17
Critical Systems
and Software Safety

18-540 Distributed Embedded Systems
Philip Koopman
November 13, 2000

Required Reading: Ganssle, “Disaster”
Sections 3.3 & 3.5 from Storey, Safety-Critical Computer Systems
Assignments

◆ Next lecture read about human factors
  • The assigned reading only talks about risk-related issues; human interface issues are of course broader than that.

◆ Project part #5 due Wednesday 11/15

◆ Next homework is #8, due Friday 11/17
Where Are We Now?

- Where we’ve been:
  - Traditional Reliability/Fault Tolerance

- Where we’re going today:
  - Software Safety
  - System Safety

- Where we’re going next (lectures re-ordered for better coverage):
  - Human factors
  - Validation/Verification/Certification (and guest from ADtranz)
  - Design Methodologies
  - Ethics and miscellaneous topics
  - Internet toaster ovens, Bluetooth, and all that
Preview

◆ General safety engineering
  • Terminology
  • Basic Techniques (FMEA/FTA)

◆ Probabilities and consequences
  • Issues of probability of catastrophic failures

◆ What is software safety (really?)
Safety Engineering

- Largely based on industrial environments such as chemical plants

- Hazards based on uncontrolled release of energy
  - Risk was associated with amount of energy and time (e.g., explosion)
  - Risk was reduced via containment, minimizing potential energy in system, supervised operation in risky situations

- Embedded system engineering has to encompass
  - Release of energy from controlled system (physical damage)
  - Release of information from controlled system (security)
  - Avoiding inability to release energy/information (reliability/denial of service)
Terminology

- **Hazard:**
  - A situation with actual or potential danger to people, environment, or material

- **Incident (near miss):**
  - Something that under other circumstances would have been an accident

- **Accident (also called a mishap):**
  - Events that cause death, injury, environmental, or material damage

- **Risk:**
  - A combination of probability of hazards, and severity of likely outcomes.
  - Example: risk from lightning is
    - $5 \times 10^{-7}$ deaths per person-year.
    - $1.25 \times 10^{-6}$ injuries per person-year.
  - BART rail system has 250,000 year Mean Time to Hazard spec. per car
Basic Analysis Technique – FMEA

◦ Failure Mode and Effects Analysis (FMEA)
  • Probably the most commonly used technique in embedded system design
  • Looks for consequences of component failures (forward chaining technique)
  • Limitation: requires expert analysis to decide what to analyze

◦ Failure Mode and Effects Criticality Analysis (FMECA)
  • Similar to FMEA but with two columns added
    – Overall assessment of criticality
    – Possible actions to reduce criticality
FIGURE 14.9
<table>
<thead>
<tr>
<th>Item</th>
<th>Failure Modes</th>
<th>Cause of Failure</th>
<th>Possible Effects</th>
<th>Prob.</th>
<th>Level</th>
<th>Possible Action to Reduce Failure Rate or Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motor Case</td>
<td>Rupture</td>
<td>a. Poor workmanship</td>
<td>Destruction of missile</td>
<td>0.0006</td>
<td>Critical</td>
<td>Close control of manufacturing processes to ensure that workmanship meets prescribed standards. Rigid quality control of basic materials to eliminate defectives. Inspection and pressure testing of completed cases. Provision of suitable packaging to protect motor during transportation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b. Defective materials</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>c. Damage during transportation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>d. Damage during handling</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>e. Overpressurization</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FIGURE 14.10
A sample FMECA.
Fault Tree Analysis (FTA)

- Origins: 1961 to ensure no accidental Minuteman missile launches
- Analyzes possible causes of hazards, but you already have to have the list of hazards to begin with (backward chaining technique)
- Problems:
  - Doesn’t represent real-time issues
  - Doesn’t represent system state or operating modes
Portion of a fault tree for a patient monitoring system.

```
Sensor failure
\|--\--
|    |    
|    | Nurse fails to input item
|    | Sensor failure
|    | Too low frequency
\|--\--
|    | Human sets limits within required
|    | Too low measurement
\|--\--
|    | Vitals signs exceed
\|--\--
|    | Vitals signs exceed critical limits but not as exceeding limits erroneously reported
\|--\--
|    | Treatment administered
\|--\--
|    | Wrong or inadequate
```

etc.
Basic Analysis Technique – Event Tree

- **Start with a possible primary component failure**
  - For each downstream event, trace out possible combinations of component failures
  - Compute probability of failure at each stage and then sum overall probabilities

- **CAUTION!**
  - *Just because you guess a bunch of probabilities to make the tree doesn’t mean the answer is right, much less good to 4 significant digits!*
Figure 3.2  Probabilistic analysis applied to an event tree.
Safety vs. Reliability vs. Security

- **Reliability is composable; safety is emergent**
  - Reliability of a specific component can be improved (e.g., via redundancy)
    - Improving reliability reduces chance of system failure
  - A component can’t be “safe” – only a system can be “safe”
    - Component interactions can lead to unexpected emergent properties and hazards
    - Even a system operating according to specifications can still be unsafe (specification defects)

- **Safety is a bit more like security**
  - It only takes one security hole to make a system unsecure
  - It only takes one safety gap to make a system somewhat unsafe
  - The major difference is intent
    - Security holes tend to be exploited by people once they are known
    - Hazards tend to manifest if given enough operating hours
Techniques To Make Systems Safer

◆ Address risk explicitly; don’t bury the issue
  • No system is perfect; but you should at least try to address the issue

◆ Use a good process
  • A good process enables (but does not guarantee) a good product

◆ Design system at an appropriate level of reliability/availability/safety
  • Commonly, this means looking for single-point failures
  • Isolate safety-critical portions of system and apply more attention to them
  • Simplicity is a virtue
  • Plan for the unexpected (exceptions)

◆ Perform verification/validation/certification
  • Reviews/inspections
  • Testing (but, you can’t test long enough to ensure ultra-reliability)
  • Formal methods
  • Hazard analysis

◆ Include people as part of the system safety design
Lessons That Should Have Been Learned

- In carefully designed systems, most catastrophes are caused by multiple interacting events
  - BUT, if that doesn’t mean simple things won’t cause problems in other systems!

