Software System Safety

Nancy G. Leveson

MIT Aero/Astro Dept. (leveson@mit.edu)
http://sunnyday.mit.edu

© Copyright by the author, July 2002. All rights reserved. Copying without fee is permitted provided that the copies are not made or distributed for direct commercial advantage and provided that credit to the source is given. Abstracting with credit is permitted.
Types of Accidents

- Component Failure Accidents
  Single or multiple component failures
  Usually assume random failure

- System Accidents
  Arise in interactions among components
  No components may have "failed"
  Caused by interactive complexity and tight coupling
  Exacerbated by the introduction of computers.
Interactive Complexity

- Complexity is a moving target
- The underlying factor is intellectual manageability

1. A "simple" system has a small number of unknowns in its interactions within the system and with its environment.

2. A system is intellectually unmanageable when the level of interactions reaches the point where they cannot be thoroughly
   - planned
   - understood
   - anticipated
   - guarded against

3. Introducing new technology introduces unknowns and even "unk-unks."

Computers and Risk

*We seem not to trust one another as much as would be desirable. In lieu of trusting each other, are we putting too much trust in our technology? . . . Perhaps we are not educating our children sufficiently well to understand the reasonable uses and limits of technology.*

Thomas B. Sheridan
A Possible Solution

- Enforce discipline and control complexity
  - Limits have changed from structural integrity and physical constraints of materials to intellectual limits

- Improve communication among engineers

- Build safety in by enforcing constraints on behavior

Example (batch reactor)

**System safety constraint:**
Water must be flowing into reflux condenser whenever catalyst is added to reactor.

**Software safety constraint:**
Software must always open water valve before catalyst valve

Stages in Process Control System Evolution

1. Mechanical systems
   - Direct sensory perception of process
   - Displays are directly connected to process and thus are physical extensions of it.
   - Design decisions highly constrained by:
     - Available space
     - Physics of underlying process
     - Limited possibility of action at a distance
Stages in Process Control System Evolution (2)

2. Electromechanical systems
   - Capability for action at a distance
   - Need to provide an image of process to operators
   - Need to provide feedback on actions taken.
   - Relaxed constraints on designers but created new possibilities for designer and operator error.

Stages in Process Control System Evolution (3)

3. Computer–based systems
   - Allow multiplexing of controls and displays.
   - Relaxes even more constraints and introduces more possibility for error.
   - But constraints shaped environment in ways that efficiently transmitted valuable process information and supported cognitive processes of operators.
   - Finding it hard to capture and present these qualities in new systems.
The Problem to be Solved

- The primary safety problem in computer-based systems is the lack of appropriate constraints on design.

- The job of the system safety engineer is to identify the design constraints necessary to maintain safety and to ensure the system and software design enforces them.
Safety $\neq$ Reliability

Accidents in high–tech systems are changing their nature, and we must change our approaches to safety accordingly.

Confusing Safety and Reliability

From an FAA report on ATC software architectures:

"The FAA’s en route automation meets the criteria for consideration as a safety–critical system. Therefore, en route automation systems must possess ultra–high reliability."

From a blue ribbon panel report on the V–22 Osprey problems:

"Safety [software]: ...

Recommendation: Improve reliability, then verify by extensive test/fix/test in challenging environments."
Does Software Fail?

**Failure:** Nonperformance or inability of system or component to perform its intended function for a specified time under specified environmental conditions.

A basic abnormal occurrence, e.g.,
- burned out bearing in a pump
- relay not closing properly when voltage applied

**Fault:** Higher-order events, e.g.,
- relay closes at wrong time due to improper functioning of an upstream component.

All failures are faults but not all faults are failures.

Reliability Engineering Approach to Safety

**Reliability:** The probability an item will perform its required function in the specified manner over a given time period and under specified or assumed conditions.

