Well-formed Dependency and Open-loop Safety

Based on Slides by Professor Lui Sha
Reminders and Announcements

- Announcement: CS 424 is now on Piazza:
  - piazza.com/illinois/fall2016/cs424
- Reminder: HW1 is out. Due this Thursday. See course webpage:
  - http://courses.engr.illinois.edu/cs424/
Recap

- Reliability for a given mission duration $t$, $R(t)$, is the probability of the system working as specified (i.e., probability of no failures) for a duration that is at least as long as $t$.

- The most commonly used reliability function is the exponential reliability function:

$$R(t) = e^{-\lambda t}$$

where $\lambda$ is the failure rate.
Triple Modular Redundancy

- Which case is TMR?

TMR has a lower reliability in the long term. How come?
Implications of the Postulates

\[ R(Effort, Complexity, t) = e^{-kC \frac{t}{E}} \]

- Note: splitting the effort greatly reduces reliability.
Analytic Redundancy and Complexity Reduction

- Partial redundancy via simple backup that meets only safety-critical requirements
Example: A Sorting Exercise

- **Sorting:**
  - Bubble sort: easy to write but slower, $O(n^2)$
  - Quick sort: faster, $O(n \log(n))$, but more complicated to write

- Joe remembers how to do bubble sort, but is not perfectly sure of quick sort (has a 50% chance of getting it right).

- Joe is asked to write a sorting routine:
  - Correct and fast: A
  - Correct but slow: B
  - Incorrect: F

Critical requirement: Must pass!
Solution

- Simplicity to “control” complexity

Joe will get at least a “B”.

input

O(n log(n))

quick-sort

O(n) if input is sorted

bubble-sort

O(n^2) otherwise

output
Simplex Architectural Pattern

A simple verifiable core; diversity in the form of 2 alternatives; feedback control of the software execution.

Data Flow Block Diagram

Better performance, but less reliable
Example

- Component with mean time to failure = 10 years. Compare the reliability of:
  a) Using this component alone
  b) TMR using three versions of this component
Example

- Component with mean time to failure = 10 years. Compare the reliability of:
  a) Using this component alone
  b) TMR using three versions of this component

After 1 year
Example

- Component with mean time to failure = 10 years. Compare the reliability of:
  a) Using this component alone
  b) TMR using three versions of this component

After 1 year

Answer:

a) \( r(t) = e^{-\lambda t} = e^{-(1/10).1} = 0.9048 \)
Example

- Component with mean time to failure = 10 years. Compare the reliability of:
  a) Using this component alone
  b) TMR using three versions of this component

After 1 year

Answer:

\[ r(t) = e^{-\lambda t} = e^{-(1/10) \cdot 1} = 0.9048 \]
\[ r(t)^3 + 3r(t)^2 (1 - r(t)) = 0.9745 \]
Example

Component with mean time to failure = 10 years. Compare the reliability of:

a) Using this component alone
b) TMR using three versions of this component

After 15 years
Example

Component with mean time to failure = 10 years. Compare the reliability of:

a) Using this component alone
b) TMR using three versions of this component

After 15 years

Answer:

a) \[ r(t) = e^{-\lambda t} = e^{-\left(\frac{1}{10}\right) \cdot 15} = 0.2231 \]
Example

Component with mean time to failure = 10 years. Compare the reliability of:
  a) Using this component alone
  b) TMR using three versions of this component

After 15 years

**Answer:**

a) \( r(t) = e^{-\lambda t} = e^{-(1/10) \cdot 15} = 0.2231 \)

b) \( r(t)^3 + 3r(t)^2 (1 - r(t)) = 0.1271 \)
Example

Component with mean time to failure = 10 years. Compare the reliability of:

a) Using this component alone
b) TMR using three versions of this component
c) Using this component with a reduced complexity backup ($C = 0.1$)

After 15 years
Example

- Component with mean time to failure = 10 years.
  Compare the reliability of:
    a) Using this component alone
    b) TMR using three versions of this component
    c) Using this component with a reduced complexity backup \((C = 0.1)\)

After 15 years

Answer:

c) \(r_1(t) = e^{-\lambda t} = 0.2231\), \(r_b(t) = e^{-0.1\lambda t} = 0.8607\)
Example

Component with mean time to failure = 10 years. Compare the reliability of:

a) Using this component alone
b) TMR using three versions of this component
c) Using this component with a reduced complexity backup \((C = 0.1)\)

After 15 years

Answer:

c) \(r_1(t) = e^{-\lambda t} = 0.2231\), \(r_b(t) = e^{-0.1\lambda t} = 0.8607\)

\(1 - (1 - r_1(t))(1 - r_b(t)) = 0.8918\)
Example

- Component with mean time to failure = 10 years (at unit complexity and unit budget). Compare the reliability of:
  a) Using this component alone
  b) TMR using three versions of this component assuming same total budget

After 1 year
Example

Component with mean time to failure = 10 years (at unit complexity and unit budget). Compare the reliability of:
   a) Using this component alone
   b) TMR using three versions of this component assuming same total budget

After 1 year

Answer:

a) \( r(t) = e^{-\lambda t} = e^{-(1/10).1} = 0.9048 \)
Example

- Component with mean time to failure = 10 years (at unit complexity and unit budget). Compare the reliability of:
  - a) Using this component alone
  - b) TMR using three versions of this component assuming same total budget

After 1 year

Answer:

a) \( r(t) = e^{-\lambda t} = e^{-(1/10) \cdot 1} = 0.9048 \)

b) \( r_2(t) = e^{-3\lambda t} = 0.7408 \)

\[
r_2(t)^3 + 3r_2(t)^2 (1 - r_2(t)) = 0.8333
\]
Lessons Learned?
Lessons Learned

- More components/redundancy is not always better
- When budget is finite, more components means “spreading thinner” → lower reliability
- Having a simple (i.e., low complexity) back-up significantly improves reliability!
Well Formed Dependencies

*Informal intuition:* A reliable component should not *depend* on a less reliable component (it defeats the purpose).
Well Formed Dependencies

- *Informal intuition:* A reliable component should not *depend* on a less reliable component (it defeats the purpose).

