Case Studies in Software and Computing System Safety: Accidents and Lessons Learned

Terry Hardy

NASA GSFC Systems Engineering Seminar

August 2, 2011

Introduction

- Background and Definitions
- Case Studies: System Safety Process
- Case Studies: Software Engineering and Support Processes
- Overarching Lessons Learned
- The Way Forward
- Final Thoughts

Introduction

- For simple systems and operations, risk tradeoffs are easy.
- For complex systems the risk tradeoffs are harder: airplanes, automobiles, nuclear power plants, rockets.
- Safety is not obvious in complex systems, in part because we cannot rely on experience.
- Understanding the risks in complex systems requires a systematic approach: system safety.

System Safety Defined

System Safety: the application of special technical and managerial skills to the systematic, forward-looking identification and control of hazards throughout the life cycle of a project, program, or activity.

- System safety is both a management doctrine and an engineering discipline
- System safety is forward-looking, not just experienced-based
- System safety seeks to design-in safety and assure that safety is considered throughout development
- System safety is implemented through a System Safety Process

The objective of system safety is to prevent accidents

System Safety Is Used in Many Industries

System safety analyses are being used to identify hazards and safety risks in the design and operation of:

- Commercial aircraft
- Automobiles
- Railroads and subways
- Passenger and freight ships
- Mine operation systems
- Chemical production plants
- Power generating plants, including nuclear
- Military systems
- Medical devices
- Space launch vehicles and space systems

System Safety Process

System Safety Process: the structured application of safety management and engineering principles, criteria, and techniques to address safety.

A System Safety Process generally includes the following elements:

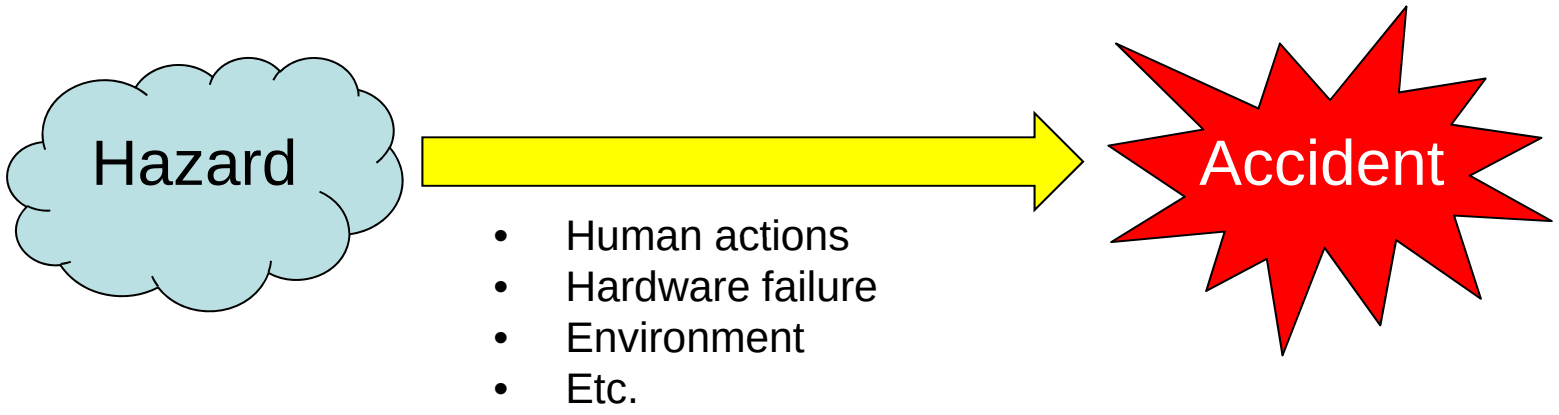
- Safety planning
- Hazard identification
- Hazard risk assessment, and associated risk decision making
- Risk reduction and hazard controls
- Risk reduction verification
- Hazard tracking and anomaly reporting

Hazard/Hazard Analysis Defined

Hazard: The potential for harm.

Hazard Analysis: Identification and evaluation of existing and potential hazards and hazardous conditions and the recommended mitigation for the hazard sources and risk found.

A hazard is the *potential* for harm, but hazards can become accidents if the conditions are right.



