Software Disasters and the Importance of Proper Testing

EFCOG 2021

Software Disasters and the Importance of Proper Testing

The cost of software problems or errors is a significant problem to global industry, not only to the producers of the software but also to their customers and end users of the software.

Software Disasters and the Importance of Proper Testing

The Art of Software Testing, Glenford J. Myers

- A necessary part of a test case is a definition of the expected output or result.
- A programmer should avoid attempting to test his or her own program.
- A programming organization should not test its own programs.
- Thoroughly inspect the results of each test.
- Test cases must be written for invalid and unexpected, as well as valid and expected, input conditions.
  - Examining a program to see if it does what it is supposed to do is only half the battle. The other half is seeing whether the program does what it is not supposed to do.
  - Avoid throw-away test cases unless the program is truly a throw-away program.
  - Do not plan testing effort under the assumption that no error will be found.
- The probability of the existence of more errors in a section of a program is proportional to the number of errors already found in that section.
  - Testing is an extremely creative and intellectually challenging task.
  - Testing is the process of executing a program with the intent of finding errors.
- A good test case is one that has a high probability of detecting an as-yet undiscovered error.
- A successful test case is one that detects an as-yet undiscovered error.

Glenford Myers-Computer Scientist
Glenford Myers is an American computer scientist, entrepreneur, and author. He founded two successful high-tech companies, authored eight textbooks in the computer sciences, and made important contributions in microprocessor architecture. He holds a number of patents, including the original patent on "register scoreboarding" in microprocessor chips. He has a BS in electrical engineering from Clarkson University, an MS in computer science from Syracuse University, and a PhD in computer science from the Polytechnic Institute of New York University

Software Disasters and the Importance of Proper Testing

The Complete Guide to Software Testing, Bill Hetzel

Key Testing Principles:

- Complete testing is not possible.
- Testing is creative and difficult.
- An important reason for testing is to prevent errors.
- Testing is risk-based.
- Test must be planned.
- Testing requires independence.

National Security always matters, obviously. But the reality is that if you have an open door in your software for the good guys, the bad guys get in there too.
Tim Cook

Dr. **William C. Hetzel** is an expert in the field of software testing. He compiled the papers from the 1972 Computer Program Test Methods Symposium, also known as the Chapel Hill Symposium, into the book *Program Test Methods.* The book, published in 1973, details the problems of software validation and testing.
In 1988 Gelperin and Heztel write the article The Growth of Software Testing. In it they discuss four major models for software testing. The first two are Phase Models, and the second two are Life Cycle Models.[
Gelperin and Hetzel developed the STEP methodology for implementing the original IEEE-829-1998 Standard for Software and System Test Documentation. Their firm was instrumental in gaining recognition for testing as a separate discipline within the software industry.

Software Disasters and the Importance of Proper Testing

Effective Software Testing: 50 Specific Ways to Improve Your Testing, Elfriede Dustin

Developer testing is an important step towards accountability. It gives developers a way to demonstrate the quality of the software they produce.
Kent Beck

1. Involve Testers from the Beginning
2. Verify the Requirements
3. Design Test Procedures As Soon As Requirements Are Available
4. Ensure That Requirement Changes Are Communicated
5. Beware of Developing and Testing Based on an Existing System
6. Understand the Task At Hand and the Related Testing Goals
7. Consider the Risks
8. Base Testing Efforts on a Prioritized Feature Schedule
9. Keep Software Issues in Mind
10. Acquire Effective Test Data
11. Plan the Test Environment
12. Estimate Test Preparation and Execution Time

Elfriede Dustin is author of the book “Effective Software Testing” and lead author of “Automated Software Testing” and “Quality Web Systems.”

Software Disasters and the Importance of Proper Testing

Effective Software Testing: 50 Specific Ways to Improve Your Testing, Elfriede Dustin

13. Define Roles and Responsibilities
14. Require a Mixture of Testing Skills, Subject-Matter Expertise, and Experience
15. Evaluate the Tester's Effectiveness
16. Understand the Architecture and Underlying Components
17. Verify That the System Supports Testability
18. Use Logging to Increase System Testability
19. Verify That the System Supports Debug and Release Execution Modes
20. Divide and Conquer
21. Mandate the Use of a Test-Procedure Template and Other Test-Design Standards
22. Derive Effective Test Cases from Requirements
23. Treat Test Procedures as "Living" Documents
24. Utilitize System Design and Prototype
25. Use Proven Testing Techniques when Designing Test-Case Scenarios.

