FAQ-01
In General, are there problems with software development?

Software Problems: Industry Observations

As software becomes increasingly important, the potential impact of bad code will increase to match, in the view of Peter G. Neumann, a computer scientist at SRI International, a private R&D center in Menlo Park, CA. In the last few decades, software defects have wrecked a European satellite launch, delayed the opening of the hugely expensive Denver airport for a year, destroyed a NASA Mars mission, killed four marines in a helicopter crash, induced a U.S. Navy ship to destroy a civilian airliner, and shut down ambulance systems in London, leading to as many as 30 deaths. And because of our growing dependence on the Net, Neumann says, "We're much worse off than we were five years ago. The risks are worse and the defenses are not as good. We're going backwards-and that's a scary thing."[1]

"Software that performs well is useless if it ultimately fails to meet business user requirements. But while research study after research study shows that requirements errors are the number one cause of software project failures (costing U.S. companies approximately $30 billion a year in total), many organizations continue to deliver requirements that are unclear, ambiguous or incomplete. These erroneous requirements then go on to be tracked, traced and perfectly executed, resulting in a system that "does what was required" but unfortunately doesn't do what business users thought they asked for." [2]

"Requirements, as every CIO knows, are a problem, but CIOs may not be aware of just how catastrophic the problem has become. Analysts report that as many as 71 percent of software projects that fail do so because of poor requirements management, making it the single biggest reason for project failure—bigger than bad technology, missed deadlines or change management fiascoes." [3]

Software Industry Common Characteristics and Trends[4]

- The vast majority of software problems are traceable to errors made during the specification phase within the development process.
- One of the most significant features of software is branching, i.e., the ability to execute alternative series of commands, based on differing inputs. This feature is a major contributing factor for another characteristic of software - its complexity. Even short programs can be very complex and difficult to fully understand.
- Typically, testing alone cannot fully verify that software is complete and correct. In addition to testing, other verification techniques and a structured and documented development process should be combined to ensure a comprehensive validation approach.
- Unlike hardware, software is not a physical entity and does not wear out. In fact, software may improve with age, as latent defects are discovered and removed. However, as software is constantly updated and changed, such improvements are sometimes countered by new defects introduced into the software during the change.
- Unlike some hardware failures, software failures occur without advanced warning. The software's branching that allows it to follow differing paths during execution, may hide some latent defects until long after a software product has been introduced into the marketplace.

Software Industry Common Characteristics and Trends[4] (Cont.)

- Another related characteristic of software is the speed and ease with which it can be changed. This factor can cause both software and non-software professionals to believe that software problems can be corrected easily. Combined with a limited understanding of software, it can lead managers to believe that tightly controlled engineering is not needed as much for software as it is for hardware. In fact, the opposite is true. **Because of its complexity, the development process for software should be even more tightly controlled than for hardware, in order to prevent problems that cannot be easily detected later in the development process.**
- Seemingly insignificant changes in software code can create unexpected and very significant problems elsewhere in the software program. The software development process should be sufficiently well planned, controlled, and documented to detect and correct unexpected results from software changes.
- Given the high demand for software professionals and the highly mobile workforce, the software personnel who make maintenance changes to software may not have been involved in the original software development. Therefore, accurate and thorough documentation is essential.
- Historically, software components have not been as frequently standardized and interchangeable as hardware components. However, medical device software developers are beginning to use component-based development tools and techniques. Object-oriented methodologies and the use of off-the-shelf software components hold promise for faster and less expensive software development. However, component-based approaches require very careful attention during integration. Prior to integration, time is needed to fully define and develop reusable software code and to fully understand the behavior of off-the-shelf components.

Do Industry and Other Government Agencies have Systemic Problems with Software?

Software bugs, or errors, are so prevalent and so detrimental that they cost the U.S. economy an estimated $59.5 billion annually. [5,12]

- Between 1992 and 1998, the FDA reported that 8% of all medical devices recalled were due to software failures. [6]
- Radiation Cancer Therapy Machine Mishaps in 1985-86 due to Safety Critical Software Control Errors [7]
  - In a 20-month period between June 1985 and January 1987, Therac-25 radiation therapy machines used in both the US and Canada administered massive overdoses of electron beam radiation to at least six cancer patients, with at least three deaths attributed to radiation overdose.
  - Frequent malfunctions were being reported (up to 40 per day per machine) with no effective manufacturer response or information sharing among users.
  - Operators performed setup more quickly than accommodated by design.
  - Software safety interlock had an unknown “race” condition (1/256) that manifested itself only during these “quick” setups.

