http://www.g2zero.com/2006/07/notable_entries_from_the_softw_1.html

July 06, 2006

Entries from the Software Failure Hall of Shame, Part 1

If your organization is lucky and competent enough to have a popular or high profile software project or offering, it may be only a latent defect or unknown security vulnerability away from global notoriety. There are hundreds of examples of software-related failures, some minor and some catastrophic and resulting in the loss of millions of dollars and even deaths.

The point is that in hindsight, organizations responsible for buggy software projects would have preferred to invest in improved quality tools and processes rather than face the backlash, market loss, and business disruption caused by the defects.

Examples of some of the more notable software failures from a wide variety of industries include:

- **The Toyota Prius engine management flaw.** In October of 2005, the Toyota Motor Company voluntarily recalled 75,000 of its hybrid vehicles because a software glitch that may have shut down the engine. Given the high price of gasoline at the time and the rising interest from consumers in hybrid vehicles, the recall could have been a major blow to the manufacturer. However, due to Toyotas quick response, most consumers never experienced the flaw, and while the company may have suffered slightly from the negative publicity, it managed to avoid having its defect become permanently associated with the vehicle line or with hybrid safety.

- **Failing Sony televisions.** As in the Toyota example, Sony wasn't sure how many of its 400,000 LCD and rear-projection TVs would show signs of a software defect, but the company decided it had to upgrade each of the sets. The software bug and remediation effort, reported in February of 2006, meant that in many cases Sony would have to send a technician to manually update the TVs. By proactively addressing the issue, Sony has so far managed to avoid intense media scrutiny or customer backlash.

- **Incredibly cheap US Airways tickets.** In April of 2005, the ticketing system for US Airways issued incorrect fares for several hours. Some tickets were offered for under $2. The system was quickly fixed, but the airline felt compelled to honor the drastically reduced fares. While this may have been the right decision from a public relations standpoint, the loss of revenue certainly didn't help the company as it struggled to work its way out of bankruptcy.

- **Numerous FAA issues.** In November 2005, the Federal Aviation Administration rolled out software patches to Boston and other airports to improve ground-based radar systems. The software it was replacing had a defect that didn't allow the system to see two planes approaching each other on the runways. In another example, a software failure from a backup system that was designed to handle planned server downtime resulted in a three-hour loss of air traffic control communication between over 800 planes, with five near misses.

- **The Marines' Osprey crashes.** In December 2000, the US Marine Corps' new hybrid plane-helicopter, known as the V-22 Osprey, crashed, killing four Marines. The accident was

the result of a failure in a hydraulic line, compounded by a software error. The report on the accident stated that the software defect caused “rapid and significant changes to prop-rotor pitch," which were compounded as the pilot continued to reset the system as trained, resulting in an increasingly unstable aircraft. While the Osprey program continues, this and other crashes leads many to believe that the V-22 is an inherently flawed aircraft.

- **The Mars Climate Orbiter loss.** In 1999, NASA lost contact with one of its well-publicized unmanned spacecraft, the Mars Orbiter. During the subsequent investigation into its loss, many issues that lead to its demise were uncovered. However, the final cause was later determined to be a result of a problem within a software application that was responsible for converting differing units of measurement. Specifically, “The 'root cause' of the loss of the spacecraft was the failed translation of English units into metric units in a segment of ground-based, navigation-related mission software.” Other recent successes, such as the longer than expected life of the two new Martian robot rovers, have pushed the Orbiter loss to the periphery, although you can bet that NASA software engineers will always check their conversion code for all future spacecraft.

- **The destruction of an Ariane 5 rocket.** In 1996, a new version of the Ariane rocket exploded on its maiden flight. This was exactly the opposite outcome that the European rocket consortium responsible for the rocket had hoped for, as its fiery demise was and is the type of high profile disaster story that the media latches onto. The panel that investigated the explosion’s cause determined that a faulty computer program was to blame. The loss was attributed to software that shut down because of an internal variable exceeding a limit imposed by the underlying code.

- **The USS Yorktown ends up adrift.** With all the threats to modern military ships, from missiles to mines to more exotic weapons, the least appealing way to have a ship disabled would be an internal software glitch. But that is just what happened to a prototype US Navy ship in the fall of 1997. When attempting to adjust a valve setting, a sailor entered a zero into a database field. The result? The ship was out of action for two hours. The fault was traced to an internal system, running on 27 remote terminals with Windows NT front ends, which did not have the proper data field level validation. For the Navy and the software company that wrote the computer code, the embarrassment was tempered only by the fact that the error occurred during testing, not during a military engagement.

  Interesting, this, like many other software failures, often spawns a variety of damaging urban legends, in this case impacting the reputation of Microsoft. A common version of the story among IT professionals is that a Windows bug disabled a Navy ship. While Microsoft has been at the center of many reports and discussion of software quality, in this instance, a contractor’s sloppy programming, not its software, was to blame.

- **The WMF bug.** There are many instances in which Microsoft software has failed; particularly security vulnerabilities that have given the company a reputation for buggy code. The ways the vendor has reacted to it real and reported software defects in its high profile flagship products like Windows and Office illustrate how the issue of responding to software defects is now a major challenge. In late 2005, a security vulnerability in a ubiquitous component of Windows became widely discussed in the trade press and technical forum and blog world. Since the code was present even in up-to-date products and the potential for malicious exploitation was believed to be high, the desire for a quick patch was strong.

When the official patch was seen as taking too long to arrive, a non-Microsoft developer took it upon himself to write, and then release his own patch. The non-official patch garnered major IT industry publicity when some security firms suggested companies test and then install it in lieu of the official remedy.

What is so intriguing about this event is that the Microsoft response was not slow, but due to the global spread of information and exploit code, many felt that waiting two weeks was too long. Now, not only the defects, but the reaction to a software problem, requires serious attention by software providers.

About the author: Tom Rhinelander is an analyst with the New Rowley Group. He has written and worked extensively with industry vendors and user companies on a wide variety of technology issues, including the improvement of software quality.

Posted by Admin on July 6, 2006 10:03 AM | Permalink