# Common System and Software Testing Pitfalls

**TSP-2014 Conference**
**Pittsburgh, Pennsylvania**

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Donald G. Firesmith, Principle Engineer
3 November 2014

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.

# Topics

Testing:

- What is Testing?
- Software vs. System Testing
- Why Test?
- Limitations of Testing?

Testing Pitfalls:

- Taxonomy / Ontology
- Testing Challenges
- Addressing these Challenges
- Goals and Potential Uses
- Example Pitfall

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

- General Pitfalls
- Test-Type-Specific Pitfalls

Remaining Limitation and Questions

Future Work

# What is Testing?

Testing

> The execution of an Object Under Test (OUT) under specific preconditions (for example, pretest mode, states, stored data, or external conditions) with specific inputs so that its actual behavior (outputs and postconditions) can be compared with its expected or required behavior

Notes:

- The OUT can be:
  - An executable requirements, architecture, or design model
  - A system or subsystem
  - A software application, component, or unit
- Not just software
- Requires execution (do not confuse testing with QE or with other means of verification such as analysis, certification, demonstration, inspection, …)
- Requires ***controllability*** to set up preconditions and provide inputs
- Requires ***observability*** to determine outputs and postconditions

# *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.

# 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:

- Different levels of testing have different defect removal efficiencies (DREs)
- Different types of testing uncover different types of defects

May provide false positive and false negative results

Cannot prove the OUT works properly under all inputs and conditions

# Testing Pitfalls

Testing Pitfall

- A *human mistake* that unnecessarily and unexpectedly causes testing to be:
  - Less effective at uncovering defects
  - Less efficient in terms of time and effort expended
  - More frustrating 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
- Having sufficient frequency (and consequences ) to be a significant risk

