Data inputs (data flows)
  - Trigger events such as state changes and temporal events
- *Actual Behavior:*
  - *During Test:*
    - Calls, commands, and messages (control flows)
    - Data outputs (data flows)
  - *Postconditions*: post-test mode, states, stored data, or external conditions

What is Testing? – 2

Testing requires:

- ***Controllability*** to:
  - Establish preconditions
  - Provide input stimuli
- ***Observability*** to verify:
  - Establishment of preconditions
  - Correctness of stimuli
  - Correctness of outputs
  - Correctness of postconditions

Why Test?

In roughly decreasing order of importance, the goals of testing are to:

- **Uncover Significant Defects** in the object under test (OUT) by causing it to behave incorrectly (e.g., to fail or enter a faulty state) so that these underlying defects can be identified and fixed and the OUT can thereby be improved.
- **Provide Evidence** that can be used to determine the OUT's:
  - Quality
  - Fitness for purpose
  - Readiness for shipping, deployment, or being placed into operation
- **Support Process Improvement** by helping to identify:
  - Development processes that introduce defects
  - Testing processes that fail to uncover defects
- **Prevent Defects** by:
  - Testing executable requirements, architecture, and design models so that defects in the models are fixed before they can result in defects in the system/software.
  - Using Test Driven Development (TDD) to develop test cases and then using these test cases to drive development (design and implementation)

Limitations of Testing

Cannot be exhaustive (or even close)

Cannot uncover all defects because different types of testing:

- Have different defect removal efficiencies (DREs)
- Uncover different types of defects

May provide false positive and false negative results due to:

- Defects in the test case (e.g., incorrect test input or incorrect test data)
- Defects in the test environment(s)
- Poor configuration management of the OUT, the test environment, and the test case

Cannot prove the OUT works properly under all inputs and conditions

Testing-Pitfall-Related Challenges

A great many different ways exist to screw up testing.

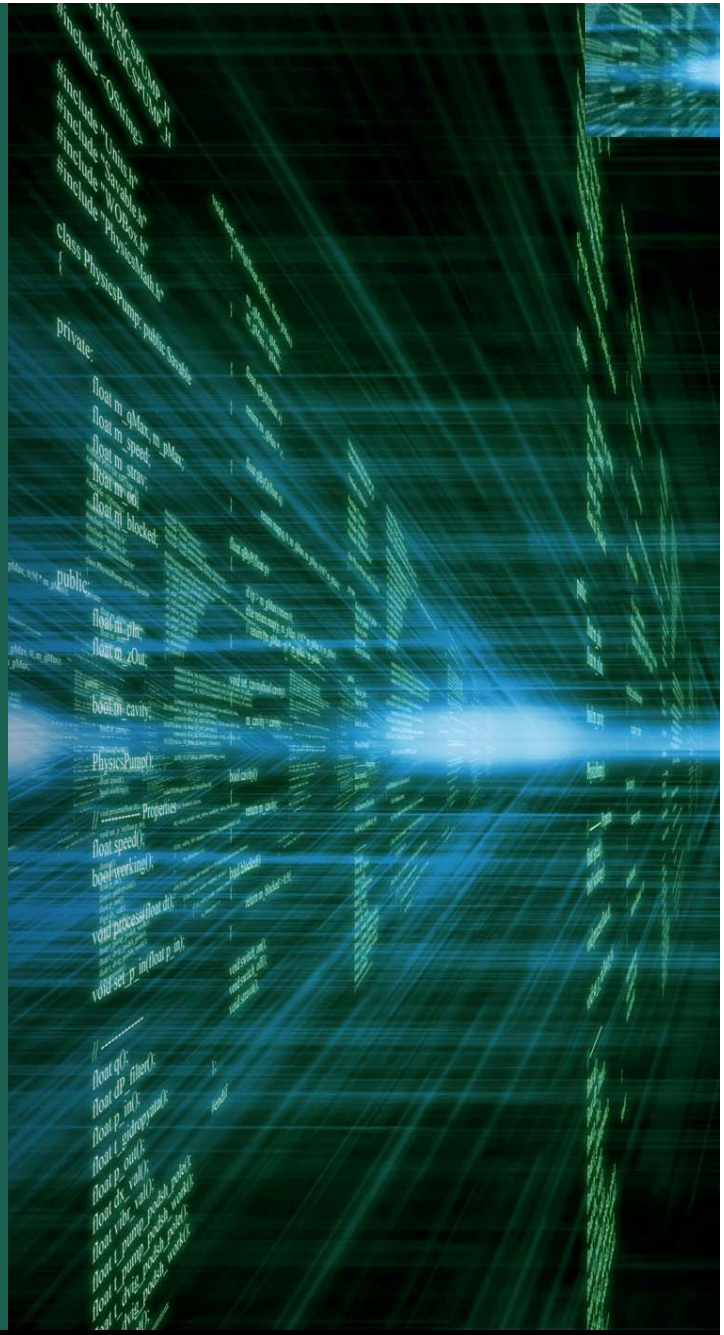
Multiple testing pitfalls are observed on just about every project.

Different programs often exhibit different testing pitfalls.

In spite of many excellent how-to testing books, we see projects falling into these same testing pitfalls over and over again.

A Taxonomy of System & Software Testing Types

Testing Pitfalls



Testing Pitfalls

Testing Pitfall

- A *human mistake* that unnecessarily and unexpectedly causes testing to be:
  - Less effective at uncovering significant defects
  - Less efficient in terms of time and effort expended
  - Less satisfying and fun to perform
- A bad decision, an incorrect mindset, a wrong action, or failure to act
- A *failure* to adequately:
  - Meet a testing challenge
  - Address a testing problem
- A way to screw up testing

Common Testing Pitfall

- Observed numerous times on different projects
- Has sufficient frequency (and consequences) to be a significant risk

Software vs. *System* Testing

These testing pitfalls occur when testing:

- Software (applications, components, units)
- Systems (subsystems, hardware, software, data, personnel, facilities, equipment, documentation, etc.)
- Executable models (requirements, architecture, design)

Most pitfalls apply to ***both*** software and system testing.

