 ISSN : 0976-8491 (Online) | ISSN : 2229-4333 (Print)

A Bug Ignorance Causes Big Expensive on Smart Systems

[1]Sri Krishna Chaitanya Rudraraju, [2]M.Krishna
[1,2]Dept. of CSE, Sir CRR College of Engineering, Eluru, AP, India

Abstract
Computers are the pervasive technology of our time. As computer become critically tied to human life, it also becomes more important that interactions with them are under control. They are no longer a novelty, but are integrated into the fabric of our world, performing both high and low-level tasks. That is, computers may be used to eliminate heavy, redundant work and more. Sophisticated machines have been deployed to perform remote surgery or detect subterranean landmines in repopulated civilian areas. The increasing importance of computers in our lives means that it is essential that the design of computer systems incorporates techniques that can ensure reliability, safety and security. This paper will examine technological mishaps involving the use of computers. This review will include notorious software bugs that have affected finance, communication, transit, defense, health and medicine and others systems or industries. The sequence and etiology of these accidents will be discusses as well as how catastrophes may be avoided in the future through lessons and practices based on research.

Keywords
Component, Formatting,Style, Styling, Insert (Key Words).

I. Introduction

A. The Nature of Errors

Despite their obvious success, computer-related errors occur more frequently than necessary. To determine the best way to be avoided, one must identify the series of incidents that led to the "accident". There are several theories on the nature of errors that may occur in a system, defined as a set of human and non-human elements interacting to achieve a common goal. In a complex system, the cause may be harder to discern, for it may be impossible to identify discrete elements or map the relationships between elements of the set. Some theorists have suggested all errors are human errors and can manifest themselves in many forms. They can be slips, because an action conducted is not what is intended, or they may be lapses, such as memory failures or omissions. Mistakes can involve errors in planning, although action may proceed as intended. The situation could have been inadequately assessed, and/or could have involved a lack of knowledge.

Reason distinguishes between three kinds of human errors – active, latent and lack of blocks. Active errors are the ones which are immediately discernible, while latent errors are much harder to detect, and may require considerable analysis to discover or understand and missing unpredictable blocks of code. "The car hit the lamppost" is an example of an active error. If the driver had been under the influence of alcohol, or was driving 15 miles over the speed limit, these would be related but latent errors and he is unable to predict exact accident or failure. The latent errors that were deemed to have caused the Challenger accident were traced back nine years, and in the case of the faulty pacemakers recalled by Guidant, the latent errors were traced back four years. The active errors (like the errors in security which occurred in the attacks of September 11th, 2001) draw our attention, but the actual long-term improvements in system design and safety will only occur after the detailed latent analysis which must follow. Solution should be tested with high priorities based on accidents.

B. Computers and Risk

Risk assessment can indicate existing or potential system vulnerabilities. However, it is important to note, hazards can never be completely eliminated from all systems. The failure to design safety-critical systems could lead to loss of life, destruction to the environment and may result in financial penalties when used over time.

The various sources and the effects can characterize the nature of computer-related risk. Source of problems arising from software development includes: system conceptualization, requirements definition, system design and hardware and software implementation. Sources of problems in system operation and use are: natural disasters, animals, infrastructural factors, hardware malfunction and software misbehavior. Our increasing dependence on computational machines to perform both low and high-level tasks indicates that risk should be thoroughly identified and accommodated.

1. Case Studies Related to Failures

In a complex system, a long history of latent errors may be ignored or undetected until a catastrophic event occurs. We will discuss complex system failure by application, citing well-known software errors that have led to the dramatic loss of resources in the space, transportation, communication, government, and health care industries including:

- NASA Mars Surveyor Program (1998 &1999)
- Patriot Missile Defense System (1991)
- Iran Air Flight 655 (1988)
- AT&T Breakdown (1990)
- Guidant Technologies ICDs (2005)
- Therac-25 (1986)
- Challenger Space Shuttle Disaster (1986)
- Public In-convince

(i). Space

(a). NASA Mars Surveyor '98 Program

The Mars Surveyor Program deployed two spacecrafts to study Martian climate and the subsurface environment, each launched separately. Lockheed Martin Astronautics was selected by NASA as the prime contractor for the project. Both crafts, the Mars Climate Orbiter, launched on December 11, 1998 and the Mars Polar Lander, launched January 3, 1999, were lost. Both were launched at Cape Canaveral Air Station, USA, and carried instruments to map the surface of Mars and detect aspects of its environment, such as ice reservoirs of traces of water.