Software and Computing System Safety

Software Safety: the aspects of software engineering and software assurance that provide a systematic approach to identifying, analyzing, and tracking software mitigation and control of hazards and hazardous functions (e.g., data and commands) to ensure safer software operation ***within a system***.*

Software Safety:

- is a system safety issue
- is not just an activity for software engineers to worry about
- efforts should be included in each element of the system safety process

* from NASA-STD-8719.13B

Why Should We Worry?

- When system safety and software safety methods are used correctly they can help us prioritize resources, reduce risk, and prevent accidents.
- When applied inappropriately, system safety methods can lead to overconfidence, underestimation of risks, and spending resources on the wrong things.
- Broad concerns:
  - Failures in the implementation of the System Safety Process with respect to software and computing systems
  - Underestimation of the importance of software engineering and support processes (configuration management, quality assurance, etc.)

Case Studies:
System Safety Process

Why Study Accidents?

"Anticipating and identifying how a design can fail – or even just be perceived to fail – is the first step in making it a success."

- Henry Petroski, *Success Through Failure*

"Progress, far from consisting in change, depends on retentiveness. Those who cannot remember the past are condemned to repeat it."

- George Santayana

"I don't want to make the wrong mistake."

- Yogi Berra

System Safety Process

A System Safety Process generally includes the following elements:

- Safety planning
- Hazard identification
- Hazard risk assessment, and associated risk decision making
- Risk reduction and hazard controls
- Risk reduction verification
- Hazard tracking and anomaly reporting

Case Studies:
System Safety Process

Safety Planning

Safety Planning

- Safety planning includes:
  - planning for the management of system safety
  - emergency planning in case something goes wrong
- System safety planning is typically implemented through a System Safety Program Plan (SSPP)
- Emergency planning is typically implemented through emergency response plans, a.k.a., emergency action plans, mishap preparedness and contingency plans, or mishap response plans

Safety must be planned to be effective.

System Safety Planning: Accident

- On August 11, 1985, Union Carbide's Institute, West Virginia facility leaked methylene chloride and aldicarb oxime, toxic chemicals used to manufacture the pesticide Temik. Six employees were injured, and more than 100 residents were sent to the hospital.
- The release was traced to a storage tank that was overheated due to a leak in a steam valve, allowing steam to leak into a heating jacket.
- Overheating caused a runaway reaction to occur, resulting in release of toxic gases.

System Safety Planning: Software

- After the Bhopal accident, Union Carbide had installed a computerized air-monitoring system that could identify and detect releases of hazardous gases and predict whether the gases would be contained within the release area.
- Company officials initially stated that the computer that was supposed to inform them whether a gas cloud would be contained had failed.
- However, Union Carbide officials later admitted that the computer system had never been programmed to detect aldicarb oxime.
- The company had only purchased the basic (low cost) version of the cloud diffusion/prediction software.

System Safety Planning: Lessons

- System safety activities must begin early in development to be effective.
- Many safety decisions are actually made prior to acquisition or design.
- Risk decisions in relation to cost, schedule, performance, and safety are often defined in the acquisition strategy.
- Space system related mishaps:
  - CONTOUR
  - NEAR
  - Phobos 1

References:
Pool, B., “System Wrongfully Blamed in Union Carbide Leak : SAFER Fights Off Dark Cloud of Bad Publicity,” Los Angeles Times, August 20, 1985.

Schlager, N., *Breakdown: Deadly Technological Disasters,* Visible Ink Press, 1995.

Case Studies:
System Safety Process

Hazard Identification

Typical Hazard Identification Methods

Hazard: A potential for harm

Several approaches are typically used to help identify hazards:

- Hazardous element and component checklists based on previous experience and analyses
- Updating a hazard analysis developed from a previous program
- Use of design practices, regulations, and standards to assist in the development of analyses
- Identification of failure states --- failure to operate, operates incorrectly/erroneously, operates inadvertently, etc.
- Review of accident reports

Hazard Identification: Accident

- On January 8, 1997, an employee of Central Pre-Mix Concrete in Yakima, Washington was fatally injured during maintenance operations.
- Sand and gravel was transported by over-the-road trucks from an off-site pit to the plant for washing and screening. A blade mill with 3-foot diameter blades preconditioned the aggregates. The product was sold or used to supply the company's ready mix operation.
- The employee died when the blade mill in which he was working inadvertently started up.