The vast majority of ***software*** testers ***must*** address ***system*** testing issues:

- Software executes on hardware, and how well it executes depends on:
  - That hardware
  - Other software running on the same hardware
- Software communicates over:
  - "External" networks (Internet, NIPRNet, SIPRNet, WAN, LAN, MAN, etc.)
  - Data-center-internal networks connecting servers and data libraries (e.g., SAN)
  - Busses within systems (embedded software)
- Software must meet quality requirements (thresholds of relevant quality characteristics and attributes) that are actually system-level, not software-level.

Pitfall Taxonomy and Ontology

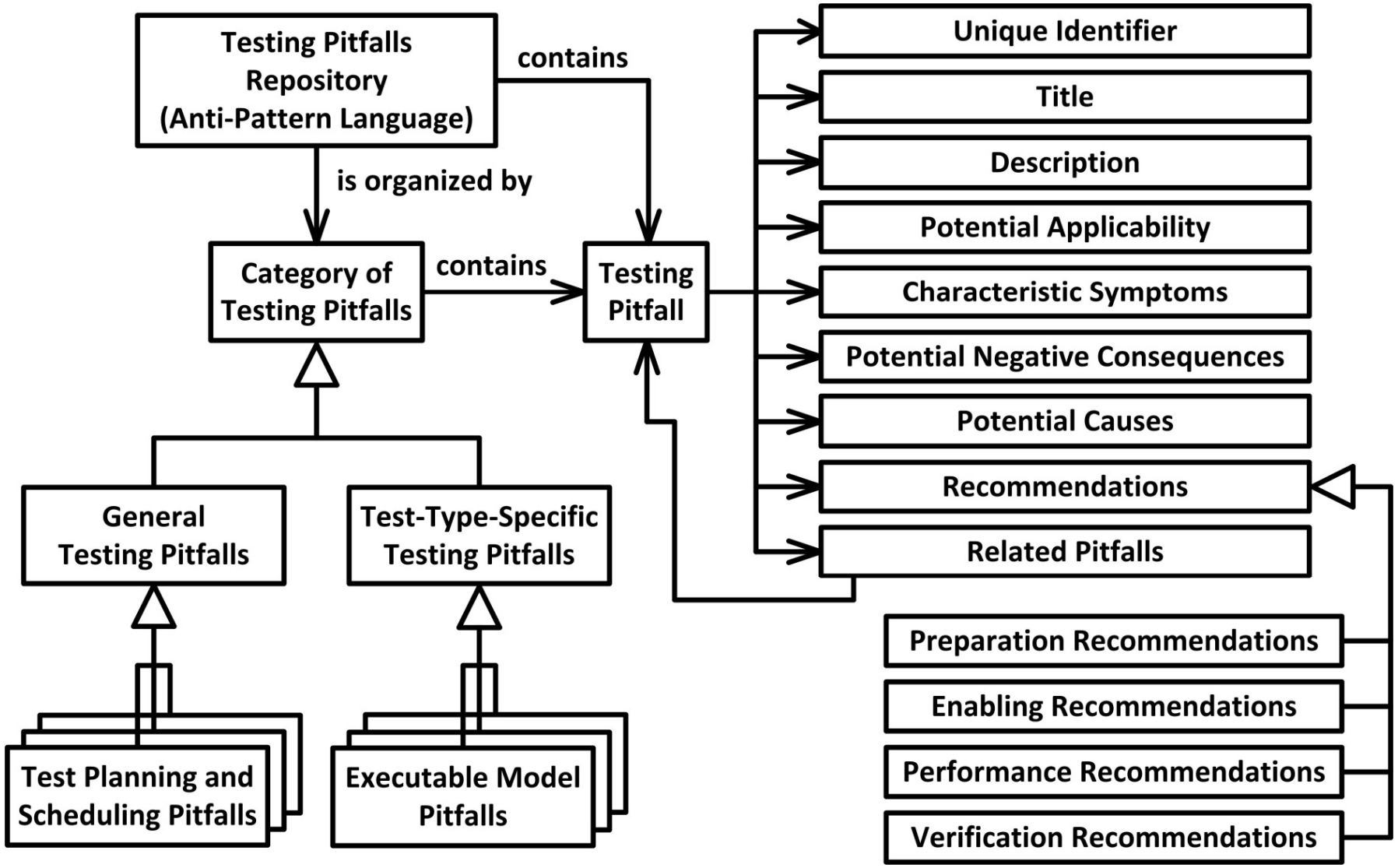
Testing Pitfall ***Taxonomy***

A hierarchical classification of testing pitfalls into categories and subcategories

Testing Pitfall ***Ontology***

A hierarchy of concepts concerning testing pitfalls that uses a shared vocabulary to denote the types, properties, and interrelationships of these concepts

Testing Pitfall Taxonomy and Ontology



Addressing these Challenges

Testing Pitfalls Taxonomy and Ontology

Anti-Pattern Language of how-not-to do testing

Common System and Software Testing Pitfalls (Addison-Wesley, 2014)

(Note: 35% conference discount)

- ***92*** pitfalls classified into ***14*** categories
- Technically reviewed by 47 international testing SMEs

Current taxonomy/repository with new pitfalls and pitfall categories:

- ***176*** pitfalls classified into ***24*** categories
- SEI Testing Pitfalls Wiki (currently under development)

Goals

To become the de facto industry-standard taxonomy of testing pitfalls

To reduce the incidence of testing pitfalls and thereby improve testing effectiveness and efficiency

To improve the quality of the objects under test (OUTs)

Potential Uses

Training materials for testers and testing stakeholders

Standard terminology regarding commonly occurring testing pitfalls

Checklists for use when:

- Producing test strategies/plans and related documentations
- Evaluating contractor proposals
- Evaluating test strategies/plans and related documentation (quality control)
- Evaluating as-performed test process (quality assurance)
- Identifying test-related risks and their mitigation approaches

Knowledge base for relatively simple ***test pitfall advice tool***

Categorization of pitfalls for ***test metrics*** collection, analysis, and reporting

Example – Testing and Engineering Processes Not Integrated (GEN-PRO-7)

Description The testing process(es) are not adequately integrated into the overall system/software engineering process(es).

