Complex Systems & Therac-25

From Week 1

Jorge Simosa

Systems and Complexity

- System = interacting **set of components** with a specified behavior at the **interface** with its environment

- Complexity Problems:
  - **Emergent properties** – show up only when the system is complete, "surprise"
  - **Propagation of effects** – small change → big effects
  - **Incommensurate scaling** – design for small model, broken for large model
  - **Trade-offs** – fixing one problem results in another problem

Complexity

Sources of Complexity

- Five Symptoms:
  - Large # of components
  - Large # of connections
  - Many irregularities: many exceptions
  - Long specifications
  - Size of design team
- Sources:
  - Excessive Generality
  - Patches – counteracting features
  - Performance vs. Simplicity tradeoff

Coping with Complexity

- Simplifying design principles
  - E.g. “Avoid excessive generality”
- Modularity
  - Split up system, consider separately
- Abstraction (e.g. RPC, Transactions)
  - Interfaces (hide the details)
  - Helps avoid propagation of effects
- Hierarchy (e.g. DNS)
- Layering (e.g. Internet)

Client/Server Model

- Modularity: messages to request services
- Fault tolerance: modules check messages for errors
- Secure: messages are only avenue for attackers

- On a single machine (soft modularity), there are problems such as fate sharing (e.g. callee crash -> caller crash)

- With “separate” machines for client and server (enforced modularity), we can use remote-procedure calls (RPC)
  - Need to account for unreliable network

Therac-25

- Three modes of operation: 1) Electron Therapy, 2) X-ray, 3) Field-light
  - Requires manual positioning and other parameters
- Hardware interlocks of Therac-20 were replaced with software interlocks.
- “Software interlock” = Boolean flag (who sets? who verifies?)
- Three parallel threads: 1) Keyboard input, 2) Turntable Position, 3) Beam setting & more

Therac-25 Pitfalls

- Key Take-away: Complex Systems Fail for Complex Reasons

- Faults:
  - Design of Operator Interface
    - E.g. useless error messages, no documentation, false alarms
  - Software
    - E.g. race conditions, overflows, lack of safety features
  - System design approach
    - E.g. lack of failure analysis, unaccounted failure modes, complex code
  - Design Iteration
    - E.g. previous failures, no follow-through on problems, “sweep it under the rug”
  - Other
    - E.g. independent checks, lack of software quality control

Complexity Sample

Spring 2012 – II. Complexity

6. Chapter 1 of the text describes several techniques for coping with complexity: Modularity (M), Abstraction (A), Layering (L), Hierarchy (H), Design for iteration (D), and Indirection (I).

A. Helps the designers incorporate feedback in system implementations.

Answer: Design for Iteration (D).

B. Helps simplify the task of debugging a complex system by letting implementers deal with smaller components

Answer: Modularity (M).

C. Makes it easier for designers to take advantage of delayed binding in system implementations.

Answer: Indirection (I).

D. Ensures that the implementation will obey the robustness principle.

Answer: Abstraction (A).

E. If this is done correctly, it can help reduce the number of inter-module interactions in large systems.

Answer: Hierarchy (H).

Therac-25 Sample

Spring 2010 – I. Reading Questions

3. Based on the description of the Therac-25 in the paper by Leveson and Turner which of the following statements are true?

A. The hardware interlocks present in the Therac-20 were also present in the Therac-25.

Answer: False

B. A detailed fault tree analysis of the Therac-25 estimated the probability of the wrong mode being selected to be 4×10^{-9}.

Answer: False

C. The Therac-25 software acquired locks in the wrong order, leading to disastrous consequences.

Answer: False

Client/Server Sample

Spring 2011 – II. Client/Server

4. Which of these statements about client/server organization are true or false?

A. In a client-server system, the client and the server always run on different machines.

Answer: False

B. A client-server system can be faster than a system using the same code to do the actual work, but in which everything runs in the same process, in spite of the added cost of communication between the client and the server.

Answer: True

C. In a client-server system that uses remote procedure call, the only important difference from a similar system that uses local procedure call is that the remote calls may be substantially slower.

Answer: False

D. A server that wants to provide at-least-once semantics to a client must store a table of previous responses.

Answer: False

More Samples

- See:
  - ‘12 Q1 – II. Complexity
  - ‘11 Q1 – II. Client/Server
  - ‘10 Q1 – I. Reading Questions #3
  - ‘09 Q1 – III. BeanBag.com
  - ‘08 Q1 – I. Reading Questions #4
  - ‘07 Q1 – II. RPC Semantics
  - ‘06 Q1 – Ben’s OS (RPC)
  - ‘05 Q1 – I. Complex Systems #1, #2
  - ‘04 Q1 – Local RPC
  - ‘02 Q1 – I. Reading Questions #1