*(Note: Most software-related accidents result from errors in specified requirements or function and deviations from assumed conditions.)*

- Concerned primarily with failures and failure rate reduction
  - Parallel redundancy
  - Standby sparing
  - Safety factors and margins
  - Derating
  - Screening
  - Timed replacements
Reliability Engineering Approach to Safety (2)

- Assumes accidents are the result of component failure.
  - Techniques exist to increase component reliability
    Failure rates in hardware are quantifiable.
  - Omits important factors in accidents.
    May even decrease safety.

- Many accidents occur without any component “failure”
  - e.g. Accidents may be caused by equipment operation
    outside parameters and time limits upon which
    reliability analyses are based.
  - Or may be caused by interactions of components
    all operating according to specification

Highly reliable components are not necessarily safe.
Software Component Reuse

• One of most common factors in software–related accidents

• Software contains assumptions about its environment.
  Accidents occur when these assumptions are incorrect.
  
  − Therac−25
  − Ariane 5
  − U.K. ATC software

• Most likely to change the features embedded in or controlled by the software.

• COTS makes safety analysis more difficult.

*Safety and reliability are different qualities!*

Software–Related Accidents

• Are usually caused by flawed requirements

  − Incomplete or wrong assumptions about operation of controlled system or required operation of computer.

  − Unhandled controlled–system states and environmental conditions.

• Merely trying to get the software “correct” or to make it reliable will not make it safer under these conditions.
Software–Related Accidents (con’t.)

- Software may be highly reliable and “correct” and still be unsafe.
  - Correctly implements requirements but specified behavior unsafe from a system perspective.
  - Requirements do not specify some particular behavior required for system safety (incomplete)
  - Software has unintended (and unsafe) behavior beyond what is specified in requirements.

System Safety

- A planned, disciplined, and systematic approach to preventing or reducing accidents throughout the life cycle of a system.

  - “Organized common sense ” (Mueller, 1968)
  - Primary concern is the management of hazards:
    - Hazard
      - identification
      - evaluation
      - elimination
      - control
    - through
      - analysis
      - design
      - management
  - MIL–STD–882
System Safety (2)

- Hazard analysis and control is a continuous, iterative process throughout system development and use.

<table>
<thead>
<tr>
<th>Conceptual development</th>
<th>Design</th>
<th>Development</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hazard identification</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hazard resolution</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verification</td>
<td></td>
<td></td>
<td>Change analysis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Operational feedback</td>
</tr>
</tbody>
</table>

- Hazard resolution precedence:
  1. Eliminate the hazard
  2. Prevent or minimize the occurrence of the hazard
  3. Control the hazard if it occurs.
  4. Minimize damage.

- Management

---

Process Steps

1. Perform a Preliminary Hazard Analysis
   Produces hazard list

2. Perform a System Hazard Analysis (not just Failure Analysis)
   Identifies potential causes of hazards

3. Identify appropriate design constraints on system, software, and humans.

4. Design at system level to eliminate or control hazards.

5. Trace unresolved hazards and system hazard controls to software requirements.
Specifying Safety Constraints

• Most software requirements only specify nominal behavior
  Need to specify off–nominal behavior
  Need to specify what software must NOT do

• What must not do is not inverse of what must do

• Derive from system hazard analysis

Process Steps (2)

6. Software requirements review and analysis
   Completeness
   Simulation and animation
   Software hazard analysis
   Robustness (environment) analysis
   Mode confusion and other human error analyses
   Human factors analyses (usability, workload, etc.)
Process Steps (3)

7. Implementation with safety in mind
   - Defensive programming
   - Assertions and run-time checking
   - Separation of critical functions
   - Elimination of unnecessary functions
   - Exception-handling etc.