Potential Applicability This pitfall is potentially applicable anytime that engineering and testing processes both exist.

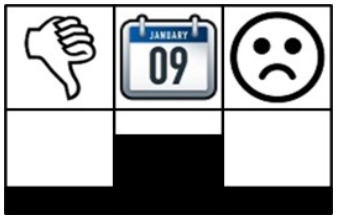
Example – Testing and Engineering Processes Not Integrated (GEN-PRO-7)

Characteristic Symptoms

- There is little or no discussion of testing in the system engineering documentation: System Engineering Management Plan (SEMP), Software Development Plan (SDP), Work Breakdown Structure (WBS), Project Master Schedule (PMS), or System Development Cycle (SDC).
- All or most of the testing is done as a completely independent activity performed by testers who are not part of the project engineering team.
- Testing is treated as a separate specialty engineering activity with only limited interfaces with the primary engineering activities.
- Test scheduling is independent of the scheduling of other development activities.
- Testers are not included in the requirements teams, architecture teams, or any cross-functional engineering teams.
- Testers are not invited to engineering reviews or walk-throughs.

Example – Testing and Engineering Processes Not Integrated (GEN-PRO-7)

Potential Negative Consequences



- Few nontesters understand the scope, complexity, and importance of testing.
- Testers do not understand the work being performed by other engineers including both technology and application domain.
- Testers are unaware of changes to the requirements, architecture, design, and implementation.
- Testing is less effective and less efficient.
- The system/software engineering process and testing process are incompatible.
- The project master schedule and the testing schedule are incompatible.
- Testing work products are not delivered in time for major project reviews and decision points.
- Engineering work products are not sufficiently testable.
- Testing risks are not incorporated into the project risk repository.
- The requirements engineering, architecture engineering, design, and implementation tools are not sufficiently compatible with the testing tools.
- Etc.

Example – Testing and Engineering Processes Not Integrated (GEN-PRO-7)

Potential Causes

- There was inadequate communication between testers and other system or software engineers (for example,).
  - Requirements engineers, architects, designers, and implementers do not communicate changes to the requirements, architecture, design, and implementation to the testers.
- Testers were not involved in:
  - Initial overall planning
  - Determining and documenting the overall engineering process
- Testers did not have sufficient expertise or experience in the relevant technology and the application domain.
- System and software engineers did not respect the testers, feeling that they were not qualified to take part in project.
- System and software engineers were not involved with the verification of the testing work products.
- The people determining and documenting the overall engineering process did not have significant testing expertise, training, or experience.
- The testing schedule was not integrated into the overall project schedule.
- Testing was outsourced.
- Test tool selection drove the testing process rather than the other way around.

Example – Testing and Engineering Processes Not Integrated (GEN-PRO-7)

Recommendations

- **Prepare:**
  - Include testers in the initial staffing of the project.
- **Enable:**
  - Provide a top-level briefing or training in testing to the chief system engineer, system architect, and process engineer.
- **Perform:**
  - Subject-matter experts and project testers collaborate closely with the project chief engineer or technical lead and process engineer when they develop the engineering process descriptions and associated process documents.
  - Provide high-level overviews of testing in the SEMP(s) and SDP(s).
  - Document how testing is integrated into the system development or life cycle, regardless of whether it is traditional waterfall, evolutionary (iterative, incremental, and parallel), or anything in between.
  - For example, document handover points in the development cycle when testing input and output work products are delivered from one project organization or group to another.
  - Incorporate testing into the Project Master Schedule.
  - Incorporate testing into the project's Work Breakdown Structure (WBS).
- **Verify:**
  - Determine whether testers were involved in planning the project's system or software development process.
  - Determine whether testing is incorporated into the project's System engineering process, System development cycle, System Engineering Master Plan and System Development Plan, Work Breakdown Structure, Master Schedule