Do Industry and Other Government Agencies have Systemic Problems with Software? (cont)

- Software plays an ever more critical role in the automotive industry. Software errors result in expensive vehicle recalls, a damaged reputation and even the loss of human life. [8]

- Recalls:
  - 62,369 vehicles; due to antilock brake system software
  - 5,902 vehicles; air bag control unit improperly sets a fault code causing the airbag not to deploy.
  - 127,928 vehicles; fuel pump module and the power train control module (PCM) software.

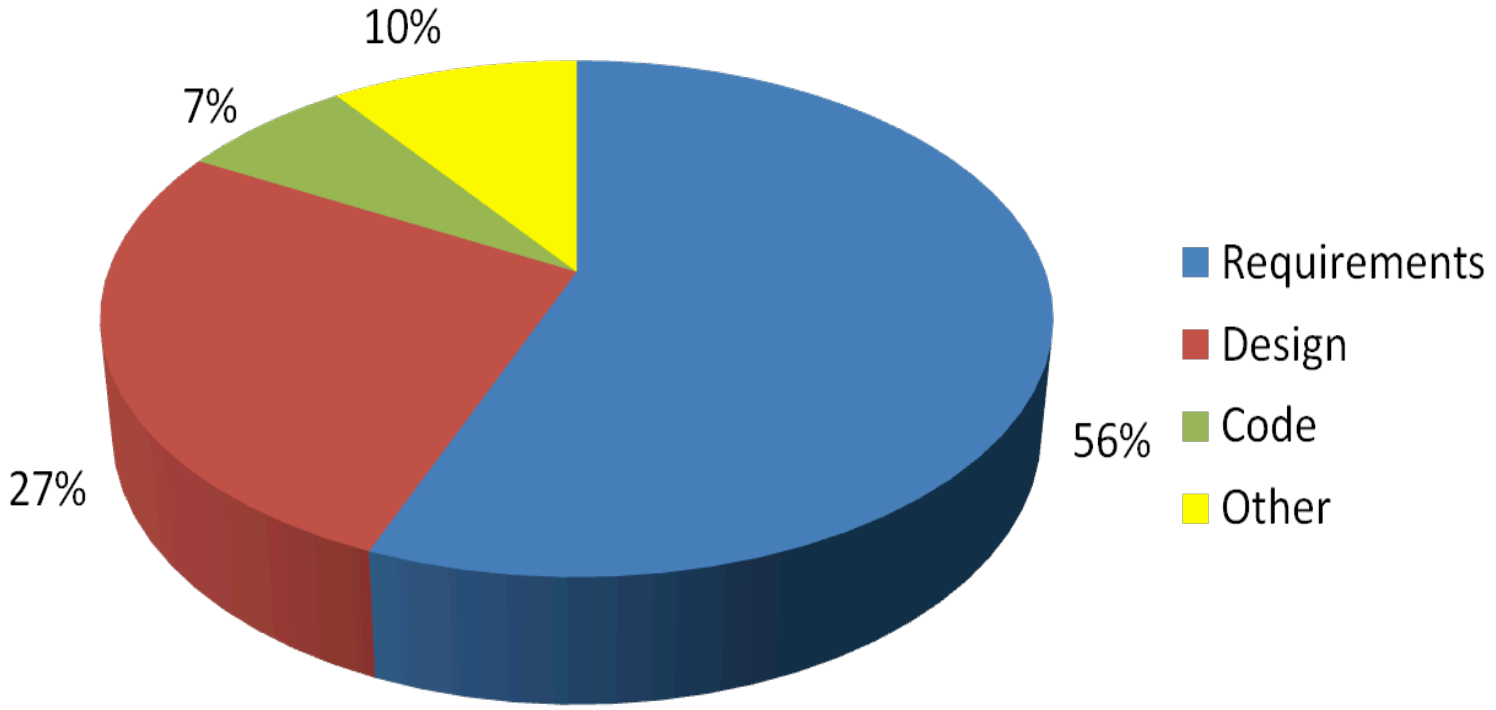
- The Software Engineering Institute estimates that 90 percent of reported cyber-security incidents result from exploits against defects in the design or code of software. [9]