The Mars Climate Orbiter's failed mission was attributed to a software error involving calculation. A report issued by NASA states the root cause was "failure to use metric units in the coding of a software file, "Small Forces," used in trajectory models. An investigation revealed that the navigation team was calculating metric units and the ground calculations were in Imperial units. The computer systems in the crafts were unable to reconcile the differences resulting in a navigation error.

The second craft was the Mars Polar Lander. This spacecraft sent its last telemetry just prior to landing on the surface of Mars. The investigation report revealed that the most likely cause of failure

was a software error that mistakenly identified the cause of a vibration as the touchdown when it was the deployment of the legs. This could have signaled the engines to cut-off 40 meters above the surface rather than on touchdown. An alternative theory suggests that it was the inadequate preheating of catalytic beds for the rocket thrusters. The remains of the spacecraft remain space.

(ii). Defense

(a). Patriot Missile Defense System (1991)

During the Gulf War, a software error was attributed to the death of 28 soldiers when the US Patriot Missile Defense System failed to intercept an incoming Scud missile that struck military barracks [10]. The "accident" was attributed to an error in calculation. The systems internal clock was measured in tenths of seconds and the actual time was reported by multiplying the internal clock's value with a 24-bit fixed-point register. As a result, two systems intended to share a universal time, instead had independent system clocks.

(b). Aegis Combat System, Iran Air Flight 655 (1988)

On July 3, 1988, the Aegis combat defense system, used by the U.S. Navy, was involved in an incident in which USS Vincennes mistakenly shot down Iran Air Flight 655 in 1988 resulting in 290 civilian fatalities [11]. The investigation inquired into all the events which occurred prior to, during, and immediately following the engagement of Track Number (TN) 4131, later identified as Iran Air Flight 655. Timelines became essential elements of the investigation, particularly as regards the short time period (minutes and seconds) in which the Commanding Officer was required to make his decision to fire. This time period is referred to as the "critical time period" throughout the report.

Using the missile guidance system, Vincennes's Commanding Officer believed the Iran Air Airbus A300B2 was a much smaller Iran Air Force F-14A Tomcat jet fighter descending on an attack vector, when in fact the Airbus was transporting civilians and on its normal civilian flight path. The radar system temporarily lost Flight 655 and reassigned its track number to a F-14A Tomcat fighter that it had previously seen. During the critical period, the decision to fire was made, and U.S. military personnel shot down the civilian plane.

(c). Ariane 5, Flight 501 Failure

It took the European space Agency 10 years and $7 billion to produce Ariane 5, a giant rocket capable of hurling a pair of three-ton satellites into orbit with each launch and intended to give Europe overwhelming supremacy in the commercial space business. All it took to explode that rocket less than a minute into its maiden voyage last June, scattering fiery rubble across the mangrove swamps of French Guiana, was a small computer program trying to stuff a 64-bit number into a 16-bit space.

One bug, one crash. Of all the careless lines of code recorded in the annals of computer science this one may stand as the most devastatingly efficient. From interviews with rocketry experts and an analysis prepared for the space agency, a clear path from an arithmetic error to total destruction emerges.

(iii). Telecommunications

(a). AT&T Breakdown (1990)

In January of 1990, unknown combinations of calls caused malfunctions across 114 switching center across the Unites States. This series of events resulted in 65 million calls unconnected nationwide. The company had a marketing campaign based on "reliability claims" of a "virtually foolproof system". The cause was attributed to a sequence of events that triggered an existing software error.

The cause of the AT&T network failure was also a "fault in the code". The improvement that was made improved the way the company reacts to the series of events that resulted in the 1991 system failure and uses internal logic to monitor switch activity.

(iv). Health Care and Medicine

Many people around the world are harmed as a direct result of medical errors that can occur while receiving medical treatment. These can be made by the human expert, novice or introduced by the computer design. Medical errors are alarmingly common, costly, and often preventable. The IOM report, To Err is Human, Building a Better Health System, increased public and political attention to the prevalence of medical errors in the Unites States. The IOM set a clear goal of a 50% reduction in medical errors over the next five years. Although the United States did not meet this benchmark, many policy makers believe this is achievable by automating informal processes and standardizing information for electronic exchange among doctors' offices, hospitals, laboratories, patients, the government, and others. However, the adoption of health information technology has been a slow process with moderate success.