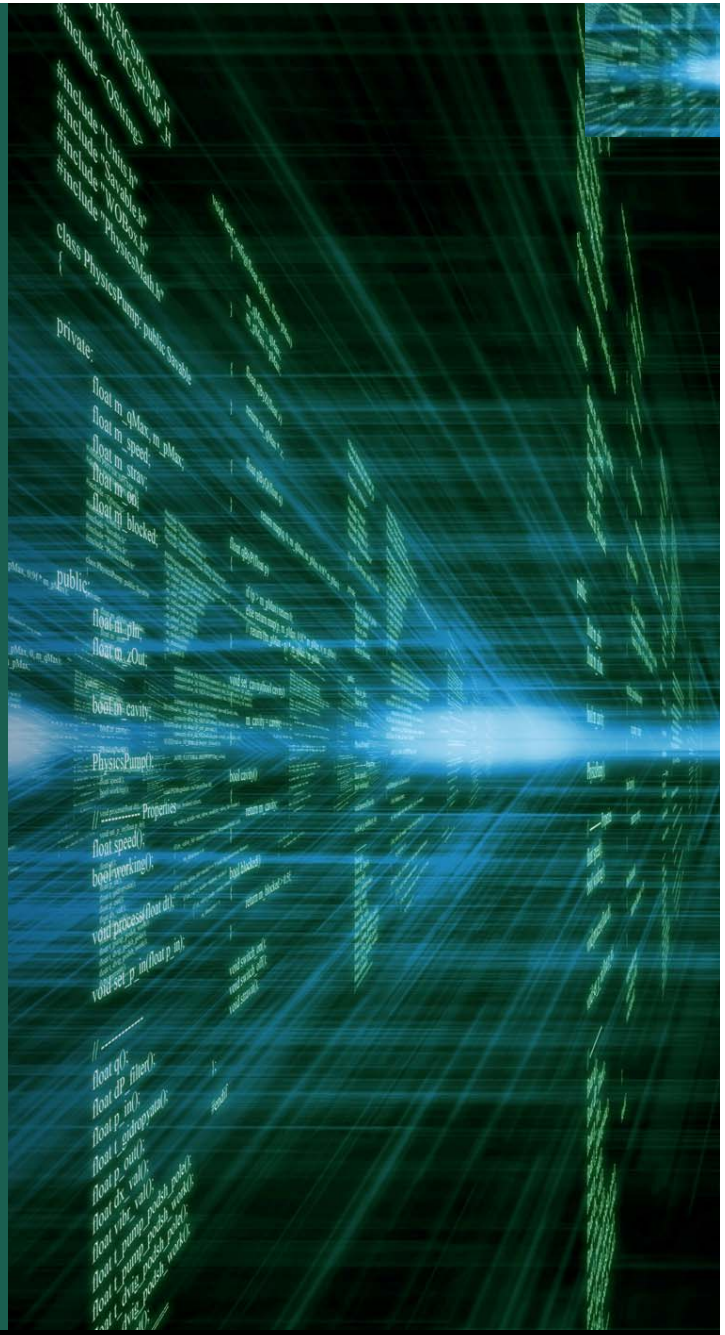
Example – Testing and Engineering Processes Not Integrated (GEN-PRO-7)

Related Pitfalls

- Testing at the End (GEN-TPS-6) If the testing and engineering processes are not properly integrated, then testing is more likely to be delayed to the end of development.
- Independent Test Schedule (GEN-TPS-7) If the testing and engineering processes are not integrated, then the test schedule will be independent of and therefore probably incompatible with the overall project master schedule.
- Testers Responsible for All Testing (GEN-STF-4) If the testing and engineering processes are not properly integrated, then the developers are more likely to believe that the testers are responsible for all testing.
- Adversarial Relationship (GEN-STF-9) If the testing and engineering processes are not integrated, then the developers and the testers are more likely to develop and adversarial rather than cooperative relationship.
- Testing as a Phase (GEN-PRO-18) If the testing and engineering processes are not properly integrated, then testing is more likely to be viewed as a phase that is separate from the rest of development.
- Testers Not Involved Early (GEN-PRO-19) If the testing and engineering processes are not properly integrated, then testers are less likely to be involved early in the development process (such as during initial planning, requirements engineering, and architecture engineering.
- Testing in Quality (GEN-PRO-23) If the testing and engineering processes are not properly integrated, then the developers will be more likely to believe that quality is the testers responsibility and that quality can be testing into the system or software.
- Developers Ignore Testability (GEN-PRO-24) If the testing and engineering processes are not properly integrated, then the developers are more likely to ignore testability.
- Inadequate Communication Concerning Testing (GEN-COM-5) If the testing and engineering processes are not properly integrated, then it is more likely that there will be inadequate communication between the testers and the rest of the engineering staff.

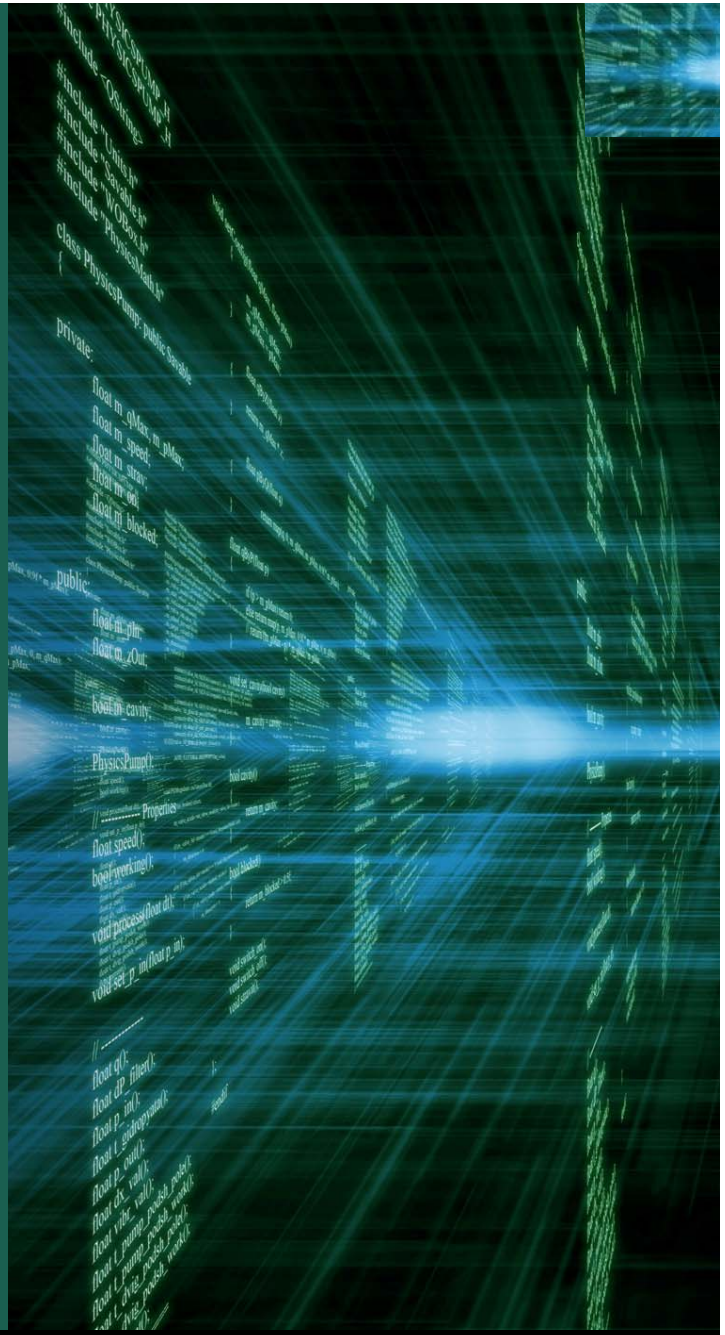
A Taxonomy of System & Software Testing Types