8. Off-nominal and safety testing

Process Steps (4)

9. Operational Analysis and Auditing
   - Change analysis
   - Incident and accident analysis
   - Performance monitoring
   - Periodic audits
A Human–Centered, Safety–Driven Design Process

**Human Factors**

- **Preliminary Task Analysis**
  - Operator Goals and Responsibilities
  - Task Allocation Principles
  - Operator Task and Training Requirements

- **Operator Task Analysis**
  - Simulation/Experiments
  - Usability Analysis
  - Other Human Factors Evaluation (workload, situation awareness, etc.)

- **Operational Analysis**
  - Performance Monitoring
  - Periodic audits
  - Change Analysis

**System Engineering**

- Identify system goals and environmental assumptions
- Generate system and operational requirements and design constraints
- Allocate tasks and generate system design (including HMI)
- Model and evaluate operator tasks and component blackbox behavior (system design)
- Design and construct components, controls and displays, training materials, and operator manuals
- Verification
- Field testing, installation, and training

**System Safety**

- **Preliminary Hazard Analysis**
  - Hazard List
  - Fault Tree Analysis
  - Safety Requirements and Constraints

- **System Hazard Analysis**
  - Completeness/Consistency Analysis
  - Simulation and Animation
  - State Machine Hazard Analysis
  - Deviation Analysis (FMECA)
  - Mode Confusion Analysis
  - Human Error Analysis
  - Timing and other analyses

- **Safety Verification**
  - Safety Testing
  - Software FTA

- **Operational Analysis**
  - Change Analysis
  - Incident and accident analysis
  - Periodic audits
  - Performance Monitoring

**A Human−Centered, Safety−Driven Design Process**
Level 1: System Purpose

- High-Level Requirements
  [1.2] TCAS shall provide collision avoidance protection for any two aircraft closing horizontally at any rate up to 1200 knots and vertically up to 10,000 feet per minute.
  Assumption: Commercial aircraft can operate up to 600 knots and 5000 fpm during vertical climb or controlled descent (and therefore the planes can close horizontally up to 1200 knots and vertically up to 10,000 fpm).

- Design and Safety Constraints
  [SC5] The system must not disrupt the pilot and ATC operations during critical phases of flight nor disrupt aircraft operation.
  [SC5.1] The pilot of a TCAS-equipped aircraft must have the option to switch to the Traffic-Advisory-Only mode where TAs are displayed but display of resolution advisories is prohibited.
  Assumption: This feature will be used during final approach to parallel runways when two aircraft are projected to come close to each other and TCAS would call for an evasive maneuver.

Example Level 1 Safety Constraints for TCAS

SC–7 TCAS must not create near misses (result in a hazardous level of vertical separation) that would not have occurred had the aircraft not carried TCAS.

SC–7.1 Crossing maneuvers must be avoided if possible.
  ↓ 2.36, 2.38, 2.48, 2.49.2

SC–7.2 The reversal of a displayed advisory must be extremely rare.
  ↓ 2.51, 2.56.3, 2.65.3, 2.66

SC–7.3 TCAS must not reverse an advisory if the pilot will have insufficient time to respond to the RA before the closest point of approach (four seconds or less) or if own and intruder aircraft are separated by less than 200 feet vertically when 10 seconds or less remain to closest point of approach.
  ↓ 2.52
Level 1: System Purpose (3)

- System Limitations
  
  L.5 TCAS provides no protection against aircraft with nonoperational or non–Mode C transponders.

- Operator Requirements
  
  OP. 4 After the threat is resolved the pilot shall return promptly and smoothly to his/her previously assigned flight path.

- Human–Interface Requirements

- Hazard and other System Analyses

Hazard List for TCAS

H1: Near midair collision (NMAC): An encounter for which, at the closest point of approach, the vertical separation is less than 100 feet and the horizontal separation is less than 500 feet.

H2: TCAS causes controlled maneuver into ground  
e.g. descend command near terrain

H3: TCAS causes pilot to lose control of the aircraft.