(a). Guidant's Implantable Cardioverter-Defibrillator (2005)

In 2005, after notifying the FDA, Guidant, a subsidiary of Boston Scientific, urgently recalled 26,000 defective ICDs. Shortly after the FDA issued a Preliminary Public Health Notification and identified Guidant as the maker of three different models of pacemakers that the FDA report identifies as having a malfunction the "could lead to a serious, life-threatening event".

Earlier that year, two physicians, Hauser and Maron, published an article a medical journal reporting the death of a 21-year old patient who had received a Prizm 2 DR model 1861 ICD pulse generator in 1991. While jogging with his girlfriend, the patient went into sudden cardiac arrest and could not be resuscitated. His ICD was returned to Guidant and they determined that the device failed during the delivery of a shock. This directly attributed to a short circuit that developed between a high-voltage wire and tube used to test the housing (casing) during manufacturing. At this point in time, Guidant was aware of 25 other device failure reports on this same model and manufacturing changes were made in 2002 to prevent short-circuiting. However, Guidant declined to inform patients or physicians and felt such a communication was inadvisable and unnecessary. The physicians, who felt it was their moral and ethical responsibility to disclose the device failure to the medical community and the public, took their concerns to the New York Times, triggering intense media attention and a wide-scale, highly publicized device recall.

(b). Canadian Cancer Therapy Machine (Therac-25, 1986) Designed by Atomic Energy of Canada, Ltd. (AECL)

Therac-25 was a software controlled radiation therapy machine used to treat people with cancer. Between 1985 and 1987 Therac-25 machines in four medical centers gave massive overdoses of radiation to six patients. An extensive investigation and report revealed that is some instances operators repeated overdoses because machine display indicated no dose given. Some patients

received between 13,000 - 25,000 rads when 100-200 needed. The result of the excessive radiation exposure resulted in severe injuries and three patients' deaths.

Causes of the errors were attributed to lapses in good safety design. Specific examples are cite failure to use safety precautions present in earlier versions, insufficient testing, and that one key resumption was possible despite an error message. The investigation also found calculation errors. For example, the set-up test used a one byte flag variable whose bit value was incremented on each run. When the routine called for the 256th time, there was a flag overflow and huge electron beam was erroneously turned on.

An extensive investigation showed that although some latent error could be traced back for several years, there was an inadequate system of reporting and investigating accidents that made it hard to determine the root cause. The final investigations report indicates that during real-time operation the software recorded only certain parts of operator input/editing. In addition, the radiation machine required careful reconstruction by a physicist at one of the cancer centers in order to determine what went wrong.

(v). Public In-Convince

(a). Student Loan Service

In August of 2006 a U.S government student loan service erroneously made public the personal data of as many as 21,000 borrowers on its web site, due to a software error. The bug was fixed and the government department subsequently offered to arrange for free credit monitoring service for those affected.

(b). Metro Rail Accident

A software problem contributed to a rail car fire in a major underground metro system in April of 2007 according to newspaper accounts. The software reportedly failed to perform as expected in detecting and preventing excess power usage in equipment on a new passenger rail car, and evacuation and shut down of part of the system.

(vi). Challenger Space Shuttle Disaster as Shown in fig. 1.

The Space Shuttle Challenger disaster occurred on January 28, 1986, when Space shuttle Challenger (mission STS-51-L) broke apart 73 seconds into its flight, leading to the deaths of its seven crew members. The spacecraft disintegrated over the Atlantic Ocean, off the coast of central Florida at 11:38 EST (16:38 UTC). Disintegration of the vehicle began after an O-ring seal in its right solid rocket booster (SRB) failed at liftoff. The O-ring failure caused a breach in the SRB joint it sealed, allowing pressurized hot gas from within the solid rocket motor to reach the outside and impinge upon the adjacent SRB attachment hardware and external fuel tank. This led to the separation of the right-hand SRBs aft attachment and the structural failure of the external tank. Aerodynamic forces broke up the orbiter.

The crew compartment and many other vehicle fragments were eventually recovered from the ocean floor after a lengthy search and recovery operation. The exact timing of the death of the crew is unknown; several crew members are known to have survived the initial breakup of the spacecraft. The shuttle had no escape system, and the impact of the crew compartment with the ocean surface was too violent to be survivable.

Fig. 1: Challenger Disaster

(a). Space Shuttle Columbia

Space Shuttle Columbia (NASA Orbiter Vehicle Designation: OV-102) was the first spaceworthy Space Shuttle in NASA's orbital fleet. First launched on the STS-1 mission, the first of the Space Shuttle program, it completed 27 missions before disintegrating during re-entry on February 1, 2003 near the end of its 28th mission, STS-107, resulting in the deaths of all crew members aboard.