Hazard Identification: Software

- At the time of the accident the employee was thawing frozen material inside the blade mill and replacing broken and worn paddle tips.
- While the employee performed this work, other workers were replacing a faulty breaker.
- The mill was controlled by a Programmable Logic Controller (PLC).
- The PLC logic had been modified in 1996 to address issues associated with power losses. However, this modification resulted in power being returned to components after a power failure.
- The faulty breaker was replaced and reset while the employee was in the mill, and resetting the breaker caused the PLC to power up as well. The PLC issued a command to power the mill on start-up.

Hazard Identification: Lessons

- Hazard identification must include:
  - software/computing system causes
  - failures of single components and scenarios where multiple things go wrong
- Software and computing system hazard analysis must include human error, process issues, and hardware failures in addition to software errors (coding, logic, inputs, commanding, etc.).
- Start-up and shut-down must occur in safe states.
- Space system related mishaps:
  - Viking inadvertent thruster firing
  - Mars Polar Lander
  - Ariane 5

Reference:
U.S. Mine Safety and Health Administration, Accident Investigation Report: Fatal Machinery Accident, Yakima - Pre-Mix #6, Central Pre-Mix Concrete Company, Yakima, Yakima County, Washington, January 8, 1997, Mine ID No. 45-00995, 1997.

Case Studies:
System Safety Process

Hazard Risk Assessment

Risk Assessment Defined

Risk: the potential for an undesirable consequence.

- *Likelihood*
- *Severity*

} Risk

Risk Assessment:

- What can go wrong?
- How likely is it?
- What are the consequences?

} Qualitative severity and likelihood "scoring methods" are used to assess risk, often supplemented by quantitative analyses

Risk Assessment: Accident

- On February 25, 2009, Turkish Airlines flight 1951 crashed short of the runway while attempting to land at Amsterdam Schiphol Airport on a flight from Istanbul, Turkey.
- Nine people were killed in the accident, including all three pilots.
- The accident report identified several factors leading to the crash, including the aircraft's automated responses to sensor failure, crew actions, and training.

Risk Assessment: Software

- As the aircraft was landing the left radio altimeter suddenly indicated an erroneous value of -8 feet.
- The autothrottle software should have handled this error and switched to the right radio altimeter. Negative values instead activated a computer system mode that set the engine thrust to IDLE.
- The aircraft lost speed, and the crew did not recognize the problem until it was too late.
- The risk of losing a radio altimeter had been evaluated prior to the accident based on failure data; the calculated risk fell below that necessary for corrective actions. However, the risk was calculated based on total flight hours instead of exposure time during the landing phase.

Risk Assessment: Lessons

- Software and computing system risks are often underestimated because of misunderstandings of complexity and false assumptions.
- Realistic risk assessments must include sensor (input) failures and effector faults in addition to software errors.
- The preconditions for mode change must be analyzed and understood.
- Effective testing must be implemented and include off-nominal conditions to validate risk assessments.
- Space system related mishaps:
  - X-31
  - Titan IV B/Milstar

Reference:
The Dutch Safety Board, “Crashed During Approach, Being 737-800, near Amsterdam Schiphol Airport, 25 February 2009,” Project Number M2009LV0225_01, May 2010.

Case Studies:
System Safety Process

Risk Reduction & Hazard Controls

Hazard Controls

- Implementation of safeguards (the hazard controls), and designing safety into the system (not analysis) reduces the risk.
- A basic tenet of system safety is that safety should be designed in, not simply added on after the fact.
- Hazard controls are used to design in safety to prevent accidents.
- Hazard controls are devices and approaches to mitigate risk presented by a hazard by reducing either the severity of the hazard or the probability of its occurrence.

Design Order of Precedence

System safety follows a preferred order of precedence for eliminating or mitigating risk:

- Eliminate the hazard or reduce the risk through design selection.
  - Substituting a hazardous material with a nonhazardous one, simplifying or modifying the design, or reducing the amount of hazardous material.
- Incorporate engineered safety features and devices.
  - Devices could be active features, such as elevator braking systems or interlocks, or passive systems, such as guardrails and personal protective equipment.
- Use warning devices
- Implement procedures and training

Hazard Controls: Accident

- On December 14, 2005, the upper reservoir of the Taum Sauk Pumped Storage Power Plant in Reynolds County, Missouri was overtopped during a pumping cycle, resulting in failure of an embankment.
- In normal operations, water flowed from the upper reservoir to a lower reservoir during the day to generate electricity during peak times, then was pumped back during off-hours using excess electricity.
- More than a billion gallons of water rushed through an empty campground and destroyed a home occupied by a state park superintendent and his family, sweeping them over a quarter mile away.