H4: TCAS interferes with other safety–related systems  
e.g. interferes with ground proximity warning
**TCAS does not display a resolution advisory**

TCAS unit is not providing RAs.
- Self-monitor shuts down TCAS unit
- Sensitivity level set such that no RAs are displayed.
- No RA inputs are provided to the display.
- No RA is generated by the logic
  - Inputs do not satisfy RA criteria
    - Surveillance puts threat outside corrective RA position.
      - Surveillance does not pass adequate track to the logic
        - Threat is non-Mode C aircraft
        - Surveillance failure
      - Altitude errors put threat in non-threat position.
        - Uneven terrain
        - Intruder altitude error
        - Own Mode C altitude error
        - Own radar altimeter error
      - Altitude errors put threat on ground
        - Uneven terrain
        - Intruder maneuver causes logic to delay RA beyond CPA
    - Process/display connectors fail
      - Display is preempted by other functions
      - Display hardware fails
  - TCAS displays a resolution advisory that the pilot does not follow.

Pilot does not execute RA at all.
- Crew does not perceive RA alarm.
  - Inadequate alarm design
  - Crew is preoccupied
  - Crew does not believe RA is correct.
  - Pilot executes the RA but inadequately
    - Pilot stops before RA is removed
    - Pilot continues beyond point RA is removed
    - Pilot delays execution beyond time allowed
2.19 When below 1700 feet AGL, the CAS logic uses the difference between its own aircraft pressure altitude and radar altitude to determine the approximate elevation of the ground above sea level (see Figure 2.5). It then subtracts the latter value from the pressure altitude value received from the target to determine the approximate altitude of the target above the ground (barometric altitude – radar altitude + 180 feet). If this altitude is less than 180 feet, TCAS considers the target to be on the ground (§1.SC4.9). Traffic and resolution advisories are inhibited for any intruder whose tracked altitude is below this estimate. Hysteresis is provided to reduce vacillations in the display of traffic advisories that might result from hilly terrain (§ FTA–320). All RAs are inhibited when own TCAS is within 500 feet of the ground.

<table>
<thead>
<tr>
<th>OWN TCAS</th>
<th>Radar Altimeter Value</th>
<th>Declared Altimeter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barometric Altimeter</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

### Figure 2.5

- **OWN TCAS**
- **Radar Altimeter Value**
- **Declared Altimeter**
- **Barometric Altimeter**
- **180-foot Allowance**
- **Declared on Ground**

---
Example Level–2 System Design for TCAS

SENSE REVERSALS

2.51 In most encounter situations, the resolution advisory sense will be maintained for the duration of an encounter with a threat aircraft. However, under certain circumstances, it may be necessary for that sense to be reversed. For example, a conflict between two TCAS–equipped aircraft will, with very high probability, result in selection of complementary advisory senses because of the coordination protocol between the two aircraft. However, if coordination communications between the two aircraft are disrupted at a critical time of sense selection, both aircraft may choose their advisories independently.

This could possibly result in selection of incompatible senses.

2.51.1 [Information about how incompatibilities are handled]

Level 3 Modeling Language Example

Description: A threat is reclassified as other traffic if its altitude reporting has been lost (§2.13) and either the bearing or range inputs are invalid; if its altitude reporting has been lost and both the range and bearing are valid but neither the proximate nor potential threat classification criteria are satisfied; or the aircraft is on the ground (§2.12).

Mapping to Level 2: §2.23, §2.29
Mapping to Level 4: §4.7.1, Traffic–Advisory
Preliminary Hazard Analysis

1. Identify system hazards

2. Translate system hazards into high-level system safety design constraints.

3. Assess hazards if required to do so.

4. Establish the hazard log.

System Hazards for Automated Train Doors

- Train starts with door open.
- Door opens while train is in motion.
- Door opens while improperly aligned with station platform.
- Door closes while someone is in doorway.
- Door that closes on an obstruction does not reopen or reopened door does not reclose.
- Doors cannot be opened for emergency evacuation.
System Hazards for Air Traffic Control

