Requirements-Related Risks
in Critical Systems

Peter G. Neumann
Principal Scientist
Computer Science Lab
SRI International, Menlo Park
California USA 94025-3493
Telephone 1-650-859-2375
E-mail Neumann@CSL.sri.com
http://www.csl.sri.com/neumann

International Conference on
Requirements Engineering
Schaumberg, Illinois
June 20, 2000

Abstract

This talk considers some of the roles that requirements engineering plays in computer system development, with particular emphasis on systems with critical requirements such as security, reliability, safety, and survivability. The RISKS archives are littered with cases attributable to requirements problems that propagate throughout development, from which many lessons need to be learned. Various possible remedies are discussed.

Acknowledgment of Support
and Contractual Disclaimer

Some of the work on which this talk is based was supported by the U.S. Army Research Laboratory (ARL) under Contract DAKF11-97-C-0020. Any opinions, findings and conclusions or recommendations expressed herein are those of the author and do not necessarily reflect the views of ARL.

System Requirements

- Critical information systems and their applications must predictably and dependably satisfy relevant stringent requirements relating to security, robustness, reliability and fault tolerance, human safety, performance, and survivability in the face of many adversities.
- Explicit requirements are also needed for interoperability, evolvability, maintainability, ease of use, and any special characteristics of the user interface.
- These requirements must be accurate, comprehensive, not overly specific, and well-defined up front.

Other System Needs

Other important factors may also need to be represented explicitly as requirements relating to

- Desired assurance levels for the resulting systems
- Software-engineering practice
- Architectures that can aid effective operational practice, as well as explicit requirements on system operation (normally a huge source of risks)

Some Generic Problems in Software Development and Use

- Developing robust systems is inherently difficult.
- Today's "Best Practices" and criteria are inadequate and poorly applied.
- Software "engineering" is typically not engineering, with sloppy requirements, never-ending bounds and buffer overflows, other flaws, ...
- Operational management is difficult and riskful.
- These problems are ubiquitous.

System Realities Today

- Distributed and networked systems are riskful. Embedded systems are increasingly dependent on other systems rather than truly standalone.
- Personal-computer OSs and software are typically flawed, bloated, inflexible, difficult to administer, not suitable for critical applications. Servers and firewalls are often misconfigured. Network protocols are inadequate. There are many risks.
- "Defense in Depth" is more like "Weakness in Depth". There are too many weak links.

System Failures and Misuses Due to Inadequate Requirements: Reliability 1

- USS Yorktown 2.75-hour outage due to unchecked divide-by-zero
- Patriot missile inaccuracies: system clock drifted; daily reboot had been (tacitly?) assumed!
- Iran Air Airbus shootdown: no requirement for rate of change of altitude on-screen, no IFF reauthentication; archaic displays
- Strategic Defense Initiative, Star-Wars Light: unimplementable, even without specifying requirements! Lacking in common sense

System Failures and Misuses Due to Inadequate Requirements: Reliability 2

• 1980 ARPAnet collapse due to faulted status messages: no parity checking in memory; unsound garbage collection

• 1986 ARPAnet Northeast cutoff: 7 trunks in single conduit; 1991 redundant fiber circuits in adjacent cables, Annandale VA

• The Year-2000 Problem: Massive shortage of sensible requirements!!!

System Failures and Misuses Due to Inadequate Requirements: Reliability 3

• Northwest Airlines 255 over Detroit. Flaps, thrust indicator computer not properly set, but not detected because warning system was down ($13 breaker) and its absence was not detected! Who checks the checker?

System Failures and Misuses Due to Inadequate Requirements: Safety 1

• Therac 25: no requirement for software interlock that was in hardware in Therac 20

• Heart-pacemaker deaths (arthritis therapy, anti-theft device), defibrillator death (cellphone); Black Hawk helicopters: no requirement for shielding out electromagnetic interference

• Heart-monitor plug fit electrical wall socket, killing patient: No requirement against it!

System Failures and Misuses Due to Inadequate Requirements: Safety 2

As an extreme example of the importance of clear requirements for and definition of the human interface, consider the gas-tank controls for John Denver's last flight (as a pilot); the control lever positions were thus:

• Up for off
• Right for left-tank
• Down for right-tank

(He thought he had run out of fuel? Were the positions clearly labelled?)

System Failures and Misuses Due to Inadequate Requirements: Security

- Denial of service attacks: almost no defenses whatsoever!
- Numerous USGovt Website hacks: poor authentication, security
- Cryptohacking (DVD, GSM, etc.): weak requirements, poor design
- John Deutch, Wen Ho Lee, Dept of State laptops: no requirement for multilevel system security or for physical controls; bad management, bad operational practices

Development Fiascos

- IRS Tax Systems Modernization, \$4B: stovepipe design, no real interoperability requirements
- FAA Air-Traffic Control System, \$4B: inadequately specified requirements, development problems
- Development problems in operating systems and applications: creeping featurism, bloatware, incompatibilities, almost no critical requirements – especially for security and reliability

Generic Desiderata

- Discipline in requirements, development, software distribution, operation, evolution, evaluations, education, training, ...
- Inherently robust secure evolvable interoperable architectures
- Responsible operational support and configuration control
- Open standards for software and, interfaces, composability, interoperability, distribution; analysis of available source
- Contracts, liabilities, incentives: compliance bonuses, noncompliance penalties

Robust Architectures

- Architectures that avoid excessive dependence on untrustworthy components
- Thin-client user platforms with minimal operating systems, where trustworthiness is required only where essential
- Well-configured & well-managed firewalls, trustworthy servers distribution paths for software, provenance on all critical software; nontrivial user authentication, bilateral peer authentication, aggressive resistance to denials of service, better protocols, ...

More on Robust Architectures

• Nonsubvertible implementations of cryptography, used pervasively, including cryptographic integrity
• Run-time detection of malicious code and misuse
• Wireless applications and mobile code add some stringent further requirements.
• Requirements need to inspire sensible system designs and architectures, but not overconstrain implementations.

Conclusions

• Requirements are a vital first step. Blame is often widely distributable, but bad requirements are where many developments go wrong. Upfront effort pays off big-time.

• Requirements must realistically encompass all critical functionality, and anticipate conflicts and essential tradeoffs among satisfaction of different requirements.

• Discipline is as important in requirements engineering as anywhere else in the development cycle and operational practice.

References

• Peter G. Neumann: papers, reports, testimonies, survivability course notes, RISKS, Illustrative Risks, CACM Inside Risks, etc.: `http://www.csl.sri.com/neumann`

• PGN, Computer-Related Risks, Addison-Wesley, 1885

• PGN, Practical Architectures for Survivable Systems and Networks, SRI final report (ARL), June 2000; `http:/www.csl.sri.com/arl-two.html` (on-line as soon as approved for public release)