The Taxonomy



A Taxonomy of System & Software Testing Types

Taxonomy – General Pitfalls



Categories of Testing Pitfalls – General

1 Test Planning and Scheduling
2 Stakeholder Involvement and Commitment
3 Management
4 Staffing
5 Testing Process
6 Test Design
7 Pitfall-Related [new]
8 Test Tools [new - split]
9 Test Environments [new - split]
10 Automated Testing [new]
11 Test Communication
12 Testing-as-a-Service (TaaS) [new]
13 Requirements
14 Test Data [new]

General Pitfalls –
Test Planning and Scheduling

No Separate Test Planning Documentation (GEN-TPS-1)

Incomplete Test Planning (GEN-TPS-2)

Test Plans Ignored (GEN-TPS-3)

Test-Case Documents as Test Plans (GEN-TPS-4)

Inadequate Test Schedule (GEN-TPS-5)

Testing at the End (GEN-TPS-6)

Independent Test Schedule (GEN-TPS-7) [new]

General Pitfalls –
Stakeholder Involvement and Commitment

Wrong Testing Mindset (GEN-SIC-1)

Unrealistic Testing Expectations (GEN-SIC-2)

Assuming Testing Only Verification Method Needed (GEN-SIC-3)

Mistaking Demonstration for Testing (GEN-SIC-4)

Lack of Stakeholder Commitment to Testing (GEN-SIC-5)

General Pitfalls –
Management

Inadequate Test Resources (GEN-MGMT-1)

Inappropriate External Pressures (GEN-MGMT-2)

Inadequate Test-Related Risk Management (GEN-MGMT-3)

Inadequate Test Metrics (GEN-MGMT-4)

Inconvenient Test Results Ignored (GEN-MGMT-5)

Test Lessons Learned Ignored (GEN-MGMT-6)

Inadequate Test-Related Configuration Management (GEN-MGMT-7) [new]

General Pitfalls –
Staffing - 1

Lack of Independence (GEN-STF-1)

Unclear Testing Responsibilities (GEN-STF-2)

Developers Responsible for All Testing (GEN-STF-3)

Testers Responsible for All Testing (GEN-STF-4)

Testers Responsible for Ensuring Quality (GEN-STF-5) [new]

Testers Fix Defects (GEN-STF-6) [new]

Users Responsible for Testing (GEN-STF-7) [new]

Inadequate Testing Expertise (GEN-STF-8)

Inadequate Domain Expertise (GEN-STF-9) [new]

Adversarial Relationship (GEN-STF-10) [new]

General Pitfalls –
Staffing - 2

Too Few Testers (GEN-STF-11) [new]

Allowing Developers to Close Discrepancy Reports (GEN-STF-12) [new]

Testing Death March (GEN-STF-13) [new]

All Testers Assumed Equal (GEN-STF-14) [new]

General Pitfalls –
Testing Process - 1

No Planned Testing Process (GEN-PRO-1) [new]

Essentially No Testing (GEN-PRO-2) [new]

Inadequate Testing (GEN-PRO-3)

Testing Process Ignored (GEN-PRO-4) [new]

One-Size-Fits-All Testing (GEN-PRO-5)

Testing and Engineering Processes Not Integrated (GEN-PRO-6)

Too Immature for Testing (GEN-PRO-7)

Inadequate Evaluations of Test Assets (GEN-PRO-8)

Inadequate Maintenance of Test Assets (GEN-PRO-9)

Test Assets Not Delivered (GEN-PRO-10)

Testing as a Phase (GEN-PRO-11)

General Pitfalls –
Testing Process - 2

Testers Not Involved Early (GEN-PRO-12)

Developmental Testing During Production (GEN-PRO-13) [new]

No Operational Testing (GEN-PRO-14)

Testing in Quality (GEN-PRO-15) [new]

Developers Ignore Testability (GEN-PRO-16) [moved from other category]

Failure to Address the Testing BackBlob (GEN-PRO-17) [new]

Failure to Analyze Why Defects Escaped Detection (GEN-PRO-18) [new]

Official Test Standards are Ignored (GEN-PRO-20) [new]

Official Test Standards are Slavishly Followed (GEN-PRO-21) [new]

Developing New When Old Fails Tests (GEN-PRO-22) [new]

General Pitfalls –
Test Design [new]

Sunny Day Testing Only (GEN-TD-1) [new]

Inadequate Test Prioritization (GEN-TD-2)

Test-Type Confusion (GEN-TD-3)

Functionality Testing Overemphasized (GEN-TD-4)

System Testing Overemphasized (GEN-TD-5)

System Testing Underemphasized (GEN-TD-6)

Test Preconditions Ignored (GEN-TD-7) [new]

Test Oracles Ignore Nondeterministic Behavior (GEN-TD-8) [new]

No Test Design (GEN-TD-9) [new]

Test Design Ignores Test Architecture (GEN-TD-10) [new]

No Test Case Selection Criteria (GEN-TD-11) [new]

No Test Case Completion Criteria (GEN-TD-12) [new]

Inadequate Test Oracle (GEN-TD-13) [new]

General Pitfalls –
Pitfall-Related [new]

Overly Ambitious Process Improvement (GEN-PRP-1) [new]

Inadequate Pitfall Prioritization (GEN-PRP-2) [new]

General Pitfalls –
Test Tools [new]

Over-Reliance on Test Tools (GEN-TT-1)