- Controlled aircraft violate minimum separation standards (NMAC).
- Airborne controlled aircraft enters an unsafe atmospheric region.
- Controlled airborne aircraft enters restricted airspace without authorization.
- Controlled airborne aircraft gets too close to a fixed obstacle other than a safe point of touchdown on assigned runway (CFIT).
- Controlled airborne aircraft and an intruder in controlled airspace violate minimum separation.
- Controlled aircraft operates outside its performance envelope.
- Aircraft on ground comes too close to moving objects or collides with stationary objects or leaves the paved area.
- Aircraft enters a runway for which it does not have clearance.
- Controlled aircraft executes an extreme maneuver within its performance envelope.
- Loss of aircraft control.

Exercise: Identify the system hazards for this cruise-control system

The cruise control system operates only when the engine is running. When the driver turns the system on, the speed at which the car is traveling at that instant is maintained. The system monitors the car’s speed by sensing the rate at which the wheels are turning, and it maintains desired speed by controlling the throttle position. After the system has been turned on, the driver may tell it to start increasing speed, wait a period of time, and then tell it to stop increasing speed. Throughout the time period, the system will increase the speed at a fixed rate, and then will maintain the final speed reached.

The driver may turn off the system at any time. The system will turn off if it senses that the accelerator has been depressed far enough to override the throttle control. If the system is on and senses that the brake has been depressed, it will cease maintaining speed but will not turn off. The driver may tell the system to resume speed, whereupon it will return to the speed it was maintaining before braking and resume maintenance of that speed.
Hazards must be translated into design constraints.

<table>
<thead>
<tr>
<th>HAZARD</th>
<th>DESIGN CRITERION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train starts with door open.</td>
<td>Train must not be capable of moving with any door open.</td>
</tr>
<tr>
<td>Door opens while train is in motion.</td>
<td>Doors must remain closed while train is in motion.</td>
</tr>
<tr>
<td>Door opens while improperly aligned with station platform.</td>
<td>Door must be capable of opening only after train is stopped and properly aligned with platform unless emergency exists (see below).</td>
</tr>
<tr>
<td>Door closes while someone is in doorway.</td>
<td>Door areas must be clear before door closing begins.</td>
</tr>
<tr>
<td>Door that closes on an obstruction does not reopen or reopened door does not reclose.</td>
<td>An obstructed door must reopen to permit removal of obstruction and then automatically reclose.</td>
</tr>
<tr>
<td>Doors cannot be opened for emergency evacuation.</td>
<td>Means must be provided to open doors anywhere when the train is stopped for emergency evacuation.</td>
</tr>
</tbody>
</table>

Example PHA for ATC Approach Control

<table>
<thead>
<tr>
<th>HAZARDS</th>
<th>REQUIREMENTS/CONSTRAINTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. A pair of controlled aircraft violate minimum separation standards.</td>
<td>1a. ATC shall provide advisories that maintain safe separation between aircraft.</td>
</tr>
<tr>
<td></td>
<td>1b. ATC shall provide conflict alerts.</td>
</tr>
<tr>
<td>2. A controlled aircraft enters an unsafe atmospheric region.</td>
<td>2a. ATC must not issue advisories that direct aircraft into areas with unsafe atmospheric conditions.</td>
</tr>
<tr>
<td>(icing conditions, windshear areas, thunderstorm cells)</td>
<td>2b. ATC shall provide weather advisories and alerts to flight crews.</td>
</tr>
<tr>
<td></td>
<td>2c. ATC shall warn aircraft that enter an unsafe atmospheric region.</td>
</tr>
</tbody>
</table>
### Example PHA for ATC Approach Control (2)