Hazard Controls: Software

- To reduce the risk of overtopping, water levels were monitored with pressure transducers; conductivity probes were also used in case of a problem with the pressure transducers.
- A Programmable Logic Controller (PLC) was designed to command shut down of the pumps when the water was 2 feet from the top; there was a 60-sec delay before alarms would sound to account for water movement. Automatic shutdown occurred when pressure transducers or two probes showed high water after the 60-sec delay.
- The pressure transducers became detached and gave erroneous readings.
- One of the two conductivity probes was placed too high; with the 60-sec delay, the pumps never shut down.
- The construction of the embankment and the lack of an overflow spillway contributed to the accident.

Hazard Controls: Lessons

- Designing out the hazard is more effective in most cases than safety devices, and passive safety devices are generally more effective than active (automated) controls.
- Redundancy may provide a false sense of security, and may not truly exist when we need it.
- Hazard controls must be independent and verifiable.
- Software hazard controls are only as effective as the sensors and effectors associated with them.
- Space system related mishaps:
  - STS-51F
  - STRV 1-C

Reference:
FERC Independent Panel of Consultants (IPOC), “Taum Sauk Upper Dam Breach FERC No. P-2277 Technical Reasons for the Breach of December 14, 2005,” May 24, 2006.

Case Studies:
System Safety Process

Risk Reduction Verification

Verification of Risk Reduction

- The safety verification and validation process is intended to determine that the design solution has met all the safety requirements (verification) and that the correct system is being built (validation).
- The verification and validation process, if performed correctly, will provide evidence that risk has been reduced.
- Hazard controls identified in the system safety process should be translated into a set of measurable and verifiable safety requirements to allow engineering implementation of those safeguards.

Verification Methods

Acceptable methods for verification

- Analysis, including models, simulations, algorithms, similarity
- Test, including functional, environmental, etc.
- Demonstration, such as clearances, accessibility, fit, reliability, etc.
- Inspection, such as workmanship, dimensions, quality, physical conditions, software documentation, etc.

Verification: Accident

- On June 22, 2009, a Washington Metropolitan Area Transit Authority (WMATA) Metrorail train traveling on the Red Line near the Fort Totten station struck the rear of another rail car.
- Nine people were killed in the accident, and 52 others were transported to area hospitals for treatment.
- The NTSB determined that the probable cause of the accident was a failure of the track circuit modules.

Verification: Software

- A “parasitic oscillation” generated by power output transistors of the track-circuit transmitter created a spurious signal.
- The spurious signal fooled the automatic train control system into thinking the track was clear when in fact another train was on the track.
- The automatic train control system allowed the second railcar to proceed forward and strike the car already on the track.
- Previous incidents of spurious signals had been reported.
- WMATA failed to employ system wide track circuit verification tests to assure that the automatic control system would function as expected under conditions of spurious signals.

Verification: Lessons

- Complex systems can be difficult to verify, and require multiple approaches.
- Testing should be conducted under realistic operating conditions, including known failures and anomalous conditions.
- Testing should not be limited to components but also to systems to identify unexpected interactions.
- Space system related mishaps:
  - Lewis Spacecraft
  - WIRE

Reference:
U.S. National Transportation Safety Board, “Collision of Two Washington Metropolitan Area Transit Authority Metrorail Trains Near Fort Totten Station, Washington, D.C., June 22, 2009,” NTSB/RAR-10/02, Synopsis July 27, 2010.