Cost Only / Primary Test Tool Selection Criteria (GEN-TT-2) [new]

Test Tools Drive Testing Process (GEN-TT-3) [new]

Unusable Test Tool (GEN-TT-4) [new]

Incompatible Test Tools (GEN-TT-5) [new]

Inadequate Test Tool Training (GEN-TT-6) [new]

General Pitfalls –
Test Environments

Incomplete Test Environment (GEN-TE-1) [new]

Poor Preparation for Numerous Platforms (GEN-TE-2) [new]

Insufficient Number of Test Environments (GEN-TE-3)

Missing Type of Test Environment (GEN-TE-4) [new]

Poor Fidelity of Test Environments (GEN-TE-5)

Inadequate Test Environment Quality (GEN-TE-6)

Test Environments Inadequately Tested (GEN-TE-7) [new]

Inadequate Testing in a Staging Environment (GEN-TE-8) [new]

Insecure Test Environment (GEN-TE-9) [new]

Improperly Configured Test Environment (GEN-TE-10) [new]

General Pitfalls –
Automated Testing - 1 [new]

Automated Testing not Treated as a Project (GEN-AUTO-1) [new]

Insufficient Automated Testing (GEN-AUTO-2) [new]

Automated Testing Replaces Manual Testing (GEN-AUTO-3) [new]

Automated Testing Replaces Testers (GEN-AUTO-4) [new]

Inappropriate Distribution of Automated Tests (GEN-AUTO-5) [new]

Inadequate Automated Test Quality (GEN-AUTO-6) [new]

Excessively Complex Automated Tests (GEN-AUTO-7) [new]

Automated Tests Not Maintained (GEN-AUTO-8) [new]

Insufficient Resources Invested (GEN-AUTO-9) [new]

Inappropriate Automation Tools (GEN-AUTO-10) [new]

General Pitfalls –
Automated Testing - 2 [new]

Unclear Responsibilities for Automated Testing (GEN-AUTO-11) [new]

Postponing Automated Testing Until Stable (GEN-AUTO-12) [new]

Automated Testing as Silver Bullet (GEN-AUTO-13) [new]

Incompatible Automation Tools (GEN-AUTO-14) [new]

Testers Make Good Test Automation Engineers (GEN-AUTO-15) [new]

General Pitfalls –
Test Communication

Inadequate Architecture or Design Documentation (GEN-COM-1)

Inadequate Discrepancy Reports (GEN-COM-2)

Inadequate Test Documentation (GEN-COM-3)

Source Documents Not Maintained (GEN-COM-4)

Inadequate Communication Concerning Testing (GEN-COM-5)

Inconsistent Testing Terminology (GEN-COM-6) [new]

Redundant Test Documents (GEN-COM-7) [new]

General Pitfalls –
Testing-as-a-Service (TaaS) [new]

Cost-Driven Provider Selection (GEN-TaaS-1) [new]

Inadequate Oversight (GEN-TaaS-2) [new]

Inadequate Collaboration (GEN-TaaS-3) [new]

Lack of Outsourcing Expertise (GEN-TaaS-4) [new]

Inappropriate TaaS Contract (GEN-TaaS-5) [new]

TaaS Provider Improperly Chosen (GEN-TaaS-6) [new]

No or Late Delivery of Testing Assets (GEN-TaaS-7) [new]

TaaS Vendor Obtains “Vendor Lock” (GEN-Taas-8) [new]

General Pitfalls – Requirements

Tests as Requirements (GEN-REQ-1) [new]

Ambiguous Requirements (GEN-REQ-2)

Obsolete Requirements (GEN-REQ-3)

Missing Requirements (GEN-REQ-4)

Incomplete Requirements (GEN-REQ-5)

Incorrect Requirements (GEN-REQ-6)

Requirements Churn (GEN-REQ-7)

Improperly Derived Requirements (GEN-REQ-8)

Verification Methods Not Adequately Specified (GEN-REQ-9)

Lack of Requirements Trace (GEN-REQ-10)

Titanic Effect of Deferred Requirements (GEN-REQ-11) [new]

Implicit Requirements Ignored (GEN-REQ-12) [new]

General Pitfalls –
Test Data [new]

Inadequate Test Data (GEN-DATA-1) [moved from other category]

Poor Fidelity of Test Data (GEN-DATA-2) [new]