Software Disasters and the Importance of Proper Testing

Effective Software Testing: 50 Specific Ways to Improve Your Testing, Elfriede Dustin

26. Avoid Including Constraints and Detailed Data Elements within Test Procedures
27. Apply Exploratory Testing
28. Structure the Development Approach to Support Effective Unit Testing
29. Develop Unit Tests in Parallel or Before the Implementation
30. Make Unit-Test Execution Part of the Build Process
31. Know the Different Types of Testing Support Tools
32. Consider Building a Tool Instead of Buying One
33. Know the Impact of Automated Tools on the Testing Effort
34. Focus on the Needs of Your Organization
35. Test the Tools on the Application Prototype
36. Do Not Rely Solely on Capture/Playback
37. Develop a Test Harness When Necessary

Software Disasters and the Importance of Proper Testing

Effective Software Testing: 50 Specific Ways to Improve Your Testing, Elfriede Dustin

38. Use Proven Test-Script Development Techniques
39. Automate Regression tests When Feasible
40. Implement Automated Builds and Smoke Tests
41. Do Not Make Nonfunctional Testing an Afterthought
42. Conduct Performance Testing with Production-Sized Databases
43. Tailor Usability Tests to the Intended Audience
44. Consider All Aspects of Security, for Specific Requirements and System-Wide
45. Investigate the System's Implementation To Plan for Concurrency Tests
46. Set Up an Efficient Environment for Compatibility Testing
47. Clearly Define the Beginning and End of the Test-Execution Cycle
48. Isolate the Test Environment from the Development Environment
49. Implement a Defect-tracking Life Cycle
50. Track the Execution of the Testing Program

Software Disasters and the Importance of Proper Testing

Software Disaster's due to Lack of Testing

Software Disasters

- .

Great Software has seemingly limitless potential to solve human problems – and it can spread around the world in a blink of an eye. Malicious code moves just as quickly, and when software created for the wrong reason, it has a huge and growing capacity to harm millions of people.

Craig Federighi

Software Disasters

Software Configuration Error Caused Plane Crash

- An executive of Airbus Group has confirmed that the crash of an Airbus A400M military transport was caused by a faulty software configuration. Marwan Lahoud, chief marketing and strategy officer for Airbus, told the German newspaper *Handelsblatt* on Friday that there was a "quality issue in the final assembly" of the components of the aircraft engine.

- **https://en.wikipedia.org/wiki/2015_Seville_Airbus_A400M_crash**
- The problem appeared to be a compatibility issue between the recorders and the DGA's data reading system, rather than an issue with the condition of the recorders themselves.
- Airbus Chief Strategy Officer Marwan Lahoud confirmed on 29 May that incorrectly installed engine control software caused the fatal crash. "The black boxes attest to that there are no structural defects [with the aircraft], but we have a serious quality problem in the final assembly.

Software Disasters

Nasa’s Metric Confusion Caused Mars Orbiter Loss

- WASHINGTON (AP) Mars Climate Orbiter was speeding through space and speaking to NASA in metric for nine months. In fact, the engineers on the ground were replying in non-metric English.
- The Mars Climate Orbiter was a 638-kilogram robotic space probe launched by NASA on December 11, 1998 to study the Martian climate, Martian atmosphere, and surface changes and to act as the communications relay in the Mars Surveyor '98 program for Mars Polar Lander.
- However, on September 23, 1999, communication with the spacecraft was permanently lost as it went into orbital insertion.
- The spacecraft encountered Mars on a trajectory that brought it too close to the planet, and it was either destroyed in the atmosphere or escaped the planet's vicinity and entered an orbit around the Sun.
- An investigation attributed the failure to a measurement mismatch between two software systems: metric units by NASA and US Customary units by spacecraft builder Lockheed Martin.

Software Disasters

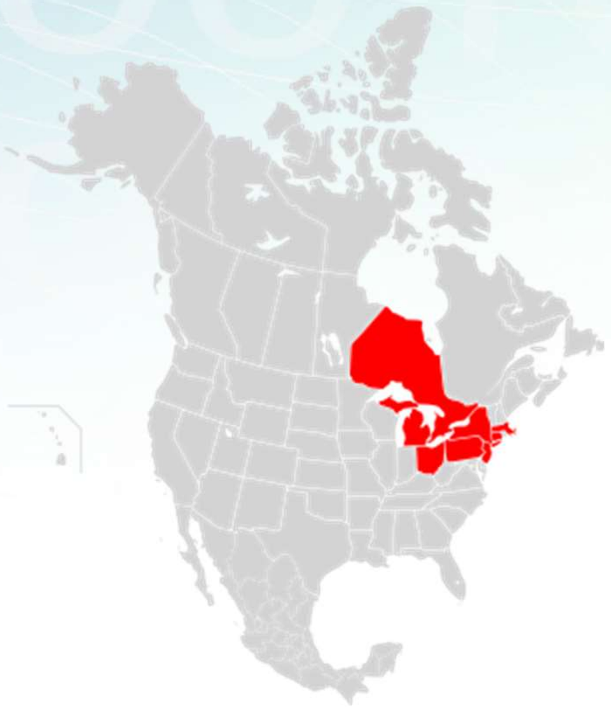
Recalled 990,000 Vehicles for Air Bag Malfunction

- According to a bulletin from the National Highway Traffic Safety Administration, the recall affects 989,701 vehicles registered in the U.S.
- The recall stems from a flaw in Nissan's occupant classification system software -- the software that determines whether the front passenger seat is occupied. When that software detects a passenger, it activates the airbags around the passenger seat. It believes the seat is empty, and deactivates those airbags.
- Unfortunately, the software installed on the vehicles listed above may incorrectly determine that the passenger seat is empty when it is, in fact, occupied. If that were to happen, and if the vehicle were subsequently involved in an accident, the passenger-seat airbags would fail to deploy, increasing the possibility of injury or death.

