



# Common System and Software Testing Pitfalls

**MITRE System Architecture Technical Exchange Meeting (SA TEM)**  
**8 April 2014**  
**(updated based on presentation)**

**Donald Firesmith**, Principle Engineer  
Software Solutions Division  
Software Engineering Institute (SEI)  
Carnegie Mellon University  
Pittsburgh, PA 15213



# Topics

Challenges

Addressing these Challenges

Goal and Potential Uses

Individual Testing Pitfalls

Individual Pitfall Documentation

Example Pitfall

Categories of Testing Pitfalls

Individual Testing Pitfalls:

- General Testing Pitfalls
- Test Type Specific Testing Pitfalls

Limitations

Future Work



# 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

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

Documents testing pitfalls organized into categories:

- Commonly occurring
- Both system and software testing pitfalls

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

- 92 pitfalls organized into 14 categories
- Technically reviewed by 47 testing SMEs from around the world

New Pitfalls and Pitfall Categories

(<http://donald.firesmith.net/home/common-testing-pitfalls>):

- 127 pitfalls organized into 18 categories



# Goal and Potential Uses

## Goal:

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

## Potential Uses:

- Training materials for testers and testing stakeholders
- Standard terminology regarding commonly occurring testing pitfalls
- Checklists for use when:
  - Producing test plans and related documentations
  - Evaluating contractor proposals
  - Evaluating test 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 metrics collection



# Individual Testing Pitfalls – Documentation

Name and Unique ID (based on type of testing pitfall)

Description

Potential Applicability

Characteristic Symptoms

Potential Negative Consequences

Potential Causes

Recommendations (Prepare, Enable, Perform, Verify)

Related Pitfalls



# Example – Wrong Testing Mindset (GEN-SIC-1)

**Description:** Some of the testers and other testing stakeholders have the wrong testing mindset.

**Potential Applicability:**

- This pitfall is always potentially applicable.



# Example – Wrong Testing Mindset (GEN-SIC-1)

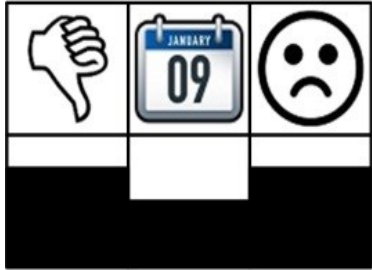
## Characteristic Symptoms:

- Some testers and other testing stakeholders believe that the purpose of testing is to demonstrate that the system works properly rather than to determine where and how it fails.[13]
- Testers believe that it is their job (responsibility) to verify or “prove” that the system works, rather than identify where and when it doesn’t work.[14]
- Some testers and other testing stakeholders begin testing assuming that the system or software works, so testing is only performed to show this.
- Managers and acquirers believe that testing is a cost center (that is, an expense rather than an investment) because they do not see the value of the products produced (for example, test documentation, test cases, and test environments) or the costs avoided due to testing.
- Testers believe that the purpose of testing is to find out how the system actually behaves, without considering how it should or must behave. In this mindset, testing is unrelated to requirements, and defects are subjective and all in the “eye of the beholder.”
- Only normal (primary and alternative) behavior is tested.
- There is little or no testing of:
  - Exceptional (error-, fault-, or failure-tolerant) behavior
  - Input data:
    - There is no testing to identify incorrect handling of invalid input values.
    - Test inputs include only middle-of-the-road values rather than boundary values and corner cases.





# Example – Wrong Testing Mindset (GEN-SIC-1)



## Potential Consequences:

- There is a high probability that:
  - the delivered system or software will contain a significant number of residual defects, especially related to abnormal behavior (e.g., exceptional use case paths)
  - these defects will unacceptably reduce its reliability and robustness (e.g., error, fault, and failure tolerance)
- Customer representatives, managers, and developers have a false sense of security that the system functions properly.



# Example – Wrong Testing Mindset (GEN-SIC-1)

## Potential Causes:

- Customer representatives, managers, and developers believe that testing can be exhaustive.
- Those performing testing were told (either implicitly or explicitly) that their job is to verify or “prove” that the system works.[15]
- Testers were simply unaware of the proper mindset.
- Developers tested their own software,[16] so that there was a “conflict of interest” (that is, they were to build software that works and then use testing to show that it does not work). This is especially a problem with small, cross-functional development organizations or teams that “cannot afford” to have separate testers (that is, professional testers who specialize in and have specialized expertise in testing).
- There was insufficient schedule allocated for testing, so there was only sufficient time to test the normal behavior (for example, normal, “sunny-day” use case paths).
- The organizational culture is very success oriented, and management gave the testers the strong impression that:
  - Management does not want to hear any “bad” news (that is, that any significant defects were found in the system).
  - They discourage testers from working “too hard” to uncover defects.
- Testers who report lots of defects are not considered “team players” and are held responsible for any resulting cost or schedule overruns.