- Just because you can get away with it doesn’t make it safe
  - Challenger O-ring problem was in part based on an escalation of what they (thought they) could get away with
  - Just because a component worked last time doesn’t mean it is safe in the new application

- Humans are part of the system
  - Operators can help the system or make things worse
  - Often managers are too quick to blame humans for problems that are really due to over-complicated systems
  - Often the cause of a mishap is rooted in the operating environment/culture
Civil Aircraft Hazard Categories

- **Catastrophic**
  - Prevents continued safe flight and landing

- **Hazardous**
  - Large reduction in safety margins; perhaps fatal injuries to some passengers

- **Major**
  - Significant reduction in safety margins; perhaps non-fatal injuries to passengers

- **Minor**
  - Slight reduction in safety margins; reasonable workarounds

- **No effect**
  - Failure causes no effect

- **Note:**
  - Increase in crew workload is a significant factor to be considered
How Improbable Is Improbable?

- What if cars had safety-critical electronics – what failure rate is acceptable?
  - Assume that we’re talking about catastrophic failures, not fender benders

- Aircraft:
  - “Extremely improbable” is $10^{-9}$ per hour
  - From last lecture, $55 \times 10^6$ hours/year => once every 18 years in US fleet

- Cars:
  - $3 \times 10^{10}$ hours/year for US fleet
  - At $10^{-9}$ per hour, this gives 30 events per year in US fleet (545 times higher)
  - One possible approach to equalizing automotive risk:
    - Planes carry about 100 times more people than cars, so only need to improve risk by a factor of 5.45 for cars to achieve parity
    - Set “extremely improbable” criterion to $10^{-10}$ per hour for cars.
Embedded Distributed System Failures

- In addition to all the above, there can be network problems
  - Network failures can be attacked by using replicated networks
  - Network packet errors due to noise are a problem

- Be sure to calculate effects of dropped network packets!
  - Contributing causes of lost packets:
    - High bit error rate
    - Noise bursts due to electric motor operation
    - Retries not supported in order to simplify scheduling
    - Collision-based communication protocols
  - Event triggered systems – loss of packet can leave system in incorrect state
  - Time triggered systems – repeated loss of packet can cause loss of control loop stability
    - It doesn’t take many lost packets to lead to problems in a large scale fleet
Levels Of Criticality

- **First-order critical systems: directly mission critical**
  - Medical equipment (controlled by FDA)
  - Nuclear power (controlled by NRC)
  - Aviation (controlled by FAA)

- **National infrastructure: indirectly mission critical**
  - Utilities: Internet, power, phones, gas, oil distribution
  - What happens if you call 911 and the phone doesn’t work?
  - This area is receiving more attention due to info warfare issues

- **Critical to everyday life/business**
  - Desktop computing applications (e-mail, web, financial reports)
  - Coordination of traffic lights
  - Access to control computers in web-ified house of the future
  - Vehicles; mass transit operation

- **Some things just aren’t that critical – it’s OK if they fail once in a while**
Safety-Critical Failure Handling

- **Fail Operational**
  - Even though something fails the system keeps working
  - Usually accomplished through redundancy

- **Fail-over to reduced capability system**
  - Simpler algorithms
  - Mechanical backup
  - Person

- **Fail Safe**
  - Identify a safe state and transition to that safe state upon failure
    - Tension between safety and availability; a system with 0% availability might well be 100% safe
  - Sometimes use a reduced capability system as “limp home” to a safe state
What Is “Software Safety” (really?)

◆ Levson says:
  • “Software system safety implies that the software will execute within a system context without contributing to hazards”

◆ Manifestations of unsafe software
  • Incorrect software (software doesn’t meet requirements)
  • Software is correct, but requirements specify an unsafe behavior
  • Software is correct, but requirements fail to specify a behavior required for safety
  • Software exhibits additional behavior not precluded by specifications

◆ Another way of looking at software safety:
  • Software is where a lot of the complexity ends up
  • Complexity can lead to hazards
  • Software safety is really about taming complexity to achieve system safety
Software Safety Point/Counter-Point

- **Software safety researchers:**
  - We’re understanding how to create safer software
    - More recently, we’re understanding how to create safer specifications
  - Use a good process, use good judgment in not over-reaching, and things are likely to be OK

- **Counter-point:**
  - Good process facilitates good product, but does not ensure it
  - Society (and engineers) will always press for more complexity than we can safely implement
  - Complex systems tend to suffer from obscure, complex failures, perhaps leading to a situation of “normal accidents” being the expected situation

- **Human nature**
  - People are willing to take risks if they perceive sufficient benefit
  - The human perceived risk function is highly nonlinear and somewhat “irrational”
Review

◆ **General safety engineering**
  - Basic Techniques:
    - FMECA – failure mode and effect criticality analysis
    - FTA – fault tree analysis
    - Event tree (be cautious about believing actual numbers; better for relative comparisons)
  - See suggested reading for alternate methods beyond these three:
    - Storey, Safety-Critical Computer Systems, Chapter 3
    - Leveson, *Safeware*, Chapter 14

◆ **Use good design techniques to improve system safety**
  - Consider the entire system (including people)
  - Use good process, but count on defects still making it to the field
  - Weight failure modes by probability and consequences; spend limited resources to reduce overall risk

◆ **What is software safety?**
  - It’s about managing complexity to reduce system-level risk