Software Engineering for Safety

Rajender S. Bakhshi
Judith Hankin

Agenda

- We will look at what a safety critical system is.
- We will look the current techniques used to engineer safety critical systems.
- We will also introduce you to some directions that can be taken to improve safety engineering.

Introduction

- Many safety-critical systems rely on software to achieve their purposes.
- Safety-critical systems are:

  A computer, electronic or electromechanical system whose failure may cause injury or death to human beings. E.g. an aircraft or nuclear power station control system.

Cont…

- Important: Safety is a system problem.
- Software can contribute to:
  - A system's safety or
  - Compromise the system into a dangerous state.
- Thus, Software Engineers require a clear understanding of the software's role in, and interactions with the system.

Hazard Analysis

- **What is hazard analysis?**

The process involves analysing the system to find:

- its potential dangerous states
- associating levels of risk with these states
- estimating their probability of occurrence

- **Why should we carry out a hazard analysis?**

- performing one is vital in order to develop a safe system.
- they help us identify & categorise hazards the system must deal with
- It helps us determine requirement priorities & resources allocated to them at the requirements stage.

5

Some definitions

- **Accident/mishap** – *an unplanned event / chain of events that can lead to human death or injury.*

  Can also be extended to include *damage to the environment & property.*
- **Hazard** – *a danger or risk* (dictionary definition).

- **System level hazard** – *a state that can lead to an accident.*

6

More detail

- **Who should carry out the hazard analysis?**
  - Systems engineers
  - Domain experts
  - Safety advisors

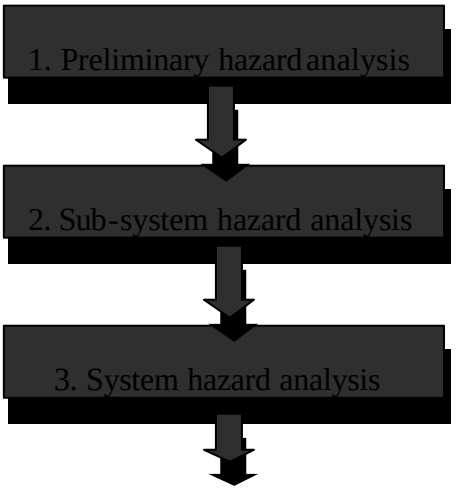
These groups should carry out the hazard analysis together and draw from their different areas of expertise to identify the hazards.

- **When should we carry out a hazard analysis?**
  - Prior to the requirements stage of the project.

- **How should we carry out the hazard analysis?**
  - Often, especially with large systems, it is more efficient to structure a hazard analysis into different phases. (N. G. Leveson, 1986).

7

Phases (1)



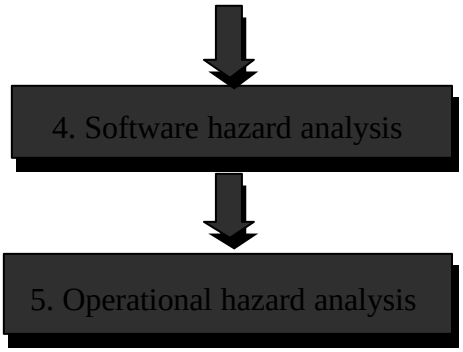
Principle hazards identified.
Risk of hazard assessed.

Conduct more detailed analysis
For safety-critical sub-components.

Hazards that arise when when s
Sub-system components are
Integrated.

8

Phases (2)



Concerned with finding software related hazards.
Can be done in phases 2 & 3.

Deals with hazards which appear as Result of using the system.
Concerned with user interface & Operator error.

We need to record details about the hazards identified during the different stages of analysis e.g hazard identified, p(occurrence), severity of occurrence & estimated risk.

Analysis Techniques

- We can use a variety of methods to determine the components that contribute to the existence of hazards and those that prevent the occurrence of a hazard. Such methods include:

- Fault tree analysis
- FMECA - **f**ailure **m**odes, **e**ffects & **c**riticality **a**nalysis
- HAZOP - **haz**ards & **op**erability analysis.

Hazard Analysis & Requirements

- By performing a hazard analysis we can identify the safety requirements that need to be incorporated into our software.

- The safety requirements act as constraints on the software which may be required to have methods of:
  - **Prevention** - not allowing the system to enter hazardous states.
  - **Detection** - spot when the system has entered dangerous state(s).
  - **Correction** - move the system from a dangerous state.

Safety Requirements Specification & Analysis

- Common tools used in the design of safety-critical systems are redundancy and formal methods.
- Redundancy (focussed more on hardware):

  Used to detect and recover from errors, either in hardware or software.
- Formal:

  Mathematically based techniques for the specification, development and verification of software and hardware systems.

Specification cont…

- Formal methods also enable to investigate whether safety properties are reserved.
- Example – Avionics System

  "If the backup channel is in control and is in safe state, it will stay in a safe state".

Specification cont…

<u>System safety requirements V Software requirements</u>

There has been a problem in the translation between the system safety and software requirements.

(I.e. safety cases focus more on which software components are critical whereas software focuses more on development process rather than whether it satisfies the system safety requirement.)

Specification cont…

- To reduce this discontinuity we could reflect on how users actually use specifications to think about the complex systems.

Example:

- Focus on interface between user and controller (displays).

Designing For Safety

- Dependability is a common theme. Concerned with fault tolerance - *providing acceptable service even if faults occur* (not possible in all systems). Also common to real-time & secure systems.

- Software design must incorporate methods for:
  - Prevention - by e.g mutual exclusion, timeouts etc...
  - Detection
  - Correction

} Exception handling, warnings to operators/users, self -tests etc...