# Example – Wrong Testing Mindset (GEN-SIC-1)

## Recommendations:

- **Prepare:**

- Testers should clearly articulate the goals of testing to project management and the developers.
- Explicitly state in the project test plan that the primary goals of testing is to:
  - break the system by causing system faults and failures in order to identify residual defects so that they can be fixed
  - thereby determine the
  - quality of the system
  - system’s fitness for purpose
  - system’s readiness for shipping, deployment, and/or operation

- **Enable:**

- Provide test training that emphasizes uncovering defects by causing faults or failures.
- Provide sufficient time in the schedule for testing beyond the basic success paths.
- Hire new testers who exhibit a strong “destructive” mindset to testing.
- When relevant, identify this pitfall as a risk in the project risk repository.

- **Perform:**

- In addition to test cases that verify all normal behavior, emphasize looking for defects where they are most likely to hide (e.g., boundary values, corner cases, and input type/range verification). [16]
- Incentivize testers based more on the number of significant defects they uncover than merely on the number requirements “verified” or test cases ran. [17]
- Foster a healthy competition between developers (who seek to avoid inserting defects) and testers (who seek to find those defects).



# Example – Wrong Testing Mindset (GEN-SIC-1)

- **Verify:**
  - Determine whether the testers exhibit a testing mindset.
  - Determine whether the goals of testing are documented in the test planning documentation.
  - Determine whether any test training covers the proper testing mindset.
  - Determine whether adequate time has been scheduled to enable testing beyond the basic success paths.
  - Determine (for example, via conversation or questioning) whether testing goes beyond “demonstrate that the system works” (sunny-day path testing) to also include “demonstrate that the system does not work” (rainy-day path testing)
  - Determine whether the testers exhibit the correct testing mindset.

## Related Pitfalls:

- Inappropriate External Pressures (GEN-MGMT-2)
- Inadequate Communication Concerning Testing (GEN-COM-5)
- Unit Testing Considered Unimportant (TTS-UNT-3)



# Categories of Testing Pitfalls – General

Test Planning and Scheduling Pitfalls

Stakeholder Involvement and Commitment Pitfalls

Management-Related Testing Pitfalls

Staffing Pitfalls

Test Process Pitfalls

Test Tools and Environments Pitfalls

Automated Testing Pitfalls [new pitfall category since book]

Regression Testing Pitfalls

Test Communication Pitfalls

Requirements-Related Testing Pitfalls



# Categories of Testing Pitfalls – Test-Type Specific

Executable Model Testing Pitfalls [\[new pitfall category\]](#)

Unit Testing Pitfalls

Integration Testing Pitfalls

Specialty Engineering Testing Pitfalls

System Testing Pitfalls

User Testing Pitfalls [\[new pitfall category\]](#)

Acceptance Testing Pitfalls [\[new pitfall category\]](#)

System of Systems (SoS) Testing Pitfalls



# 1 General Pitfalls –

## 1.1 Test Planning and Scheduling Pitfalls

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]



# 1. General Pitfalls –

## 1.2 Stakeholder Involvement and Commitment Pitfalls

Wrong Testing Mindset (GEN-SIC-1)

Unrealistic Testing Expectations (GEN-SIC-2)

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





# General Pitfalls – Management-related Testing Pitfalls

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 Pitfalls

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)

Only Testers Held Responsible for Quality (GEN-STF-5) [new pitfall]

Inadequate Testing Expertise (GEN-STF-6)

Inadequate Domain Expertise (GEN-STF-7) [new pitfall]

Adversarial Relationship (GEN-STF-8) [new pitfall]



# General Pitfalls – Test Process Pitfalls 1

No 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)

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

Inadequate Test Prioritization (GEN-PRO-7)

Test-Type Confusion (GEN-PRO-8)

Functionality Testing Overemphasized (GEN-PRO-9)



# General Pitfalls – Test Process Pitfalls 2

Black-Box System Testing Overemphasized (GEN-PRO-10)

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

Test Preconditions Ignored (GEN-PRO-12) [new pitfall]

Too Immature for Testing (GEN-PRO-13)

Inadequate Test Data (GEN-PRO-14)

Inadequate Evaluations of Test Assets (GEN-PRO-15)

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

Testing as a Phase (GEN-PRO-17)

Testers Not Involved Early (GEN-PRO-18)