Case Studies:
System Safety Process

Hazard Tracking & Anomaly Reporting

Hazard Tracking/Anomaly Reporting

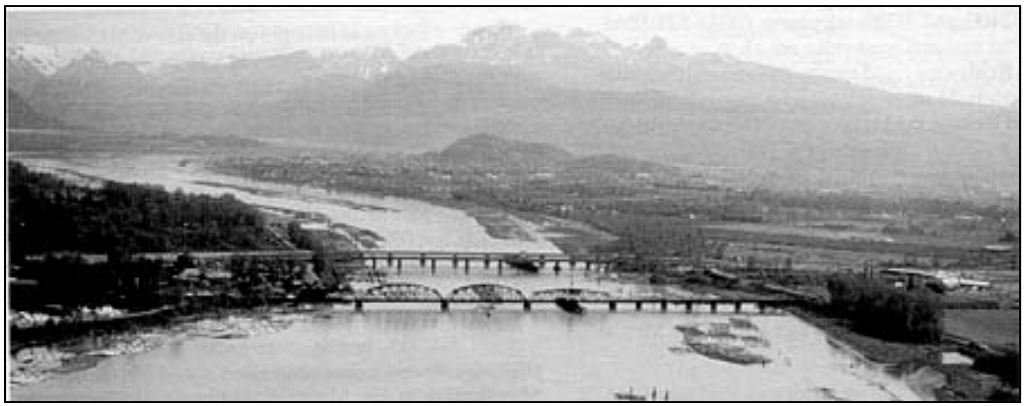
- Learning from failure is critical to improving safety.
- Hazards, controls, verifications, and problems discovered in development must be documented in order to learn.
- Hazard tracking and anomaly reporting are tools to help an organization learn.
- Simply having those tools is not enough – a process must be in place to assure that the tools are used to prevent accidents, and personnel must be trained and provided guidance in the use of the tools.

Anomaly Reporting: Accident

- On December 18, 2000, the tug *Miller Richmond* was towing two barges when one of the barges struck the Pitt River Highway Bridge in British Columbia, Canada. The collision caused extensive damage to the pier and the barge, but no one was injured.
- Just prior to the accident the tug was approaching the bridge and the tug was informed by the bridge tender that the south span of the bridge would not open.
- The tug started to turn the barges, then was informed that the bridge was open. In attempting to turn back the barges struck the pier.

Anomaly Reporting: Software

- Two sets of Programmable Logic Controllers (PLC) controlled the operation of bridge span electric motors. The bridge tender initiated an operation on the display screen and the PLC issued commands to open or close the bridge.
- On the day of the accident the bridge tender initiated operations from his display screen but the PLC never issued the command.
- The PLC system was installed in 1998, but testing of the new computerized system was insufficient following its installation.
- In the year 2000 approximately 40% of attempted bridge openings failed, but these problems were not sufficiently investigated or resolved.



Anomaly Reporting: Lessons

- Formal systems must exist to report anomalies and analyze trends.
- Organizations must do more than report anomalies, but they must act on them, including identification of root cause and corrective actions.
- Testing should not be limited to components; system testing should also be done to identify unexpected interactions.
- Space system related mishaps:
  - DART

Reference:
Transportation Safety Board of Canada (TSB), “Striking of a Bridge Tugboat *Miller Richmond* and Barges Miller 201 and Miller 206 Pitt River Highway Bridge British Columbia 18 December 2000,” Report Number M00W0303, May 6 2003.

Case Studies:

Software Engineering and Support Processes

Software Engineering & Supporting Processes

Inattention to software engineering and supporting processes has contributed to a number of accidents:

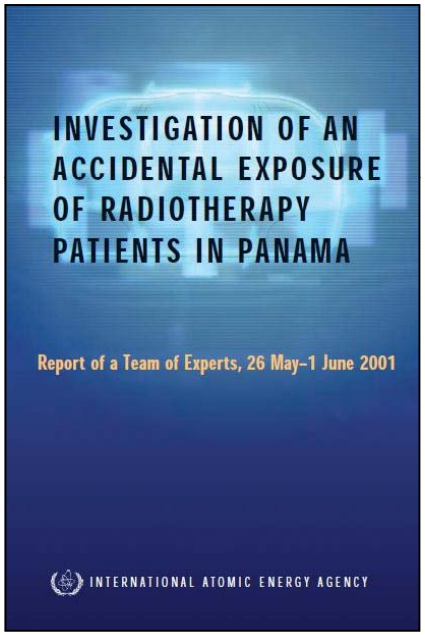
- Configuration/Change Management
- Quality assurance
- Maintenance
- Training
- Supplier Management
- Documentation

Configuration/Change Management: Hatch

- On March 7, 2008, the Edwin I. Hatch Nuclear Power Plant in Georgia was inadvertently forced into a controlled emergency shutdown.
- An update was installed to software used to perform both business functions and to monitor chemical and diagnostic data of primary control systems. After installing the update the computer system engineer rebooted the system.
- Rebooting the system caused the safety systems to erroneously interpret the lack of data as a drop in the water reservoirs that cool the radioactive fuel rods. System software then initiated the shutdown.
- Following this incident improvements were made to separate safety-critical software from business support software.