# Taxonomy and Ontology

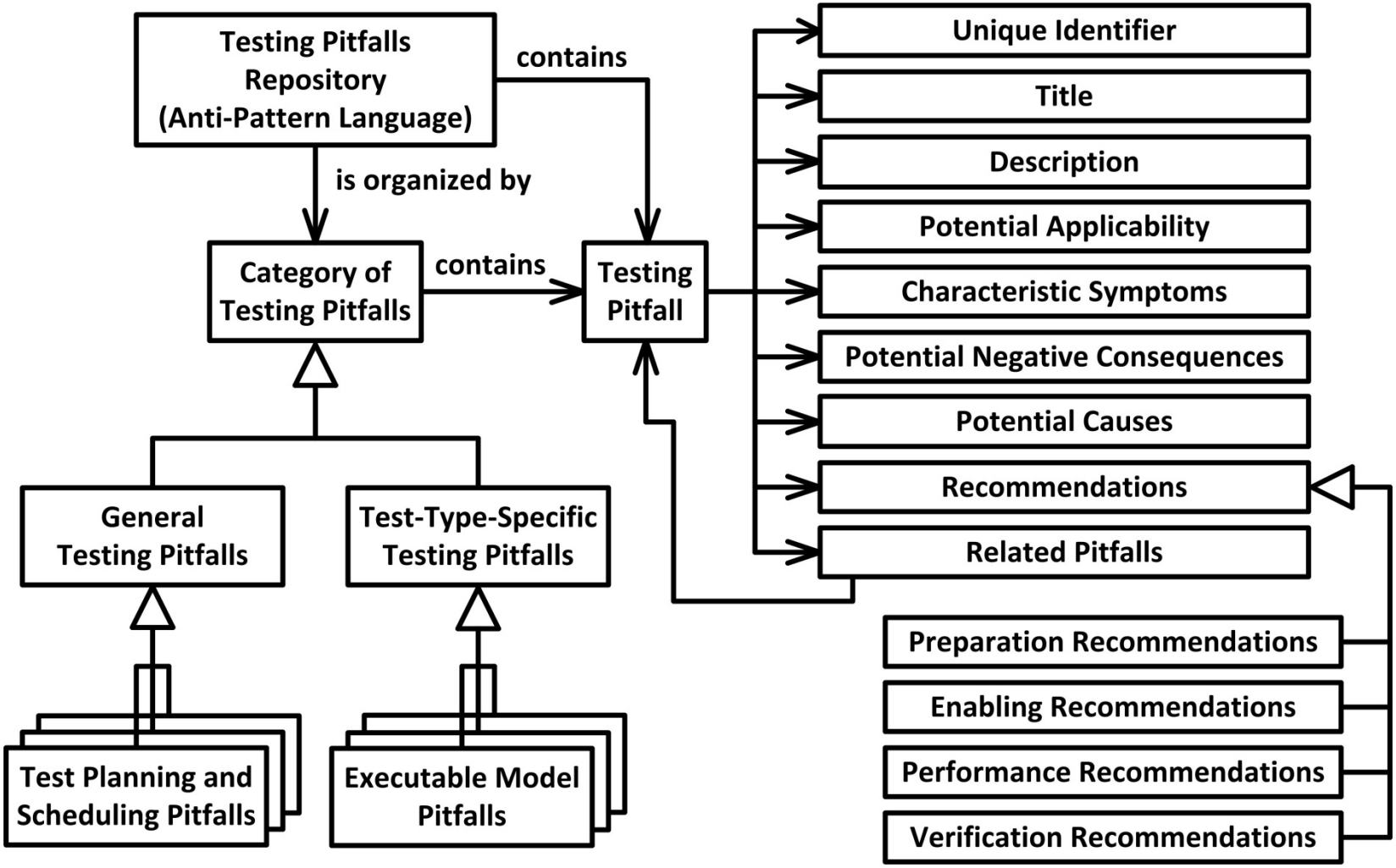
Testing Pitfall ***Taxonomy***

A hierarchical classification of testing pitfalls into categories

Testing Pitfall ***Ontology***

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

# Testing Pitfall Taxonomy and Ontology



# Testing 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.

# 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 testing international SMEs

Current taxonomy/repository with new pitfalls and pitfall categories:

- ***145*** pitfalls classified into ***21*** categories
- http://sites.google.com/a/firesmith.net/donald-firesmith/home/common-testing-pitfalls [Work in progress - new content is draft and may be incomplete]

Potential 2nd Edition of Pitfalls Book or Supplement to 1st Edition

Potential Testing Pitfalls Wiki

# Goals and Potential Uses

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

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

**Description** The testing process is not adequately integrated into the overall system engineering process.

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

**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 staff members 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.

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

## Potential Negative Consequences


- There is inadequate communication between testers and other system or software engineers (for example, requirements engineers, architects, designers, and implementers).
- Few nontesters understand the scope, complexity, and importance of testing.
- Testers do not understand the work being performed by other engineers.
- Testers can produce test cases and automated testing scripts before the requirements, architecture, and design has stabilized, thereby forcing the testers to modify their test cases and test scripts as the system or software changes and incorrect hidden assumptions are uncovered.
- Testing is less effective and takes longer than necessary.

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

## Potential Causes

- Testers were not involved in determining and documenting the overall engineering process.
- The people determining and documenting the overall engineering process did not have significant testing expertise, training, or experience.
- The testing schedule has not been integrated into the overall project schedule.
- Testing was outsourced.

# 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.

# Categories of Testing Pitfalls – General

1 Test Planning and Scheduling

2 Stakeholder Involvement and Commitment

3 Management

4 Staffing

5 Testing Process

6 Pitfall-Related [new pitfall category since book]

7 Test Tools and Environments

8 Automated Testing [new pitfall category since book]

9 Test Communication

10 Testing-as-a-Service (TaaS) [new pitfall category since book]

11 Requirements

# 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 pitfall]

# General Pitfalls –
## Stakeholder Involvement and Commitment

Wrong Testing Mindset (GEN-SIC-1)

**→ Unrealistic Testing Expectations (GEN-SIC-2)**

Lack of Stakeholder Commitment to Testing (GEN-SIC-3)

# 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)

# General Pitfalls –
## Staffing

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 pitfall]

Users Responsible for Testing (GEN-STF-6) [new pitfall]

**→ Inadequate Testing Expertise (GEN-STF-7)**

Inadequate Domain Expertise (GEN-STF-8) [new pitfall]

Adversarial Relationship (GEN-STF-9) [new pitfall]

# General Pitfalls –
## Testing Process 1

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

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

Incomplete Testing (GEN-PRO-3)