Do Industry and Other Government Agencies have Systemic Problems with Software? (cont)

Colossal Software Failures: [10]

- 2004 – Hudson Bay Co. - Problems w/ software inventory contribute to $33.3 Million loss
- 2004 –UK Ireland Revenue - Software Errors contribute to $3.45 Billion tax credit overpayment
- 2004 – Avis Europe – ERP System cancelled after $54.5 Million investment
- 2004 – Ford Motor Co. – Purchasing System abandoned after deployment; $400 Million investment
- 1996 – Ariane 5 rocket launched by the European Space Agency exploded just forty seconds after its lift-off ; software error in the inertial reference system; specifically a 64 bit floating point number relating to the horizontal velocity of the rocket with respect to the platform was converted to a 16 bit signed integer (Software reused from Ariane 4).

Distribution of Software Bugs: Industry Report[11]



Requirements defects are 56% of total defects in systems per industry report.

Bibliography

1	Mann, Charles, "Why Software is So Bad," MIT Technology Review, July 2002.
2	Dr. Joe Marasco, "The Importance of Testing Software Requirements," SearchSoftwareQuality.com, Oct. 8, 2007.
3	Christopher Lindquist, "Fixing the Software Requirements Mess," www.cio.com, Nov. 15, 2005.
4	General Principles of Software Validation; Final Guidance for Industry and FDA Staff, U.S. Food and Drug Administration; January 11, 2002
5	National Institute of Standards and Technology (NIST), Press Release (for Planning Report 02-3 referenced below), June 28, 2002 http://www.nist.gov/public_affairs/releases/n02-10.htm
6	U.S. Food and Drug Administration, General Principles of Software Validation, Final Guidance for Industry and FDA Staff, January 11, 2002 http://www.fda.gov/cdrh/comp/guidance/938.html
7	Leveson, Nancy, Safeware - System Safety and Computers, Addison Wesley, 1995, Reading, MA, 680 pp. http://sunnyday.mit.edu/papers/therac.pdf
8	Independent Verification and Validation of Automotive Software, Rovsing, http://webserver.rovsing.dk/uploads/File/brochures/Automotive_IVV_small.pdf
9	Software Assurance, Department of Homeland Security, http://www.uscert.gov/reading_room/infosheet_SoftwareAssurance.pdf
10	Software Hall of Shame, http://spectrum.ieee.org/sep05/1685/failt1
11	Gary E. Mogyorodi, "What is Requirements Testing?", CrossTalk, Vol 16, No. 3, Mar 03.
12	Planning Report 02-3, The Economic Impacts of Inadequate Infrastructure for Software Testing National Institute of Standards and Technology (NIST); May 2002, http://www.nist.gov/director/prog-ofc/report02-3.pdf

Backup Material

Robotics Severity Definitions

- Severity 1

  a) Prevent the accomplishment of an essential capability
  b) Jeopardize safety, security, or other requirement designated critical

- Severity 2

  a) Adversely affect the accomplishment of an essential capability and no work-around solution is known
  b) Adversely affect technical, cost or schedule risks to the project or life cycle support of the system, and no work-around solution is known

Human-Rated Severity Definitions

- Severity 1: A failure which could result in the loss of the human-rated system, the loss of flight or ground personnel, or a permanently disabling personnel injury.

- Severity 1N: A failure which would otherwise be Severity 1 but where an established mission procedure precludes any operational scenario in which the problem might occur, or the number of detectable failures necessary to result in the problem exceeds requirements.

- Severity 2: A failure which could result in loss of critical mission support capability.

- Severity 2N: A failure which would otherwise be Severity 2 but where an established mission procedure precludes any operational scenario in which the problem might occur or the number of detectable failures necessary to result in the problem exceeds requirements.