Software Disasters

Northeast Blackout

- The **Northeast blackout of 2003** was a widespread power outage through Northeastern and Midwestern United States, and the Canadian province of Ontario
- At the time, it was the world's second most widespread blackout in history.
- The outage, affected an estimated 10 million people Canada, and 45 million people U.S.
- The blackout's proximate cause was a software bug in the alarm system at the control room of FirstEnergy, an Akron, Ohio–based company,
- The “bug”, rendered operators unaware of the need to redistribute load after overloaded transmission lines drooped into foliage. What should have been a manageable local blackout cascaded into the collapse of much of the Northeast regional electricity distribution system.



Software Disasters

Northeast Blackout

- **Findings**
  - FirstEnergy (FE) did not operate its system with appropriate voltage criteria."
  - FirstEnergy "did not recognize or understand the deteriorating condition of its system."
  - FirstEnergy "failed to manage adequately tree growth in its transmission rights-of-way."
  - Finally, the "failure of the interconnected grid's to provide effective real-time diagnostic support."

- A generating plant in Ohio, went offline amid high electrical demand, putting a strain on high-voltage power lines ,which later went out of service when they came in contact with "overgrown trees".
- This trip caused load to transfer to other transmission lines, which were not able to bear the load, tripping their breakers. Once these multiple trips occurred, many generators suddenly lost parts of their loads, so they accelerated out of phase with the grid at different rates, and tripped out to prevent damage. The cascading effect that resulted ultimately forced the shutdown of at least 265 power plants.

- **Computer failure**
- A software bug known as a race condition existed in General Electric Energy's Unix-based XA/21 energy management system. Once triggered, the bug stalled FirstEnergy's control room alarm system for over an hour. System operators were unaware of the malfunction. The failure deprived them of both audio and visual alerts for important changes in system state.

- Unprocessed events queued up after the alarm system failure and the primary server failed within 30 minutes.
- Then all applications (including the stalled alarm system) were automatically transferred to the backup server, which itself failed.
- The server failures slowed the screen refresh rate of the operators' computer consoles from 1–3 seconds to 59 seconds per screen.
- The lack of alarms led operators to dismiss a call from American Electric Power about the tripping and reclosure of a 345 kV shared line in northeast Ohio.
- But, after the control room itself lost power, control room operators informed technical support (who were already troubleshooting the issue) of the alarm system problem.

Software Disasters

Overexposure of radiation therapy patients in National Cancer Institute, Panama City

It is one of the 6 Famous Software Disasters.

- It was a Therapy planning software caught in a series of accidents,
- Created by Multi Data Systems International, a U.S. firm, the software (*could have been a spreadsheet (CDR)),*….not able to calculate properly the exact dosage of radiation for patients undergoing radiation therapy.

- In March 2001, serious overreactions in patients undergoing radiation therapy for cancer treatment.
  - Of the 478 patients treated for pelvic cancers, 3 of them had died, possibly from an overdose of radiation.
  - treatment times calculated using a computerized treatment planning system
  - Twenty-three of the 28 overexposed patients had died by September 2005, with at least 18 of the deaths being from radiation effects.

- The Latin American Association for Radiation Oncology established an accreditation commission.
- Accreditation will require that centers implement a comprehensive radiation oncology quality assurance program that follows international guidelines.

Software Disasters

Frenchman Sues Uber Over A Software Bug

- In what could've been a thrilling episode of To Catch a Cheater, a French businessman has filed a whopping $45 million lawsuit against Uber after a bug in the ride-sharing app allegedly allowed his wife to keep tabs on his trips, ultimately leading her to suspect he was being unfaithful.
- According to French newspaper Le Figaro, the man claimed the glitch caused the app to send notifications as well as other transportation details to his wife's handset.
- While it remains unclear what caused the malfunction, the man purportedly used his wife's device to log in to Uber, but the phone kept receiving notifications of his journeys long after he eventually logged out.
- The resident now seeks $45 million in damages from Uber, claiming the privacy flaw cost him his marriage.
- Android users should be safe as the vulnerability appears to be limited solely to "iOS versions of the app updated" after December 16.

Software Disasters

Hawaii Sends Out a State-Wide False Alarm About a Missile Strike

- In January 2018, the citizens of Hawaii were notified to take immediate cover in the face of an inbound ballistic missile strike.
- It turned out to be a false alarm, although it took over 30 minutes (and, presumably, several thousand heart attacks) before the alert was retracted.
- Investigations found that while the problem was largely due to human error, there were “troubling” design flaws in the Hawaii Emergency Management Agency’s alert origination software.

Software Disasters and the Importance of Proper Testing

In Conclusion, Testing is one of the major parts of any development and change in any software. The overall objective of testing is not to find every software bug that exists but to expose situations that could negatively impact the customer, maintainability, and usability.

Software is like gardening – one day I'll go behind the shed and clean-up. But if nobody ever goes there, does it matter a lot?

Mike Kreiger