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

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

Sunny Day Testing Only (GEN-PRO-6) [new pitfall]

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

Inadequate Test Prioritization (GEN-PRO-8)

Test-Type Confusion (GEN-PRO-9)

Functionality Testing Overemphasized (GEN-PRO-10)

# General Pitfalls –
## Testing Process 2

Black-Box System Testing Overemphasized (GEN-PRO-11)

Black-Box System Testing Underemphasized (GEN-PRO-12)

Test Preconditions Ignored (GEN-PRO-13) [new pitfall]

Too Immature for Testing (GEN-PRO-14)

Inadequate Test Data (GEN-PRO-15)

Inadequate Evaluations of Test Assets (GEN-PRO-16)

**→ Inadequate Maintenance of Test Assets (GEN-PRO-17)**

Testing as a Phase (GEN-PRO-18)

Testers Not Involved Early (GEN-PRO-19)

Developmental Testing During Production (GEN-PRO-20) [new pitfall]

# General Pitfalls –
## Testing Process 3

No Operational Testing (GEN-PRO-21)

Test Oracles Ignore Nondeterministic Behavior (GEN-PRO-22) [new pitfall]

**→ Testing in Quality (GEN-PRO-23) [new pitfall]**

Developers Ignore Testability (GEN-PRO-24) [moved from other category]

Testing the BackBlob (GEN-PRO-25) [new pitfall]

Test Assets Not Delivered (GEN-PRO-26) [combined 2 existing pitfalls]

# General Pitfalls –
## Pitfall-Related [new pitfall category]

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

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

# General Pitfalls –
## Test Tools and Environments

Over-Reliance on Testing Tools (GEN-TTE-1)

Target Platform Difficult to Access (GEN-TTE-2)

**→ Inadequate Test Environments (GEN-TTE-3)**

Poor Fidelity of Test Environments (GEN-TTE-4)

Inadequate Test Environment Quality (GEN-TTE-5)

Test Environments Inadequately Tested (GEN-TTE-6) [new pitfall]

Inadequate Test Configuration Management (GEN-TTE-7)

# General Pitfalls –
## Automated Testing [new pitfall category]

Over-Reliance on Manual Testing (GEN-AUTO-1)
[moved from Test Tools and Environments Category]

Automated Testing Replaces Manual Testing (GEN-AUTO-2) [new pitfall]

Excessive Number of Automated Tests (GEN-AUTO-3) [new pitfall]

Inappropriate Distribution of Automated Tests (GEN-AUTO-4) [new pitfall]

Inadequate Automated Test Quality (GEN-AUTO-5) [new pitfall]

Automated Tests Excessively Complex (GEN-AUTO-6) [new pitfall]

**→ Automated Tests Not Maintained (GEN-AUTO-7) [new pitfall]**

Insufficient Resources Invested (GEN-AUTO-8) [new pitfall]

Automation Tools Not Appropriate (GEN-AUTO-9) [new pitfall]

Stakeholders Ignored (GEN-AUTO-10) [new pitfall]

# General Pitfalls –
## Test Communication

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

**→ Inadequate Defect 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 pitfall]

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

**→ Cost-Driven Provider selection (GEN-TaaS-1) [new pitfall]**

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

Lack of Outsourcing Expertise (GEN-TaaS-3) [new pitfall]

Inappropriate TaaS Contract (GEN-TaaS-4) [new pitfall]

TaaS Provider Improperly Chosen (GEN-TaaS-5) [new pitfall]

# General Pitfalls –
## Requirements

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

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 Properly Specified (GEN-REQ-9)

Lack of Requirements Trace (GEN-REQ-10)

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

# Categories of Testing Pitfalls – Test-Type-Specific Pitfalls

1 Executable Model Testing [new pitfall category]

2 Unit Testing

3 Integration Testing

4 Specialty Engineering Testing

5 System Testing

6 User Testing [new pitfall category]

7 A/B Testing [new pitfall category]

8 Acceptance Testing [new pitfall category]

8 System of Systems (SoS) Testing

9 Regression Testing

# Test Type Specific Pitfalls –
## Executable Model Testing [new pitfall category]

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

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

# Test Type Specific Pitfalls –
## Unit Testing

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

**→ Conflict of Interest (TTS-UNT-2)**

Brittle Test Cases (TTS-UNT-3) [new pitfall]