Developmental Testing During Production (GEN-PRO-10) [new pitfall]



# General Pitfalls – Test Process Pitfalls 3

No Operational Testing (GEN-PRO-21)

Testing Ignores Nondeterministic Behavior (GEN-PRO-22) [new pitfall]

Ad Hoc Testing (GEN-PRO-22) [new pitfall]

Testing in Quality (GEN-PRO-24) [new pitfall]



# General Pitfalls – Test Tools and Environments Pitfalls

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

Too Many Target Platforms (GEN-TTE-2)

Target Platform Difficult to Access (GEN-TTE-3)

Inadequate Test Environments (GEN-TTE-4)

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

Inadequate Test Environment Quality (GEN-TTE-6)

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

Inadequate Test Configuration Management (GEN-TTE-8)

Developers Ignore Testability (GEN-TTE-9)

Test Assets Not Delivered (GEN-TTE-10) [combined 2 existing pitfalls]



# General Pitfalls – Automated Testing Pitfalls [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 – Regression Testing Pitfalls

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)

Test Resources Not Delivered for Maintenance (GEN-REG-5)

Only Functional Regression Testing (GEN-REG-6)

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





# General Pitfalls – Test Communication Pitfalls

Inadequate Source Documentation (GEN-COM-1) [Expanded in Scope and Renamed]

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 – Requirements Testing Pitfalls

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]



# Test-Type-Specific Pitfalls

Executable Model Testing Pitfalls [\[new pitfall category\]](#)

Unit Testing Pitfalls

Integration Testing Pitfalls

Specialty Engineering Testing Pitfalls

System Testing Pitfalls

User Testing Pitfalls [\[new pitfall category\]](#)

Acceptance Testing Pitfalls [\[new pitfall category\]](#)

System of Systems (SoS) Testing Pitfalls



# Test Type Specific Pitfalls – Executable Model Testing Pitfalls

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

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



# Test Type Specific Pitfalls – Unit Testing Pitfalls

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

Conflict of Interest (TTS-UNT-2)



# Test Type Specific Pitfalls – Integration Testing Pitfalls

Integration Decreases Testability (TTS-INT-1)

Inadequate Self-Monitoring (TTP-INT-2)

Unavailable Components (TTS-INT-3)

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



# Test Type Specific Pitfalls – Specialty Engineering Testing Pitfalls

Inadequate Capacity Testing (TTS-SPC-1)

Inadequate Concurrency Testing (TTS-SPC-2)

Inadequate Internationalization Testing (TTS-SPC-3)

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

Inadequate Interoperability Testing (TTS-SPC-5)

Inadequate Performance Testing (TTS-SPC-6)

Inadequate Reliability Testing (TTS-SPC-7)

Inadequate Robustness Testing (TTS-SPC-8)

Inadequate Safety Testing (TTS-SPC-9)

Inadequate Security Testing (TTS-SPC-10)

Inadequate Usability Testing (TTS-SPC-11)



# Test Type Specific Pitfalls – System Testing Pitfalls

Lack of Test Hooks (TTS-SYS-1)

Inadequate Testing Hooks (TTS-SYS-2)

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





# Test Type Specific Pitfalls – User Testing

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 – Acceptance Testing Pitfalls

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



# Test Type Specific Pitfalls – System of System Testing Pitfalls

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)

Poor or Missing 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)



# Limitations

Current taxonomy is Experience Based:

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

Pitfalls are not Prioritized (by frequency, consequence, or risk)

- Still need to perform industry survey



# Future Work

## Extensive Technical Review:

- New Testing Pitfall Categories
- New testing pitfalls
- Modified testing pitfalls

## 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,...)



# Questions?



# Contact Information Slide Format

## Donald Firesmith

Principle Engineer

Software Support Division (SSD)

Telephone: +1 412-268-6874

Email: [dgf@sei.cmu.edu](mailto:dgf@sei.cmu.edu)

## Web

[www.sei.cmu.edu](http://www.sei.cmu.edu)

[www.sei.cmu.edu/contact.cfm](http://www.sei.cmu.edu/contact.cfm)

## U.S. Mail

Software Engineering Institute

Customer Relations

4500 Fifth Avenue

Pittsburgh, PA 15213-2612

USA

## Customer Relations

Email: [info@sei.cmu.edu](mailto:info@sei.cmu.edu)

Telephone: +1 412-268-5800

SEI Phone: +1 412-268-5800

SEI Fax: +1 412-268-6257



## NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this presentation is not intended in any way to infringe on the rights of the trademark holder.

This Presentation may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at [permission@sei.cmu.edu](mailto:permission@sei.cmu.edu).

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

