Safety-Critical Systems

Rikard Land
Critical Systems

• Safety Critical Systems
  – Failure may injure or kill people, damage the environment
  – Example: nuclear and chemical plants, aircraft
  – (Example: Weapon industry. People will be killed if the systems work; but the right people…)

• Business Critical Systems
  – Failure may cause great financial loss
  – Example: information system. Customer information cannot be lost, or hacked

• Mission critical system
  – Failure may cause a mission to fail
  – Large values potentially wasted
  – Example: Space probe. Large sums of money, many years of waiting
References

• Leveson, *Safeware*, Addison-Wesley, 1995
• Collins, ”How Good is Good Enough?”, Communications Of The ACM, 1994

• Courtois and Parnas, ”Documentation for Safety Critical Software”, IEEE 1993
• Lutz, Robyn, ”Software Engineering for Safety: A Roadmap”, 2000
Terminology

• “Failure”
  – Software (or system) does not behave as specified
• “Mishap”
  – Includes accidents and harmful exposures
• “Hazard” or “hazardous state”
  – Mishap may occur but can still be avoided by counteraction or sheer luck
  – Examples: airplanes too close, nuclear core too hot
  – “Near miss” also considered safety problem
• Mishap ≠ failure
  – Failure does not necessarily lead to mishap
  – Several mishaps has occurred when the system was operating exactly as expected, i.e. without failure
Tradeoffs

- Often tradeoff between safety and other requirements, e.g. reliability and availability
  - The safest system is sometimes a system that does not work at all!
  - Is there a practical option not to use a system at all, considering the benefits?
  - Example: Safety matches [Leveson]
Different approaches

• Fail-safe
  – Safe in case of a fault
  – Skip any functionality
  – Example: stop robot immediately

• Fail operational
  – Stay operational in face of a fault
  – Example: An airplane cannot be shut down in the air

• Fail soft
  – Degraded operation
  – Example: Manual landing possible
Trends

• Trend 1: Software introduced into safety-critical systems.
  – Software enable so much new functionality it is tempting to use.
  – 1955 “only” 10% of weapon systems required sw, today (1986) 80% [Leveson]

• Trend 2: building systems where no manual intervention is feasible [Leveson]
Differences hw/sw

- Hardware relatively simple
- Hardware typically random errors
  - Software has built-in (design/implementation) errors, no manufacturing differences, does not wear out
- In hardware, “small errors have small consequences”
  - “No useful interpretation of tolerance is known for software” [Lutz]
- Hardware often has historical usage information
  - Control software does not
- Analog hardware infinite number of states, maths well understood [Parnas]
  - Software discrete, digital – large number of states, impossible (?) to test or analyze properly [Leveson] (1986)
Hardware

• Example: Therac radiation therapy machine.
  – Software fault (high-level radiation switched on erroneously)
  – No hardware interlock
  – Whose fault?
  – What could have been done?

• Safety is a system-level property!
A Software Engineer’s point of view

• The final system involves hardware and humans (who may make errors) and environment (e.g. birds)
  – How to design to take this into account?

• Software in itself cannot be unsafe! It is when it controls physical entities it can cause damage [Leveson]
  – Safety is a system-level property!
Options to choose

1. Intrinsically safe – incapable of causing hazard (generating sufficient energy or causing harmful exposures)
2. Prevent or minimize hazard
3. Control the hazard
4. Alarm the hazard

• These principles applicable to software
Humans

• Important part of many systems

• Modeling, testing etc. for technology -> Training, instructions, procedures for humans
Humans

• Keep humans in the loop
  – Often desirable to increase safety
  – Or at least make clear liability.
  – Example: War Games. Two people & two keys needed to launch missile

• However, a number of things to think of when involving humans:
Humans in the Loop

• Humans must have additional sources of information to make decisions
• Confidence needed in the automated system
  – Example: auto-landing system seldom used
• If warnings and shutdowns are too common, human attention is decreased
  – Operators may even do unauthorized modifications to alarm devices.
Humans in the Loop

• Danger to let computers take over routine tasks, humans are expected to monitor and intervene

• Loss of experience
  – Tendency to intervene too late
  – Example: steel plant in the Netherlands (productivity only)

• Tendency to relax from monitoring
  – Debate: air traffic control in US vs Europe

• Tendency to assume “the computer is always right”
  – Positive example: Crystal River nuclear plant
Levels of Safety

- $10^{-6}$ per hour called “ultra-reliable” [Parnas]
  - $10^6$ hours $= 114$ years
- $10^{-9}$, $10^{-7}$ failures per hour (or flight) [Leveson]
  - Mishap is not expected to happen during the lifetime of the whole fleet (of a certain airplane)

- Is it possible to guarantee something like this?
What Causes Mishaps?

• Almost always caused by multiple factors [Leveson]
  – Difficult to blame any single event or component of the system
  – Example: Three Mile Island
  – Example: problems in hardware may be corrected/controlled with software – ”vad var det vi sa, felet var mjukvarufel”.
What Causes Mishaps?