Fig. 2: Columbia Disaster

Failure of computers is not breaking news today; however the study of these projects reveals new factors for analysis.

II. Overcome Failures

Software Process is a complex process. It should be handled with care and proper understanding. Capaturing Client and User requirements properly and transforming those requirements Software Project is art of technique. USA government itself is spending 60 Billion dallors on testing. Most of the projects have failures because of developers are not capaturing requirements properly, End user could not provide his/her requirements properly, parameter negligence, Software professionals could not come up with proper technology understanding, etc., Software success or failure depend upon end user. For better understanding conduct postmortem and iterate for next project.

A. 10 Tips for Success

We came from an engineering background and know that even within tech companies, there are lots of screwed up projects run by managers who don't fully understand the intricacies of software

development. Based on these experiences, I've come up with a list of things one should understand before agreeing to go ahead with a software projects.

1. Hire a Component Software Project Manager or consulting firm.
2. Hire Quality Developers/Testers/etc.,
3. Adopt an Iterative Development Process.
4. Adopt a Modeling Design technique.
5. Good Communication needs to be built into the process and culture.
6. Organize the Project Around small, semi-Autonomous Teams.
7. Ensure there is Accountability at all a level.
8. Periodically Review and tweak the process.
9. Postmortem success or failures.
10. Preserve volumes for future use.

B. Safety-Critical Practices

As opposed to concentrating on errors, other researchers have focused on what makes certain industries such as military aircraft carriers or chemical processing highly reliable. Their findings emphasize that accidents can be prevented through good organizational design and management. Success factors include a high organizational commitment to safety, high levels of redundancy in personnel and safety measures, and a strong organizational culture for continuous learning and willingness to change.

Research suggests failure to design safety-critical systems can be attributed to a mismatch between a technological system and a human operator, manifested at the "syntactic" and "semantic" levels. A knowledge representation can be syntactically correctable, but if its semantic structure is wrong, then no amount of human technology can correct it.

There is a common misconception that increasing reliability will increase safety. Many software-related accidents have occurred despite the software being compliant with the requirements specification. Semantic mismatch is characterized by errors that can be traced to errors in the requirements – what the computer should do is not necessarily consistent with safety and reliability. Here are the main issues for further consideration:

We have inscrutable software.

1. Who is writing the software and should they be certified?
2. Who is responsible?
3. Well-designed Human Interfaces are needed.
4. Consider Modeling Concepts which reduces complexity.
5. Human factors are critical - In many complex systems human factors are critical to whether the system will operate, be understood in adverse conditions, and how preventative measures may be taken.
6. Redundancy and self-Checking, and testing are needed.

III. Conclusion

The analysis of case studies pertaining to common and severe failures depicts that a software failure at any stage could lead to the loss of lives, financial losses, wastage of time, effort and other intangible losses like discomfort, stress, good will, reputation, confidence, peace etc. In current information age the application of software has penetrated in each and every industry unlike traditional approach where software was altogether a separate entity. As software has become integral part of every product and process so there is a need to make a full proof system so that the software failures could be avoided. There is further requirement of root cause analysis of these software failures to understand the problematic area and suggest the areas of improvement in the current process as several corrective & preventive actions needs to be taken while developing products and software systems.

Already a threshold point started for the failures of software and let us all try to enhance our future software for better society. Where we can keep our coming generations safe and acknowledge them about failures which can occur. This is possible with the team work and better communication between the people who related to the software project.

References

[1] Divya Manusha Seethalam, Valiveti Karthik, Valluru Gowthami, Gudikandula Radha Krishna Murthy, Prathipati Ratna Kumar,"Software Bug lliputians which cause Giant Damage to System", 2012.

[2] Reason J. Human Error. Cambridge, Cambridge University Press, 1990.

[3] Gornick CC, Hauser RG, Almquist AK, Maron BJ. Unpredictable implantable cardioverter-defibrillator pulse generator failure due to electrical overstress causing sudden death in a young high-risk patient with hypertrophic cardiomyopathy. Heart Rhythm. 2005 Jul. 2(7), pp. 681-3.

[4] Hauser RG, Maron BJ,"Lessons from the failure and recall of an implantable cardioverter-defibrillator", Circulation. 2005 Sep 27;112(13): 2040-2. Epub 2005 Sep 19.