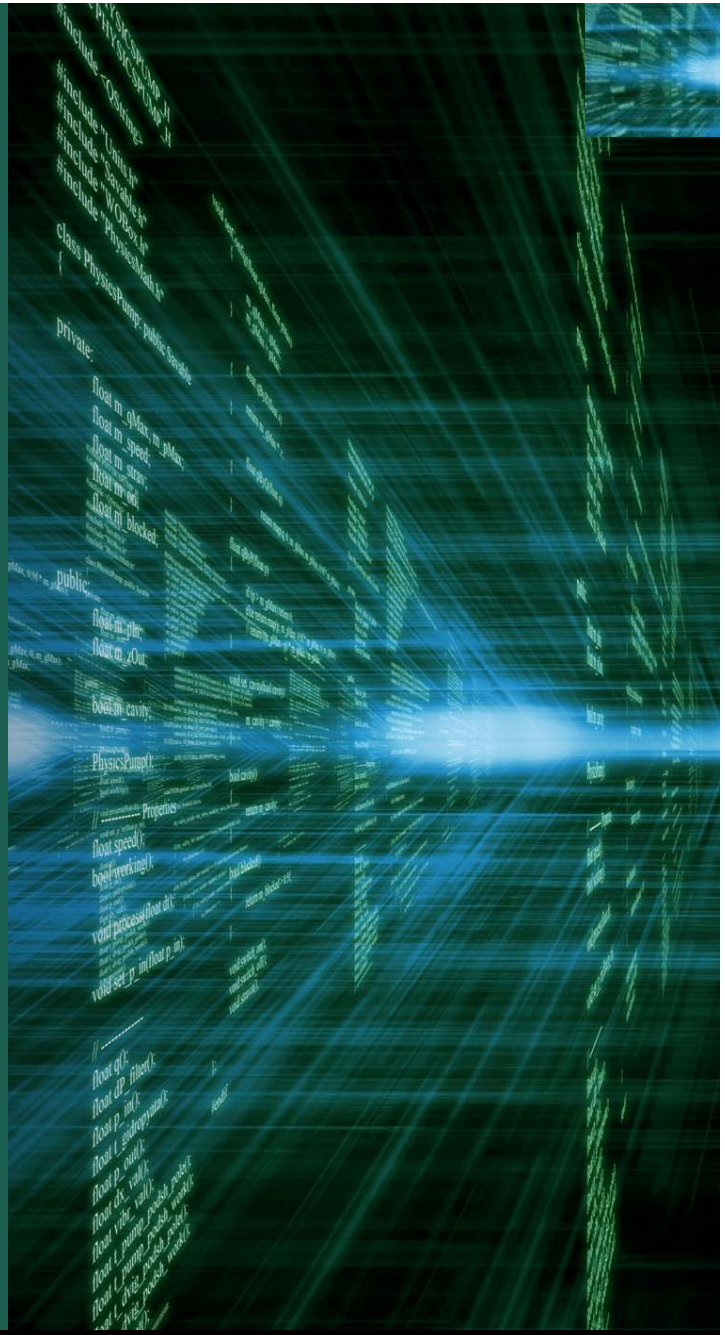
Production Data in Test Data (GEN-DATA-3) [new]

Test Data in Production Data (GEN-DATA-4) [new]

Excessive Test Data (GEN-DATA-5) [new]

A Taxonomy of System & Software Testing Types

Taxonomy – Test-Type-Specific Pitfalls



Categories of Testing Pitfalls – Test-Type-Specific Pitfalls

1 Executable Model Testing [new]

2 Unit Testing

3 Integration Testing

4 Specialty Engineering Testing

5 System Testing

6 User Testing [new]

7 A/B Testing [new]

8 Acceptance Testing [new]

9 Operational Testing [new]

10 System of Systems (SoS) Testing

11 Regression Testing

Test Type Specific Pitfalls –
Executable Model Testing [new]

Inadequate Executable Models (TTS-MOD-1) [new]

Executable Models Not Tested (TTS-MOD-2) [new]

Test Type Specific Pitfalls –
Unit Testing

Testing Does Not Drive Design and Implementation (TTS-UNT-1)

Conflict of Interest (TTS-UNT-2)

Untestable Units (TTS-UNT-3) [new]

Brittle Test Cases (TTS-UNT-4) [new]

No Unit Testing (TTS-UNT-5) [new]

Unit Testing of Automatically Generated Units (TTS-UNT-6) [new]

Test Type Specific Pitfalls – Integration Testing

Integration Decreases Testability Ignored (TTS-INT-1)

Inadequate Self-Testing (TTP-INT-2)

Unavailable Components (TTS-INT-3)

System Testing as Integration Testing (TTS-INT-4)

Test Type Specific Pitfalls –
Specialty Engineering Testing

Inadequate Capacity Testing (TTS-SPC-1)

Inadequate Concurrency Testing (TTS-SPC-2)

Inadequate Configurability Testing (TTS-SPC-3) [new]

Inadequate Interface Standards Compliance Testing (TTS-SPC-4) [new]

Inadequate Internationalization Testing (TTS-SPC-5)

Inadequate Interoperability Testing (TTS-SPC-6)

Inadequate Performance Testing (TTS-SPC-7)

Inadequate Portability Testing (TTS-SPC-8)

Inadequate Reliability Testing (TTS-SPC-9)

Inadequate Robustness Testing (TTS-SPC-10)

Inadequate Safety Testing (TTS-SPC-11)

Inadequate Security Testing (TTS-SPC-12)

Inadequate Usability Testing (TTS-SPC-13)

Test Type Specific Pitfalls – System Testing

Test Hooks Remain (TTS-SYS-1)

Lack of Test Hooks (TTS-SYS-2)

Inadequate End-to-End Testing (TTS-SYS-3)

Test Type Specific Pitfalls –
User Testing [new]

Inadequate User Involvement (TTS-UT-1) [new]

Unprepared User Representatives (TTS-UT-2) [new]

User Testing Merely Repeats System Testing (TTS-UT-3) [new]

User Testing is Mistaken for Acceptance Testing (TTS-UT-4) [new]

Assuming Knowledgeable and Careful Users (TTS-UT-5) [new]

User Testing Too Late to Fix Defects (TTS-UT-6) [new]

Test Type Specific Pitfalls –
A/B Testing [new]

Poor Key Performance Indicators (TTS-ABT-1) [new]

Misuse of Probability and Statistics (TTS-ABT-2) [new]

Confusing Statistical Significance with Business Significance (TTS-ABT-3) [new]

Error Source(s) Not Controlled (TTS-ABT-4) [new]

System Variant(s) Changed During Test (TTS-ABT-5) [new]

Test Type Specific Pitfalls –
Acceptance Testing [new]

No Clear System Acceptance Criteria (TTS-AT-1) [new]

Acceptance Testing Only Tests Functionality (TTS-AT-2) [new]

Developers Determine Acceptance Tests (TTS-AT-3) [new]

Test Type Specific Pitfalls – Operational Testing (OT) [new]

No On-Site SW Developers (TTS-OT-1) [new]

Inadequate Operational Testing (TTS-OT-2) [new]

Test Type Specific Pitfalls –
System of System (SoS) Testing

Inadequate SoS Test Planning (TTS-SoS-1)

Unclear SoS Testing Responsibilities (TTS-SoS-2)

Inadequate Resources for SoS Testing (TTS-SoS-3)

SoS Testing not Properly Scheduled (TTS-SoS-4)

