Computers in Safety-Critical Systems

Ethics and Computing

Chapter 6

Motivation

- When human welfare is at stake, the price for haphazard practices is severe.
- Computing professionals must exercise extreme care to ensure a system is safe.
- Striving for safe systems is an important part of any Code of Ethics.

Codes of Ethics and Safety

- AITP Standards of Conduct
  - To the best of my ability, insure that the products of my work are used in a socially responsible way.
- Software Engineering Code of Ethics
  - Ensure adequate testing, debugging, and review of software and related documents on which you work.

ACM Code of Ethics

1. Strive to achieve the highest quality, effectiveness and dignity in both the process and products of professional work.

5. Give comprehensive and thorough evaluations of computer systems and their impacts, including analysis of possible risks.

IEEE Code of Ethics

1. Accept responsibility in making engineering decisions consistent with the *safety, health and welfare of the public,* and disclose promptly factors that might endanger the public or the *environment.*

3. Be *honest and realistic* in stating claims or estimates based on available data.

IEEE Code of Ethics

6. Maintain and improve our technical *competence* and undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations.

Safety-Critical Systems

Requirements for computing professionals:

1. Know techniques for developing computer systems that are as safe as practical
2. Be able to arrive at a reasonably objective assessment of the level of safety

Factors Involved in Accidents

- Organizational
- Managerial
- Technical
- Sociological
- Political

Safety-Critical Systems

Definition 1: Systems with some real-time control whose failure can have a life-threatening impact.

Examples

- Aircraft/air traffic control
- Nuclear reactor control
- Missile systems
- Medical treatment systems

Safety-Critical Systems

Definition 2: Systems with some real-time control, *including the software used in the design of physical systems and structures*, whose failure can have a life-threatening impact.

Examples

- Bridge and building design
- Selection of waste-disposal sites
- Analytical models of medical treatment

Engineering a Minimum-Risk System

- "Managing Murphy's Law: Engineering a Minimum-Risk System" [Bell, 1989]
  - Safety should be a design objective
  - Risk, hazard and reliability
  - Conducting a thorough risk assessment of a system

Safety-Related Terminology

- Reliability: probability that a system component will perform its intended function for a specified period of time under normal conditions
- Hazard: potential for injury or danger
- Risk: combination of the probability of an undesired event with the magnitude of foreseeable consequences (e.g., property damage, personal injury)

Assessing Risk

- Tabulate phases of system's mission
  - Identify risk sensitivities
  - Varied time-related consequences
- Diagram each phase and determine logical relationships among components
  - Failure modes and effects analysis
  - Event tree analysis
  - Fault tree analysis
- Quantify probability of failures

Managing Risk

- Determine acceptable levels of risk
- Subjective judgments about system costs and benefits (e.g., human life, environment)
- Additional safety-related add-on systems
  - Redundant or standby systems
  - Safety shutdown systems
  - Emergency procedures
- Uncertainty

How and Why Failures Occur

- "Avoiding Weak Links" in the *Inside Risks* column of *Communications of the ACM* [Neumann, 1992] (1st edition)
  - Difficulty in assessing multiple-event failures
  - Independent multiple events
  - Correlated multiple events
  - Multiple-event failures do arise and should be better defended against

Independent Multiple Events

- Independent tests err harmoniously
  - Three independent test methods used in aircraft flutter analysis
  - Each test found no flutter problem
  - Each test in error for independent reasons
  - False conclusion exposed only after tail broke off in test flight, killing the crew

Independent Multiple Events

- Simultaneous failures
  - Three disk drives supporting the Toronto Stock Exchange each failed independently
  - Exchange shut down for three hours on August 16, 1989

Correlated Multiple Events

- In 1989, New England maintained seven logical alternative links to ARPANET
  - All seven physical links in one cable
  - Cable accidentally cut
- Associated Press lost both primary and backup connections (purposely separated cables) when both accidentally cut

Correlated Multiple Events

- Four-hour collapse of ARPANET in 1980
  - Absence of parity checking in memory (implementation deficiency)
  - Concurrent existence of a legitimate status message along with two bogus versions (dropped bits)
  - Weak garbage-collection algorithm (design oversimplification)