<table>
<thead>
<tr>
<th>HAZARDS</th>
<th>REQUIREMENTS/CONSTRAINTS</th>
</tr>
</thead>
</table>
| 3. A controlled aircraft enters restricted airspace without authorization. | 3a. ATC must not issue advisories that direct an aircraft into restricted airspace unless avoiding a greater hazard.  
3b. ATC shall provide timely warnings to aircraft to prevent their incursion into restricted airspace. |
| 4. A controlled aircraft gets too close to a fixed obstacle or terrain other than a safe point of touchdown on assigned runway. | 4. ATC shall provide advisories that maintain safe separation between aircraft and terrain or physical obstacles. |
| 5. A controlled aircraft and an intruder in controlled airspace violate minimum separation standards. | 5. ATC shall provide alerts and advisories to avoid intruders if at all possible. |

### HAZARDS | REQUIREMENTS/CONSTRAINTS
|----------|--------------------------|
| 6. Loss of controlled flight or loss of airframe integrity. | 6a. ATC must not issue advisories outside the safe performance envelope of the aircraft.  
6b. ATC advisories must not distract or disrupt the crew from maintaining safety of flight.  
6c. ATC must not issue advisories that the pilot or aircraft cannot fly or that degrade the continued safe flight of the aircraft.  
6d. ATC must not provide advisories that cause an aircraft to fall below the standard glidepath or intersect it at the wrong place. |
Requirements Validation

• Requirements are source of most operational errors and almost all the software contributions to accidents.

• Much of software hazard analysis effort therefore should focus on requirements.

• Problem is dealing with complexity
  
  1) Use blackbox models to separate external behavior from complexity of internal design to accomplish the behavior.

  2) Use abstraction and metamodels to handle large number of discrete states required to describe software behavior.

    Do not have continuous math to assist us

    But new types of state machine modeling languages drastically reduce number of states and transitions modeler needs to describe.

Requirements Analysis

• Model Execution, Animation, and Visualization

• Completeness

• State Machine Hazard Analysis (backwards reachability)

• Software Deviation Analysis

• Human Error Analysis

• Test Coverage Analysis and Test Case Generation

Automatic code generation?
Requirements Completeness

- Most software–related accidents involve software requirements deficiencies.
- Accidents often result from unhandled and unspecified cases.
- We have defined a set of criteria to determine whether a requirements specification is complete.
- Derived from accidents and basic engineering principles.
- Validated (at JPL) and used on industrial projects.

Completeness: Requirements are sufficient to distinguish the desired behavior of the software from that of any other undesired program that might be designed.

Requirements Completeness Criteria (2)

- How were criteria derived?
  - Mapped the parts of a control loop to a state machine

  ![State Machine Diagram]

  - Defined completeness for each part of state machine
    States, inputs, outputs, transitions
    Mathematical completeness
  - Added basic engineering principles (e.g., feedback)
  - Added what have learned from accidents
Requirements Completeness Criteria (3)

About 60 criteria in all including human–computer interaction.

(won’t go through them all— they are in the book)

- Startup, shutdown
- Mode transitions
- Inputs and outputs
- Value and timing
- Load and capacity
- Environment capacity
- Failure states and transitions
- Human–computer interface

Robustness
Data age
Latency
Feedback
Reversibility
Preemption
Path Robustness

Most integrated into SpecTRM–RL language design or simple tools can check them.
Design for Safety

- Software design must enforce safety constraints
- Should be able to trace from requirements to code (vice versa)
- Design should incorporate basic safety design principles

Safe Design Precedence

<table>
<thead>
<tr>
<th>HAZARD ELIMINATION</th>
<th>HAZARD REDUCTION</th>
<th>HAZARD CONTROL</th>
<th>DAMAGE REDUCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Substitution</td>
<td>Design for controllability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simplification</td>
<td>Barriers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decoupling</td>
<td>Lockins, Lockouts, Interlocks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elimination of human errors</td>
<td>Failure Minimization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reduction of hazardous materials or conditions</td>
<td>Safety Factors and Margins</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Redundancy</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Decreasing cost
Increasing effectiveness