Hazard Analysis for Software Safety

Technology Overview

17-939A
Jennifer Morris
3/15/2005

Overview

- Background on Safety and Hazard Analysis
- SFTA
  - Analyzing Software Safety (SFTA), [Leveson & Harvey 83]
  - SFTA for Product Lines [Dehlinger & Lutz 04]
- Time Petri Nets
- HAZOP
- SFMEA/SFMECA
- STAMP

Safety

- Safety: “freedom from accidents or losses” [Leveson95]
- Accident: “undesired and unplanned (but not necessarily unexpected) event that results in (at least) a specified level of loss” [Leveson95]
- Incident/near miss: “an event that involves no loss (or only minor loss) but with the potential for loss under different circumstances” [Leveson95]

Hazard Analysis

- Hazard: “a state or set of conditions of a system (or an object) that, together with other conditions in the environment of the system (or object) will lead inevitably to an accident (loss event)” [Leveson95]
- Risk: “a combination of the frequency or probability of a specified hazardous event, and its consequence” [Storey96]
- Purpose of hazard analysis:
  - identify potential hazards
  - reduce risk
### Software Fault-Tree Analysis

- Backward search from hazards to root causes
  - Qualitative (hardware FTAs also have quantitative reliability measures)
  - Code-level analysis, requirements
  - Tree of events connected by logical AND & OR gates
    - Leaf nodes indicate basic fault event or primary failure mode
    - Identify vulnerabilities in design or requirements
- Useful tool for analysts to visualize system hazards
- Limited by list of identified hazards, expertise of analyst
- Does not handle timing faults well
- Does not scale well

### Historical context: Major Software-Related Accidents

- **Therac 25** (1985-87)
- **Patriot Missile** (1991)
- **Arianne 5** (1996)
- **Mars Climate Orbiter** (1998)

### Leveson & Harvey 83

**Software Fault-Tree Analysis (SFTA)**

- **ET**: enhanced tool/method
  - The effectiveness of fault-tree analysis in supporting the design of safety-critical systems has been demonstrated. An enhanced tool/method is described for the design of safety-critical system software based on software fault-tree analysis of code-level logic. Examples are provided confirming the effectiveness of its support for safety-critical system software in design.
- **Question**: Method of analysis: How can we evaluate safety hazards in software?
- **Result**: SFTA: trace hazardous outputs to their causes through logic in software designs
- **Validation**: Example demonstrating SFTA on flight & telemetry software of UC Berkeley spacecraft FIREWHEEL (NASA/ESA)
  - 1250 lines of Intel 8080 assembly code
- Seminal paper introducing the research area of software safety
- First real attempt at adapting hazard analysis to software
Fault-Tree Format for if-then-else
[Leveson & Harvey 83]

- The effectiveness of Software fault-tree analysis in supporting the design of safety-critical systems has been demonstrated. An enhanced tool/method is described for the design of safety-critical systems based on product-line software fault-tree analysis (SFTA). Examples are provided confirming the effectiveness of its support for safety-critical system software product lines.

- Question: Method of analysis: How can we evaluate safety hazards in software product lines more efficiently?

- Validation: two small examples demonstrating product-line SFTA of a floating weather station and a pacemaker

Example: Pacemaker Product-line SFTA
[Dehlinger & Lutz 04]
**Time Petri Nets**

- Graph of software states & transitions
  - Qualitative
  - Places indicate conditions (state), transitions indicate events
  - Tokens used to restrict transitions
  - Reachability graphs
  - Identify paths to hazardous states and mitigate with interlocks
- Allows incorporation of timing into analysis
- Difficult to analyze
- Does not scale well

---

**Example: Computer-Controlled Train Warning Signal** [Leveson & Stolzy 87]

![Petri Net Diagram]

**Example: Next State**

[Leveson & Stolzy 87]

![Next State Diagram]

**Example: Petri Net Reachability Graph**

[H = Hazardous States]

![Reachability Graph Diagram]
Example: Computer-Controlled Train Warning Signal with Interlock [Leveson & Stolzy 87]

Software Hazard and Operability Study (HAZOP) [McDermid & Pumfrey 94]

- Structured “brainstorming” of hazards
  - Adapted from Chemical Industries Association
  - Table of behaviors, causes, and consequences
  - (Data) flow-based analysis & guide-words
  - Identify unexpected (hazardous) behavior & consequences
- Simple technique
- Limited by expertise/judgement of analyst
- Time/labor intensive

Example: HAZOP Guidewords [McDermid & Pumfrey 94]

- Value
- Timing
- Service Provision
- Failure Categorization

Example: HAZOP Analysis [McDermid & Pumfrey 94]

- Flow
- Protocol
- Data Type
- Omission
- Commission
- Early
- Late
- Subtitle
- Value
- Course

Example: HAZOP Analysis

- Guide Word
- Deviation
- Causes
- Co-effectors
- Effects
- DUT
- Justification / Design Proposals
Software Failure Modes & Effects Analysis

- Forward search from faults/events to hazards
  - Requirements
  - Table of faults and the hazards they cause
    - data (missing, incorrect, untimely, extra, etc.)
    - events (missing, incorrect, untimely, termination, etc.)
  - SFMCEA includes criticality of hazards
- Limited by list of faults, expertise of analyst
- Fairly easy to come up with failure modes of components
- Does not handle multiple/combined faults well
- Does not scale well

Example: FMEA
[Latz & Woodhouse 94]

<table>
<thead>
<tr>
<th>Event</th>
<th>Event fault type</th>
<th>Description</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculate pointer into a table</td>
<td>Incorrect logic</td>
<td>Pointer is miscalculated</td>
<td>Prevents past end of table preventing needed reconfiguration</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data item</th>
<th>Data fault type</th>
<th>Description</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical mode flag</td>
<td>Incorrect value</td>
<td>Flag set to true during non-critical mode</td>
<td>Unnecessary reconfiguration commanded</td>
</tr>
</tbody>
</table>

Systems-Theoretic Accident Model and Process (STAMP)

- Backward search from hazards to safety constraint violations & control flaws
  - Requirements
  - Management
- Hazards are caused when safety constraints are violated
  - inadequate control actions
  - inadequate execution of control actions
  - inadequate or missing feedback
- Addresses systemic properties of safety
- Incorrect, inadequate constraints?
- Scalability?

Bibliography

- N. G. Leveson, Software System Safety and Computers, Addison-Wesley, 1995