• Example: Ariane 5

![Diagram of Ariane 5 control system showing master and slave control computers and self-destruction mechanism.](image-url)
What Causes Mishaps

• Example: Chemical plant [Leveson]
  – Requirement: in case of fault, leave the controlled variables as they are and sound an alarm
  – Event: a catalyst had just been added, and the cooling-water flow was about to increase
  – Interpretation open: which “controlled variable”? Flow? Temperature?
  – Mishap: reactor overheated (constant flow implemented in sw)
What Causes Mishaps

• Example: Aircraft software
  – Designed to find fuel efficient altitude and speed
  – Flies into dangerous icing conditions
  – How to find during simulation?
Requirements

• Inadequate design foresight & specification errors are the greatest cause for sw related problems [Ericson, Griggs]
  – Testing can show consistency only within the requirements specified
Requirements

• Requirements build on assumptions
• Perhaps the most difficult
  – What assumptions do we make? (Example: huge undertaking analyzing US nuclear reactors, assumed e.g. the plants were built according to plan and properly operated, and independence of failure)
  – What if they are wrong? Extra measures taken? (Example: Titanic – no ship had had more than four of its water-tight compartments damaged up to that time.)
  – Intrinsically safe (definition earlier) – is it possible to be certain? (“safety matches”)
Design & Modeling

- Safety needs to be taken into account at design (not added afterwards)

- n-version programming
  - Errors in requirements will be implemented n times

- Safety systems
  - May in some cases decrease safety!

- Redundancy
  - May in some cases decrease safety!

- Formal methods
  - Tool errors, input errors
  - Addresses only part of the problem (e.g. timing)
  - Within requirements
Design & Modeling – Safety Systems

• Safety systems may themselves cause or contribute to mishaps, or even increase the possibilities of mishaps. [Leveson]
  – Example: three mile island (an indicator, lamp, failed → operators were unnoticed of a particular error)
  – Example: Ranger 6 (moon surveyor, a safety system –redundancy – short-circuit -> power loss for complete probe)

• “Redundancy is not always the correct design option to choose”
Design & Modeling – Safety Systems

• Emergency self-destruction
  – Example: French weather balloons
  – Example: ARIANE
  – Example: Torpedo (only anecdote)
Design & Modeling

• Two design principles
  – Minimize verification efforts
  – Features to increase safety must be carefully evaluated – do they increase complexity?
Testing, Simulation, V & V

• Testing certainly needed!

• Two types of verification [Leveson]:
  – Show that a fault cannot occur
  – Show that if a fault occur, it is not dangerous

• But…
Testing, Simulation, V & V

• Verification and testing can show consistency only within the requirements specified.
• Testing cannot show the absence of faults, only their presence [Dijkstra]
  – Enormous number of possible input and internal states…
• How to test response to catastrophic situations?
  – Real catastrophes are hard to come by!
  – Yes, we can simulate and provide a “controlled catastrophe”, and show that for a certain input which we define as a “nice catastrophe” we get the expected behavior…
  – Difficult to find “holes” in assumptions
  – … think an operator with a sledgehammer running amok
Documentation

• Purposes:
  – Specification
  – Review

• Goal: precise but understandable [Parnas]
  – Formal notations hard to understand.
  – Tables a cost-efficient strategy [Parnas]
  – Can be used to guide testing (selection and randomization)
Ethical Issues

• Large organizations developing
  – Whose ethics rule?
  – Who to blame?

• **Who should have done** something?
  – Who should have tested a certain case that occurred in practice (causing a disaster)?
  – Who should have foreseen a certain situation and designed for it (Example: Huygens space probe)
  – Responsibility – whose responsibility to assign a responsible (who could then have been blamed)…
  – Addressed at organizational level!
Ethical Issues

• Potential harm – made to whom? [Collins]
  – Users, Buyers, Public?

• Responsibilities of different stakeholders
  – Providers
  – Users
  – Buyers
  – Public

• Publicity test principle
  – Would we dare to openly present the tradeoffs being made?
Ethical Issues

• How to combine probability and severity?
  – How to compare an extremely unlikely but catastrophic event with a more likely but less serious event?
  – “More likely to die by getting hit by a meteor than in traffic”
  – The utilitarianism dilemma
Ethical Issues

• Conflict/trade-off safety vs. cost
• How to value human lives?
  – Example: Vägverket put a price tag on human life to calculate how to optimize traffic safety improvements
  – Deaths cheaper for society than invalids…
Ethical Issues

• Selfishness, ignorance of one’s own faults, greed
  – Importance of regulations, certifications
  – External auditing/testing/certification
  – What if inspectors have too high pressure – inspect a large number of items in too short time?

• Often some sort of requirements on internal reporting
  – What if an organization (or individuals) consciously fake results?
  – Or differ between items to be tested (high-quality) and others, to be sold (low quality)?
  – Or do not dare to report a bug found, considering the economic impact (e.g. stopping all airplanes)
Summary

• Safety is a system-level property!
• Every good thing has a drawback
  – Requirements
  – Testing
  – Safety systems & redundancy
• Only reasonable approach: combine all good practices known
  – Technical
  – Organizational & process