Inadequate SoS Requirements (TTS-SoS-5)

Inadequate Support from Individual System Projects (TTS-SoS-6)

Inadequate Defect Tracking Across Projects (TTS-SoS-7)

Finger-Pointing (TTS-SoS-8)

Test Type Specific Pitfalls –
Regression Testing

Inadequate Automation of Regression Testing (TTS-REG-1)

Regression Testing Not Performed (TTS-REG-2)

Inadequate Scope of Regression Testing (TTS-REG-3)

Only Low-Level Regression Testing (TTS-REG-4)

Test Assets Not Delivered (TTS-REG-5)

Only Functional Regression Testing (TTS-REG-6)

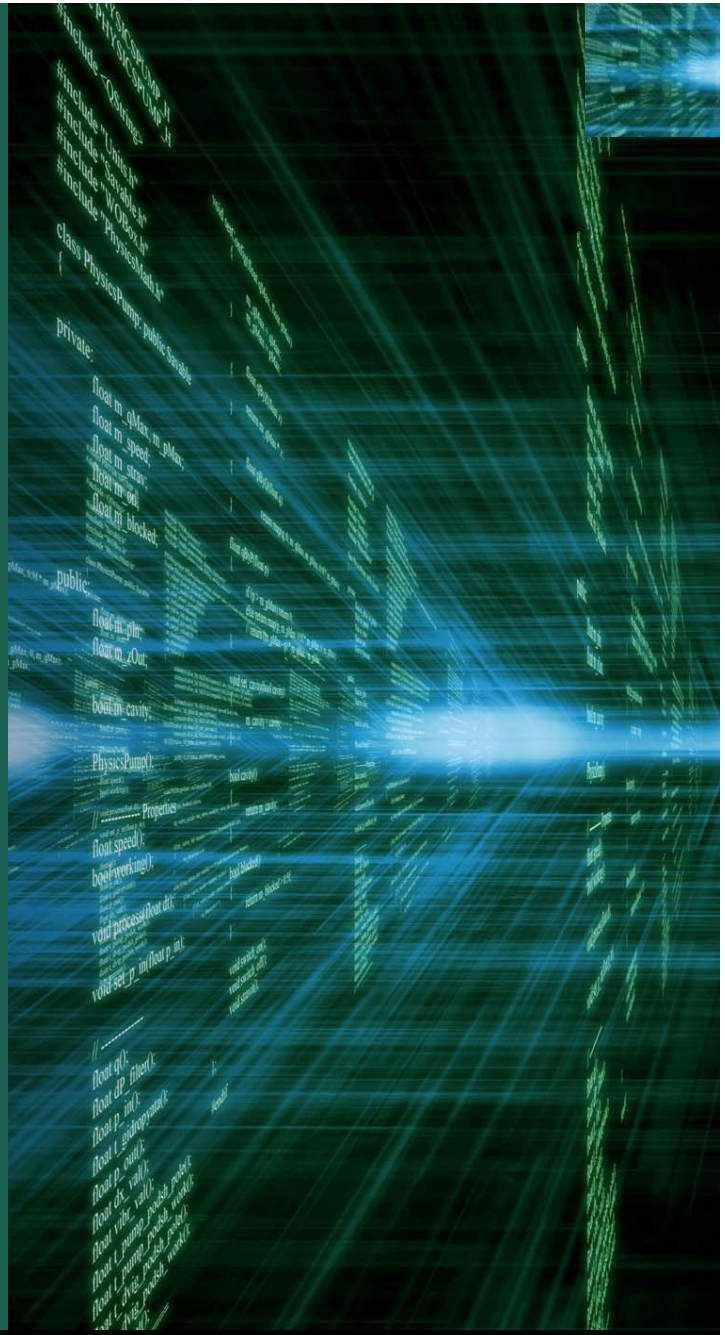
Inadequate Retesting of Reused Software (TTS-REG-7) [new]

Only Automated Regression Testing (TTS-REG-8) [new]

Regression Tests Not Maintained (TTS-REG-9) [new]

A Taxonomy of System & Software Testing Types

Conclusion



Remaining Limitation and Questions

Current Taxonomy is Experience Based:

- Based on experience testing and assessing testing programs (author, SEI ITAs, technical reviewers)
- Not the result of documentation study or formal academic research

Remaining Questions:

- Which pitfalls occur most often? With what frequency? Top 10 most common?
- Which pitfalls cause the most harm? Top 10 wrt. harm?
- Which pitfalls have the highest risk (expected harm = harm frequency x harm)?
- What factors influence frequency, harm, and risk?
  - System/software (size, complexity, criticality, application domain, software only vs. HW/SW/people/documentation/facilities/procedures…, system vs. SoS vs. PL, development/test processes)?
  - Project (type, formality, lifecycle scope, schedule, funding, commercial vs. government/military,…)
  - Organization (number, size, type, governance, management/engineering culture…)

Future Work

SEI Wiki (currently under development)

Extensive Technical Review:

- New testing pitfall categories
- New and modified testing pitfalls

Proper Industry Survey:

- Design of Experiment (DoE)
- Creation of survey questions
- Size needed for statistical validity
- Analysis of test results
- Reporting of test results

Simple Expert System:

- Inputs: answers to survey questions
- Outputs: Estimated highest risk pitfalls

Contact Information Slide Format

Donald G. Firesmith

Principal Engineer

Software Solutions Division

Telephone: +1 412-268-6874

Email: dgf@sei.cmu.edu

U.S. Mail

Software Engineering Institute

Customer Relations

4500 Fifth Avenue

Pittsburgh, PA 15213-2612

USA

Web

www.sei.cmu.edu

www.sei.cmu.edu/contact.cfm

Customer Relations

Email: info@sei.cmu.edu

Telephone: +1 412-268-5800

SEI Phone: +1 412-268-5800

SEI Fax: +1 412-268-6257