Quality Assurance: Panama Radiation Therapy

- Between Aug. 2000 and March 2001, 28 patients received excessive dosages during radiation treatments in Panama; many died from the treatments.
- Software calculates dosages and radiation times, in part based on number of protective “blocks”.
- Software limited inputs to 4 blocks, but doctors wanted 5 for some patients. Doctors used a single block 5 times as thick, not realizing calculations were different.
- Operating instructions were unclear, no operator feedback was provided, and output was not verified.
- No quality assurance procedures existed such as independent calculations, audits, process verification, inspections, etc.



Maintenance: America West Flight 2208

- On September 20, 1999, America West flight 2208 received a Ground Proximity Warning System (GPWS) warning near Agua Caliente, Arizona, indicating that the aircraft was too close to the ground.
- The crew executed an escape maneuver to avoid collision into terrain. In the escape maneuver two flight attendants were seriously injured.
- The GPWS warning was a false alarm; this airplane had 45 instances of erratic operation of the GPWS prior to this flight.
- Since 1988 Boeing had issued advisories that upgrades to hardware and software were needed to avoid false alarms.
- The operator failed to perform maintenance and upgrades on the unit.

Training: Pipeline Automated Control

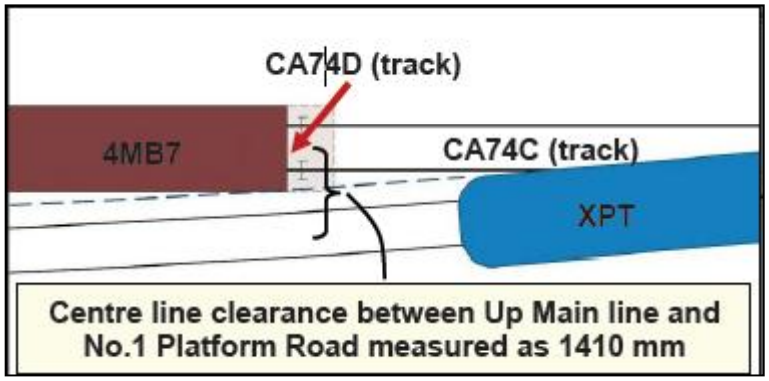
- On October 27, 2004, an 8-inch-diameter pipeline in Kansas ruptured, releasing approximately 204,000 gallons of anhydrous ammonia. Extensive environmental remediation was required.
- The leak was caused by damage during the original construction and later excavation activity.
- The pipeline controller received alarms from the automated control system, which were thought to be due to excessive ammonia delivery, not a leak. The controller overrode the alarms and automated shutdowns.
- The display included trend screens were which could have helped inform the controller of the leak, but access through the display panel was not obvious and he was not trained in their use.

Supplier Management: Allied Terminals

- On November 12, 2008, a 2 million gallon liquid fertilizer tank at Allied Terminals, Inc. in Chesapeake, VA collapsed. Two workers performing welding operations were seriously injured and a neighborhood was partially flooded.
- The CSB found that Allied had not assured that welds met accepted industry standards, and faulted Allied for its failure to perform inspections of the welds.
- The CSB also noted that the contractor hired by Allied to calculate the maximum fill height had used faulty assumptions in its analyses. The result was an overestimation of the allowable liquid level.

Documentation: Cootamundra

- On November 12, 2009, a passenger service train was being rerouted at Cootamundra, Australia, when the driver, having been told that the route was unobstructed, observed a freight train in his path. The driver applied the train brakes and stopped just before hitting the freight train.
- An investigation into the near miss determined a design error in the signaling software, which allowed the signal to declare the tracks clear even though a train was obstructing passage.
- The incident investigation faulted the documentation and quality assurance processes during the design phase for not being sufficiently robust to find such errors.
- Potential train clearance issues were identified by software engineers during development, but were not effectively documented and therefore were missed during review.



Overarching Software Safety Lessons Learned

Software Safety Lessons Learned: Accidents

Flaws in the software system safety process:

- Poor identification, documentation, and assessment of software and computing system hazards.
- Unrealistic software risk assessments.
- Risk reduction counting too much on good processes or testing.
- Testing efforts not focusing enough on actual operating environment and not sufficiently considering off-nominal conditions.
- Anomalies not factored into risk assessments or used to further explore potential system problems.

Software Safety Lessons Learned: General

- Software is not as easy to modify as we think - a poorly designed system will remain that way.
- Many catastrophic failures are due to latent design errors - past success does not guarantee an accident-free future.
- The complex interactions between software, hardware, and humans are often difficult to predict - the way the system performed in test may not be the way it will perform in operation.
- It is impossible to guarantee the absence of errors in specification, design, and development - if we don't look for them they will find us.
- Testing will never be as good as we think it is - we cannot expect to find all our problems in verification.

