Using Failure Modes to Generate Test Ideas

Dawn Haynes

Senior Trainer & Consultant

Portland SPIN Seminar – December 13, 2007

Introduction

- Who am I?
- Who are you?
- What inspired this talk?
  - The observation that we seem to write and release the same bugs over and over
  - National Science Foundation research that led to the How to Break Software series of books
  - The notion that we can learn from our mistakes

Agenda

Our Roots

- Typical failures in software
- Failure causes
- Typical test generation sources

Suggestions for Improvement

- Using failures as a guide

What are some typical failures?

- Incorrect data displayed
- Data corruption or truncation
- Slow response or hang
- Display errors
- Memory leak
- Crash or unexpected shutdown
- Incorrect error messages

What are some failure causes?

Locations of Errors & Root Causes	
Specifications	Typos, mistakes, logic errors
Diagrams and flowcharts	Ambiguities, gaps
Code	Assumptions
User documentation	Misunderstandings
Test cases	Belief in the autonomy of system components

Where do tests typically come from?

- Project specifications
- Functional specifications
- Workflow and process flow diagrams
- System specifications
- Use cases
- Technical specifications
- Design documents and diagrams
- Exploration of real software behavior

How can we use failures as a guide?

- Error guessing
- Generic fault models
- Historical failures
- Failure analysis methods

Error Guessing

- Using experience as a guide
- Remembering frequent or spectacular failures and going on a hunt
- Every software tester has a bucket of these
  - Zero, null, blank, space, -1
  - Special characters
  - … and more

Generic Fault Models

- Technology
- Process maturity
- Software behavior

Technology Areas

- Programming languages
  - Java
  - C / C++
- Architectures
  - SOA
  - Web
  - Client/server
- Technical implementations
  - Security

Research Common Technology Flaws

- Web
  - How to Break Web Software *(Whittaker & Andrews)*
- Security
  - How to Break Software Security *(Whittaker & Thompson)*
  - Common vulnerabilities database
    - http://cve.mitre.org
  - CERT
    - http://www.us-cert.gov/cas/alldocs.hmtl

Process Maturity

- Software development lifecycle & methods
  - Agile, waterfall, iterative, V-model, etc.
  - Test driven development, unit testing
  - Coding standards, code reviews
- Formal processes and standards
  - Six Sigma
  - CMMI
  - ISO, IEEE

Research Internal Processes

- Seek out standards and processes
- Understand what they will catch or prevent
- Identify what they are not good at
- Look to reduce redundancy and duplication of effort
  - Employ the “trust but verify” technique
  - Reduce system test emphasis on covered areas
- Develop tests to cover the gaps

Software Behavior

- Capabilities
  - Input
  - Output
  - Data handling (storage & retrieval)
  - Computation
- Interfaces
  - GUIs, APIs, devices
  - Operating system & file system
  - Other supporting software

Ref: How to Break Software, James Whittaker

Use Common Software Attacks

- How to Break Software
  - 17 user interface attacks
    - Long strings
    - Repeating tasks
    - Use default values
    - Interacting inputs or features
  - 6 file system interface attacks
  - Generic methods for attacking the operating system and supporting software

Ref: How to Break Software, James Whittaker

Historical Failures

- Internal sources
  - Defects from formal testing groups
  - Developer submitted issues
  - Issues reported from other staff
  - Anomalies and intermittent failures
- External sources
  - Customer submissions (defects, requests)
  - Issues reported by beta testers
  - Reports from consultants or field engineers

Customer Defects – Typical Approach

- Get bug report from customer
- Attempt to reproduce
  - If successful, give test outline to development
- Create a regression test for the bug
- Run the regression test on any patch release or new version to ensure the bug does not escape again

Results of this Approach

- Fix one bug
- Create one test
- Run same test over and over
- Eventually the bug is gone and we have a TEST in hand instead of a TEST IDEA

Analyze Failures

- Defects or other incorrect behavior
  - Perform root-cause analysis if possible
  - Group similar items (by guess or by cause)
  - Look for patterns
- Try to create your own generic tests based on these patterns
  - Seek to create test ideas instead of single tests
  - Make lists (or search for lists/heuristics/guides)

Failure Analysis Methods

- Industry research
- Informal
- Statistical (failures and failure rates)
- Failure modes and effects analysis (FMEA)
- Cause-effect graphing

Failure Modes and Effects Analysis

- FMEA is intended to document:
  - a Failure
  - its Mode
  - its Effect
  - by Analysis
- in a cause-effect manner.

Ref: Michael Schlueter, Six Sigma Web site

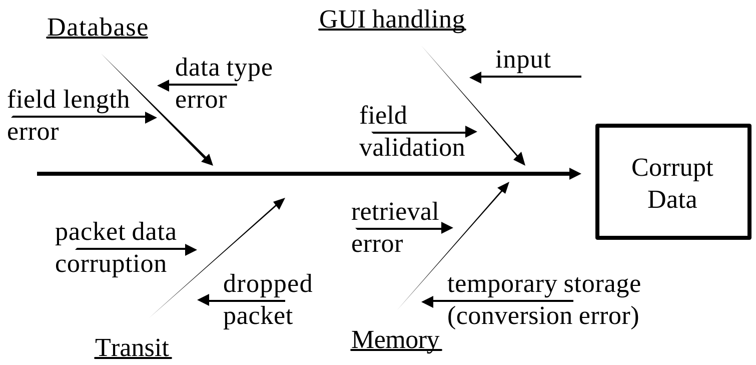
Software FMEA Example

- Failure
  - Sluggish response to client/UI
- Modes
  - Low memory condition
  - Slow network connection
  - Overloaded server
- Effects
  - User repeats operation
  - User closes application during processing

Ref: Michael Schlueter, Six Sigma Web site

Cause-Effect Graphing Tool

- Ishikawa or fishbone diagrams



Database

field length error

data type error

GUI handling

input

field validation

Corrupt Data

packet data corruption

dropped packet

Transit

retrieval error

temporary storage (conversion error)

Memory

Testing Industry Credits

- Common software errors
  - James Whittaker, Elisabeth Hendrickson, Hugh Thompson, James Bach, Cem Kaner
- Other sources
  - SQE, IEEE, ISO, ASQ
- Other references - disclaimer
  - Any missing references to the original author or source is unintentional if pointed out will be promptly correct
  - Some errors and omissions may be due to a lack of proper citing on some Web sites, books, articles and other materials I have used

Contact Information

- Dawn Haynes
- dhaynes@perftestplus.com

Good Luck

&

Happy Testing!