No Unit Testing (TTS-UNT-4) [new pitfall]

# 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 pitfall]**

Inadequate Internationalization Testing (TTS-SPC-4)

Inadequate Interface Standards Compliance Testing (TTS-SPC-5) [new pitfall]

Inadequate Interoperability Testing (TTS-SPC-6)

Inadequate Performance Testing (TTS-SPC-7)

Inadequate Reliability Testing (TTS-SPC-8)

Inadequate Robustness Testing (TTS-SPC-9)

Inadequate Safety Testing (TTS-SPC-10)

Inadequate Security Testing (TTS-SPC-11)

Inadequate Usability Testing (TTS-SPC-12)

# 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 pitfall category]

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

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

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

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

**→ Assume Knowledgeable and Careful User (TTS-UT-5) [new pitfall]**

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

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

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

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

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

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

# Test Type Specific Pitfalls –
## Acceptance Testing [new pitfall category]

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

# 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)

# General Pitfalls –
## Regression Testing

Inadequate Regression Test Automation (GEN-REG-1)

Regression Testing Not Performed (GEN-REG-2)

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

Only Low-Level Regression Tests (GEN-REG-4)

Only Functional Regression Testing (GEN-REG-5)

Inadequate Retesting of Reused Software (TTS-REG-6) [new pitfall]

# 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?
- Which pitfalls cause the most harm?
- Which pitfalls have the highest risk (expected harm = harm frequency x harm)?
- What factors (e.g., system/software size and complexity, application domain, process) influence frequency, harm, and risk?

# Future Work

Second Edition of Book

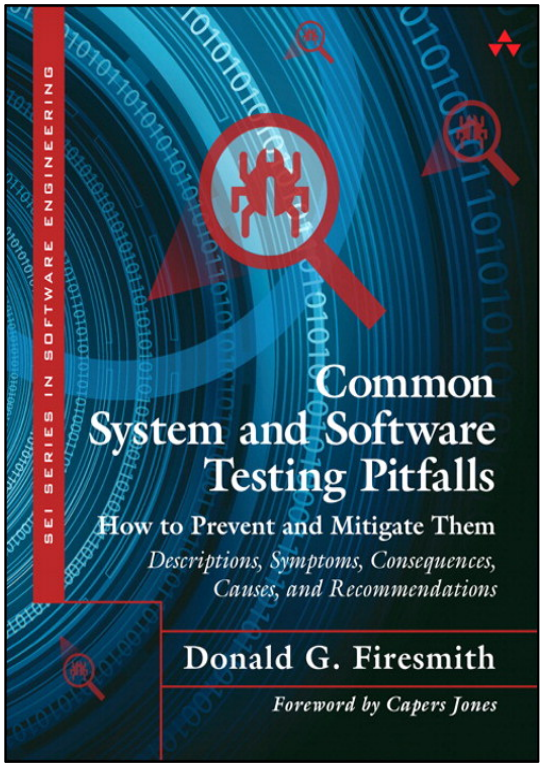
Extensive Technical Review:

- New testing pitfall categories
- New and modified testing pitfalls

Proper Industry Survey:

- How likely are the different testing pitfalls? What are the 10 most common?
- What pitfalls have the worst consequences? What are the 10 worst pitfalls?
- What pitfalls have the highest risk? What are the 10 highest risk pitfalls?
- Do the answers to these questions vary by:
  - System (size, complexity, criticality, application domain, software only vs. HW/SW/people/documentation/facilities/procedures…, system vs. SoS vs. PL)?
  - Project (type, formality, lifecycle scope, schedule, funding, commercial vs. government/military,…)
  - Organization (number, size, type, governance, management/engineering culture,…)

# Save 35%* at informit.com




Discount code:

**FIRESMITH550**

- informit.com - search on Firesmith
- Available as book & eBook
- FREE shipping in the U.S.

*Offer expires Dec 31, 2014

http://sites.google.com/a/firesmith.net/donald-firesmith/home/common-testing-pitfalls

Addison Wesley

PEARSON

# 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