- Design guideline: **Use but do not depend** on less reliable components
Well Formed Dependencies

- Component A is said to depend on B, if the correctness of A’s service depends on B’s correctness.

- Component A is said to *use* the service of B, but not depend on it for its critical service S, if S can function correctly in spite of all B’s faults.

- A system’s dependency relations are said to be well-formed if and only if critical components may *use but do not depend* on the less critical components.
Design Philosophy

- Build the system out of a reliable core and less reliable components
- Ensure that the reliable core is *minimal* (must be simple to reduce complexity – see lessons learned from reliability examples)
- The reliable core can use but do not depend on other components (i.e., failures elsewhere should not affect reliable core)
- The reliable core should ensure safety or recover from failures of other components
Sorting Revisited

- How does the reliable component depend on the less reliable component? How to fix it?

Joe will get at least a “B”.

- $O(n \log(n))$
- $O(n)$ if input is sorted
- $O(n^2)$ otherwise
Sorting Revisited

- How does the reliable component depend on the less reliable component? How to fix it?

Joe will get at least a “B”.

```
input  |  O(n log(n))  | quick-sort  | O(n) if input is sorted  | bubble-sort  | output  
```

- Less Reliable component
- Reliable component

O(n^2) otherwise
Sorting Revisited
Ensuring Well-formed Dependencies

- Resource sharing faults
  - Memory accessing fault: address space isolation
  - Hogging the CPU: CPU cycle limit
  - Timing fault: time out.

- Semantic fault
  - Wrong order: Bubble sort
  - Corrupt the input data item list: Export only a permutation function on a protected input list
Safe State

- In cyber-physical systems it is important to keep the system from harm. The reliable core must ensure that the system remains in a safe state (keep the kid away from the freeway!!) even when other components fail.

- Example:
  - If your tire blows up, safely park the car on the shoulder of the road (safe state).
Discussion: Patient Controlled Analgesia

- When pain is severe in a post-surgery patient, the patient can push a button to get more pain medication (morphine: drug overdose will cause death). This is an example of a lethal device in the hands of an error-prone operator (the patient). How can we ensure safety of software controlled PCA?
Patient Controlled Analgesia

Component list:
- Infusion pump (with embedded micro-controller)
- Oxymeter (clipped on finger to measure blood oxygen level)
- ECG Reader (taped to patient’s chest)
- Network that connects them
- Inexperienced user

Design questions:
- Q1: What is the safety core? What’s a safe state?
- Q2: What components we can use but not depend on?
- Q3: What is the fault model for each component?
- Q4: How can the safety core withstand those faults?
Discussion: Avionics

- In avionics, the autopilot must be level-A certified.
- The autopilot receives trajectory input from a flight guidance system that is only level-C certified.
- Can the overall system be level-A certified? (Note: Assume that manual flight control is a safe state)
Avionics

Component list:
- Autopilot
- Flight guidance system
- Network that connects them
- Skilled pilot

Design questions:
- Q1: What is the safety core? What’s a safe state?
- Q2: What components we can use but not depend on?
- Q3: What is the fault model for each component?
- Q4: How can the safety core withstand those faults?
Case study:

- "A 32-year-old woman was having a laparoscopic cholecystectomy performed under general anesthesia. During that procedure and at the surgeon's request, a plain film x-ray was shot during a cholangiogram.
- The anesthesiologist stopped the ventilator for the x-ray. The x-ray technician was unable to remove the film because of its position beneath the table. The anesthesiologist attempted to help the technician, but found it difficult because the gears on the table had jammed.
- Finally, the x-ray was removed, and the surgical procedure recommenced.
- At some point, the anesthesiologist glanced at the EKG and noticed severe bradycardia. He realized he had never restarted the ventilator. This patient ultimately died"

APSF Newsletter, Winter 2005
Ventilator/X-Ray Interaction

- Architecture #1: Master controller on a Windows server orchestrates ventilator and X-ray machine actions over a network. Controllers tells machines to stop and re-start, and ensures that ventilator is not off too long. Comments?
Ventilator/X-Ray Interaction

- Architecture #1: Master controller on a Windows server orchestrates ventilator and X-ray machine actions over a network. Controllers tells machines to stop and re-start, and ensures that ventilator is not off too long. Comments?

![Diagram of ventilator and X-ray machine interaction]

Problem: Not well-formed dependencies. Correct operation depends on controller, network, Windows, server, etc.
**Ventilator/X-Ray Interaction**

- **Architecture #2**: Master controller on a Windows server tells ventilator when to pause. Ventilator has a hardware timer and restarts automatically once timer expires.
Ventilator/X-Ray Interaction

- Architecture #2: Master controller on a Windows server tells ventilator when to pause. Ventilator has a hardware timer and restarts automatically once timer expires.

Much better! Reliable core is just the hardware timer (in addition to the Ventilator). Safety does not depend on anything else.
Discussion: Asimov Laws of Robotics
Discussion: Asimov Laws of Robotics

- A robot may not injure a human being.
- A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.
- A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.