Correlated Multiple Events

- AT&T long distance saturation in 1990
  - Inability to detect load- and time-dependent flaw in switch recovery software, despite extensive testing
  - Presence of flaw in every switching system of that type
  - Untolerated hardware fault mode that caused a switch to crash and recover
  - Due to heavy traffic, neighboring switches unable to adjust
  - Repeated propagation of the effect throughout all switches

How and Why Failures Occur

- "How Engineers Lose Touch" [Ferguson, 1993]
  - Engineers must verify that the system analyzed on paper corresponds well to the one in the field
  - A computer model is an abstraction, not an exact depiction of the system in the real world
  - Viewing models as abstractions serves as a warning that not all aspects of the environment may be accurately modeled
  - Nearly all engineering failures result from faulty judgments, not faulty calculations

Risk Analysis

- "The Limits of Risk Analysis" [Bell, 1989] (1st edition)
  - Responsibility for a system rests on the system's top managers
  - Managers must seriously consider and act on results of risk analysis
  - Risk analysis is only a tool to help managers make informed decisions

Evaluating Safety-Critical Software

- "Evaluation of Safety-Critical Software" [Parnas et al., 1990]
  - Standards for safety-critical applications
  - Documentation requirements
  - Testing requirements
  - Guidelines for software structure

Evaluating Safety-Critical Software

- Software exhibits *weak-link* behavior
  - Failures in unimportant parts of the code can have unexpected repercussions elsewhere
- Recommends safety-critical software be as small and simple as possible
  - Move non-safety-critical functions to other computers
- All parts of safety-critical software is safety critical

Software vs. Hardware Controllers

- Software more complex
- Software more sensitive to errors
- Software harder to test
- Software failures correlated in design, not manufacture or use
- Lack of professional software engineering standards (coming)

Software Testing Concerns

- Software cannot be tested for correctness
- Software reliability and availability are difficult to predict accurately
- Software trustworthiness (lack of serious flaws) cannot be practically measured
- Software testing still important
- Software testing benefits from independent validators (e.g., IBM's *clean room*)

Software Reviews

- Review for correct intended function
- Review for maintainable, understandable, well-documented structure
- Review modules to verify algorithm and data structure design are consistent with specified behavior
- Review code for consistency with the algorithm and data structure design
- Review test adequacy

Measuring Software Reliability

- *Reliability* is the probability that an input will not cause a failure
- *Trustworthiness* is the probability that no serious design error remains after the software passes a set of randomly chosen tests
- *Availability* is the fraction of time the system is running and assumed to be ready to function

Measuring Software Reliability

- Software failure rates cannot be predicted from failure rates of individual lines of code or subprograms
- Recommends finite-state machine model of software
  - Failure rates viewed as percentage of unacceptable (*next-state*, *output*) behavior
- Statistical hypothesis testing

Case Study

- "An Investigation of the Therac-25 Accidents" [Leverson and Turner, 1993]
  - Failure in a radiation therapy system (1985-87) resulting in six incidents of massive overdoses
  - Several failures traced to errors in special timing-dependent sequences of events in the user interface
  - Software errors cited, although hardware and software interlocks could have prevented hazards

Points to Remember

- Risk analysis is a difficult task that has a subjective component
- Assumptions about the logical independence of different types of failures are often not supported by the physical realization of the system
- A software model of a real physical system can never perfectly represent all relevant aspects of the system
  - Over-reliance on computer models that are not properly validated invites disaster

Points to Remember

- Disregarding the contribution of software in a safety-critical system is a mistake
  - Software components do contribute to risk and failure
- Focusing only on the software component of a safety-critical system is a mistake
  - Most failures have multiple contributing causes
  - Most failures could have been prevented by improving any one of several system components

Points to Remember

- No widely accepted standard for developing safety-critical software
- There are many techniques for software design, evaluation, review and testing that can be used to produce higher quality software
- Professionals engaged in development of safety-critical software should know and use such techniques

Resources

- The Risks Digest
  - http://catless.ncl.ac.uk/Risks/
- CMU Software Engineering Institute
  - http://www.sei.cmu.edu/