The Way Forward

Critical Questions

To promote improved software safety we should ask questions such as:

- Do plans reflect how business is really done?
- Is there a convincing story that the safety analysis is complete?
- Are the reports detailed enough? Are causes descriptive?
- Are the hazard controls primarily procedural rather than design changes, safety features or devices?
- Can the control strategy actually be implemented and verified?
- Has the risk assessment truly considered the worst case? What is the basis for the likelihood levels?
- Are problems found in test and design included in the hazard reports and factored into the design?

Hazard Identification

- The first step in hazard identification is understanding the system, including components, interfaces, and system boundaries.
- Find out what the software is being asked to do, in terms of functionality (not “software-ese”).
- It is not always obvious that software could cause a hazard; sometimes software monitors critical data and provides warnings, that if incorrect, could lead to an automated or manual response that could result in a hazard.
- Think in terms of loss of functionality (must work functions) and inadvertent activation (must not work functions).
- Think in terms of scenarios, not just software failure.

Hazard Cause Descriptions

- Descriptions should tell a mini story including source, mechanism, and outcome.
  - *Source:* an activity or a condition that serves as the root cause
  - *Mechanism:* a means by which the source can bring about the harm
  - *Outcome:* the harm itself that might be suffered
- Hazard causes should be clearly described
  - Vague: “Software error”
  - Better: “Software fails to issue a command leading to valve failing to open resulting in loss of reactor cooling”
- Identify the critical software components, data, and commands.

Specific Questions

- Are the sensors used for software decisions fault tolerant?
- Has mode transition been considered?
- Will the software properly handle spurious signals?
- Are checks performed before initiating hazardous operations?
- Will the software and system start up and shut down in a known, safe state?
- Does the software have a robust error handling capability?
- Etc.

Risk Assessments

Risk should be based on a number of factors including:

- Design complexity, including human interaction
- Maturity of software
- Degree of system testing
- Use of unproven technologies
- Experience of development and management personnel
- Robustness of system engineering efforts, including requirements management
- Robustness of quality, configuration management, supplier management, maintenance, and documentation processes
- Development and assurance processes for heritage and COTS software

Hazard Controls

- Combine generic good practices and specific safeguards
  - Generic requirements (coding standards, peer reviews, requirements traceability, etc.)
  - Hazard-specific controls, including human (manually closing a valve), software (automatically closing a valve), hardware (flow restrictors), and process-based controls
- Consider multiple, independent approaches
- Describe the control clearly and make sure it is verifiable
  - Vague: Operator will receive indications of valve failure
  - Better: OPSMGR CSCI will provide caution and warnings to operator if valve has failed under the following conditions

Testing

Testing should assure that software responds properly to system errors and failures (based on hazard analysis).

Tests should include at a minimum:

- Boundary conditions (in, out, crossing)
- Invalid inputs and outputs
- Input values of zero, approaching zero
- Minimum and maximum input rates
- Operator errors
- Process interrupts
- Hardware failures
- Events out of sequence

Safety testing is not the same as functional testing

Anomaly Reporting

- Do usable and effective reporting systems exist?
- Is there a common, formal system for reporting problems?
- Does a process exist to identify root cause?
- Are software problem reports factored into the hazard analysis and the design?
- Does the organization regularly review accident reports and mishap investigations?

Final Thoughts

Final Thoughts

- Software safety provides a systematic approach to identifying, analyzing, and tracking software mitigation and control of hazards and hazardous functions.
- Software safety is best implemented as part of a broader system safety process.
- Without a systematic process for identifying hazards and assessing risks we may underestimate the risk of accidents related to software and computing systems.

Final Thoughts

We should encourage a healthy skepticism in determining whether:

- Our plans are valid
- All hazards have been identified
- The risks make sense
- Our safeguards are effective
- We have verified our systems
- We learn from failure
- Our software engineering and support processes are robust

Contact Information

Terry Hardy
Director, Safety & Risk Management

Great Circle Analytics, LLC
1238 Race Street
Denver, CO, USA 80206
(720) 215-0994
thardy@gcirc.com

www.systemsafetyskeptic.com

Visit the web site for frequently updated system safety lessons learned, accident summaries, and mini-tutorials