Obstacles to Designing Safe Systems

- **Design trade-offs**

  The safety - desirable attributes trade-off. Design methods for fault tolerance can both contribute to (e.g providing predictable timing behaviour) and compromise (e.g by introducing more interactions between system components) system safety.

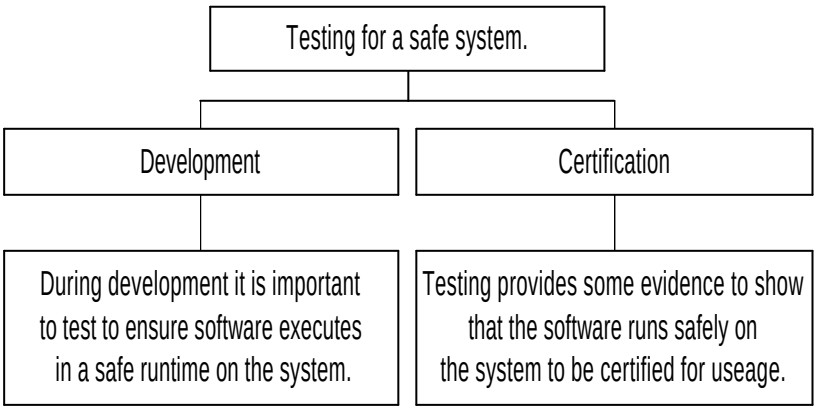
  Vulnerability to simple design errors

  Many accidents have simple causes. Assuming "small errors have small consequences" is not true in all cases e.g. *Mars Climate Orbiter*.
- **Limited use of known design techniques**

  Known good practice design techniques are not always used.

17

Testing



18

Testing

- Can verify fault tolerant aspects of the software.
- Can determine software responds appropriately.

- When testing systems we can categorise 3 main assumptions.
- Assumptions about environment - *Unsafe systems can arise due to incorrect assumptions in which the system will operate.*

Testing cont…

- Assumptions about users – *Unsafe systems can also arise due to incorrect assumptions of the user or operator of a system.*
- Assumptions about operations – *Deep knowledge and experience with an application is required for test cases to be drawn.*
- Testing is not a sufficient condition for a safe system. Failure-free tests need to be generated after failures, showing a safer system.

Certification & Standards

- Currently there are a number of problems regarding safety standards.
- To certify such a system is more complex & less well defined process than certifying non-safety critical software.
- Large safety critical systems could be comprised of many sub-systems of different domains. They could contain COTS (commercial off the shelf) software certified by less rigorous standards and by differing bodies which if integrated, would have to be re-certified.
- There is a growing need for international certification standards to be put into place.

Resources

- Safety Critical Systems (SCS):
  - IEEE video on “Developing Software for Safety Critical Systems”.
  - Bowen’s website on SCS: http://www.afm.sbu.ac.uk/

DIRECTIONS

- There is work needed in the following areas to improve the current status of safety engineering.

- Further integration of formal & informal methods.
- Constraints on safe product families & reuse.
- Testing & evaluating safety critical systems.
- Runtime monitoring.
- Education.
- Collaboration.

Further Integrating Formal & Informal Models

- **Automatic translation of informal notations into formal models.**

  Recent research in software engineering concentrates on trying to close the gap between descriptive notations & formal models.

  Descriptive notations are widely used by software engineers but do not lend themselves to automatic analysis unlike formal models.

  Descriptive Notations - e.g UML.
  Formal Models - e.g Fault tree analysis.

Integrating previously distinct formal methods

- Different methods have different strengths.
- Integrating different methods allows you to specify/analyse software at the level of detail that you want.
- Further use of formal methods aids when specifying the software/system interface. Incorrect assumptions about this interface can lead to states occurring that could compromise safety.

Constraints on safe product families and safe reuse

Safety analysis of product families

This entails how systems with similar requirements can reuse requirements analyses.

In terms of safety it is hard to characterise, formalise, and verify due to minor variations amongst systems (requirements, environment, platform).

Safe reuse of COTS software

2 main problem areas associated with this field:

- COTS software does what it is supposed to do (fitness for application)!
- COTS software confirms that it does not do what it's not supposed to do!

Testing & Evaluation

- **Requirements based testing**
  - Need to link safety requirements & test cases better. This can be done by improving test case generation & further integration of testing & requirements tools.
- **Evaluation from multiple sources**
  - "the safety & trustworthiness of the system will rest on a tripod made up of testing, mathematical review, and certification of personnel and process" Parnas, van Schouwen & Kwan.
- **Model consistency**
  - Model actual behaviour of the system as well as the operator's mental model of how they think the system behaves. Both models can be cross checked to make sure that any inconsistencies are identified & dealt with.

Runtime Monitoring

- RM = When software is used to monitor and respond to operational activity.
- It can detect and recover from hazardous states enhancing safety.
- Detection of faults leads to Problems:
  - Tradeoffs between increased safety / increased complexity.
  - Decreased availability.
  - Decreased performance.
- Example of use: against hacker attacks!!!

Education

- Suggestions include increasing the amount of exposure that undergraduate students at university get to issues concerning safety critical systems.
- There is also a need for safety courses to be aught with more links to fault-tolerance, security, systems engineering & experimental techniques.

Safety Related Fields

- Security + Survivability
- Software Architecture
- Theoretical computer science
- Human factors engineering

Summary

- We have examined a number of techniques used in safety engineering to develop safety critical systems.
- It is clear that there are some limitations to these techniques which have resulted in software failure within such systems.
- The future of safety engineering lies in advances in related fields, better testing and analysis techniques and exposing future developers to issues concerning SCS's earlier in their careers.

References

- "Safeware" N. Leveson. Addison-Wesley.
- "Software Engineering" Ian Sommerville. Addison-Wesley.
- "Software safety: why, what and how?" ACM Computing Surveys.