CS_411 Critical Systems:

http://www-compsci.swan.ac.uk/~csetzer/lectures/02/index.html

Dr. Anton Setzer

http://www-compsci.swan.ac.uk/~csetzer/index.html

Lent Term 2003
A0. Introduction, Overview

(a) A case study of a safety-critical system failing.

(b) Two aspects of critical systems.

(c) Administrative Issues.

(d) Plan.

(e) Literature
(a) A Case Study of a Critical System Failing

**Definition:** A critical system is a

- computer, electronic or electromechanical system
- the failure of which may have serious consequences, such as
  - substantial financial losses,
  - substantial environmental damage,
  - injuries or death of human beings.

Critical Systems, CS_411, Lentterm 2003, Sec. A0
Three Kinds of Critical Systems.

- **Safety-critical systems.**
  - Failure may cause injury or death to human beings.
  - Main topic of this module.

- **Mission-critical systems.**
  - Failure may result in the failure of some goal-directed activity.

- **Business-critical system.**
  - Failure may result in the failure of the business using the system.
Examples of Critical Systems

- **Safety-Critical**
  - Medical Devices.
  - Aerospace
    * Civil aviation.
    * Military aviation.
    * Manned space travel
  - Chemical Industry.
  - Nuclear Power Stations.
  - Traffic control.
    * Railway control system.
    * Air traffic control.
    * Road traffic control (esp. traffic lights).
  - Other military equipment.

Critical Systems, CS_411, Lentterm 2003, Sec. A0
Examples of Critical Systems (Cont.)

- **Mission-critical**
  - Navigational system of a space probe.
Examples of Critical Systems (Cont.)

- **Business critical**
  - Customer account system in a bank.
  - Online shopping cart.
  - Areas where secrecy is required.
    - Defense.
    - Secret service.
    - Sensitive areas in companies.
Åsta Train Accident (January 5, 2000)

Report from November 6, 2000

http://odin.dep.no/jd/norsk/publ/rapporter/

Critical Systems, CS_411, Lentterm 2003, Sec. A0
75 persons on board
express

Rena

Road crossing

10 persons on board
local

green

red?

10 persons on board
express
Sequence of Events:

- Railway with one track only. Therefore crossing of trains possible.

- According to timetable crossing of trains at Rudstad.
Train 2302 is 21 minutes behind schedule. When reaching Rena, delay is reduced to 8 minutes. Leaves Rena after a stop with green exit signal 13:06:15, in order to cross 2369 at Rudstad.

Train 2369 leaves after a brief stop Rudstad 13:06:17, 3 minutes ahead of timetable, probably in order to cross 2302 at Rena.
Rena
Train 2302
75 passengers
Rudstad
Train 2369
10 passengers

- Local train shouldn't have had green signal.

- **13:07:22** Alarm signaled to the rail traffic controller (no audible signal).

- Rail traffic controller sees alarm approx. **13:12**.

- Traffic controller couldn't warn trains, because of use of mobile telephones (the correct number hadn't been passed on to him).

- Trains collide 13:12:35, 19 persons are killed.
Investigations

• No technical faults of the signals found.

• Train driver was not blinded by sun.

• **Four incidents** of wrong signals with similar signaling systems:
  
  – Exit signal green and turns suddenly red. 
    Traffic controller says, he didn’t give an exit permission.
  
  – Hanging green signal.

  – Distant signal green, main signal red, train drives over main signal, pulls back.
    Traffic controller surprised about the green signal.

  – **18 April 2000** Train has green exit signal.
    When looking again, notices that the signal has turned red.
    Traffic controller hasn’t given exit permission.
• Several safety-critical deficiencies in the software found (some known before!)

• The software used was completely replaced.
• **SINTEF** (Foundation for Scientific and Industrial Research at the Norwegian Institute of Technology) found no mistake leading directly to the accident. **Conclusion of SINTEF:** No indication of abnormal signal status. ⇒ Mistake of train driver (died in the accident). (Human Error).

• Assessment of report by **Railcert**
  – Criticism: SINTEF was only looking for single cause faults, not for multiple causes.

Critical Systems, CS_411, Lentterm 2003, Sec. A0
Conclusion

- It is possible that the train driver of the local train was driving against a red signal.

- The fact that he was stopping and left almost at the same time as the other train and 3 minutes ahead of time, makes it likely that he received an erroneous green exit signal due to some software error. It could be that the software under certain circumstances when giving an entrance signal into a block, for a short moment gives the entrance signal for the other side of the block.

  One possible reason for that could be a level road crossing or race conditions.

- Even if this particular accident was not due to a software error, apparently this software has several safety-critical errors.
Conclusion (Cont.)

- In the protocol of an extremely simple installation (Brunna, 10 km from Uppsala), which was established 1957 and exists in this form in 40 installations in Sweden, a safety-critical error was found when verifying it with a theorem prover formally 1997.

- Lots of other errors in the Swedish railway system were found during formal verification.
Causal Factors.

We consider a **three-level model** (Leveson, pp. 48 – 51).

- **Level 1: Chain of events.**
  - Described above.

- **Level 2: Conditions**, which allowed the events on level 1 to occur.

- **Level 3: Conditions and Constraints,**
  - that allowed the conditions on the second level to cause the events on the first level, e.g.
    - Technical and physical conditions.
    - Social dynamics and human actions.
    - Management system, organizational culture.
    - Governmental or socioeconomic policies and conditions.
Problems found in a Level 3 analysis form the root causes.

- **Root causes**, weaknesses in general classes of accidents which contributed to the current accident but might affect future accidents.
- **If** the problem behind a root cause is not fixed, **almost inevitably an accident will happen again**.
- Many examples, in which despite a thorough investigation the root cause was not fixed and the accident happened again.
- **Example: DC-10 cargo-door saga.**
  - Faulty closing of the cargo door caused collapsing of the cabin floor.
  - One DC-10 crashed 1970, killing 346 people.
  - As a consequence a fix to the cargo doors was applied.
  - The root cause, namely that the collapsing of the cabin floor when the cargo door opens wasn't fixed.
  - 1972, the cargo door latch system in a DC-10 failed, the cabin floor collapsed and only by chance the plane was not lost.
Level 2 Conditions

- The driver left the station although the signal should have been red.
  - The local train probably had for short period a green light, caused by a software error.
  - Incidents had happened before, but were not investigated.
  - Software wasn’t written according to the highest standards.

- The local train left early.
  - Due to the fact that the train driver was relying on his watch (which might go wrong) and not on an official clock.

- The local train drove over a possibly at that moment red light.
  - There was no ATC (automatic train control) installed, driving over a red light.
Level 2 Conditions (Cont.)

- The traffic controller *didn’t see the control light*.
  - Control panel was badly designed.
  - A visual warning signal is not enough, in case the system detects a possible collision of trains.

- The rail controller *couldn’t warn the driver*, since he didn’t know the mobile telephone number.
  - To rely on a mobile telephone network in a safety-critical system is extremely careless.
    * Mobile phones often fail.
    * The connections might be overcrowded.
    * Connection to mobiles might not work in certain areas of the railway network.
  - The **procedure for passing on the mobile phone number** wasn’t managed.
• The fire safety of the train engines was not very good.
Level 3 Constraints and Conditions

- **Cost-cutting precedes many accidents.**
  - Difficult, to maintain such a small railway line.
  - Resulting cheap solutions might be dangerous.

- **Flaws in the software.**
  - Control of railway signals is a safety-critical system and should be designed with high level of integrity.
  - Very difficult to write correct protocols for distributed algorithms.
  - Need for **verified design** of such software.
Level 3 Constraints and Conditions (Cont.)

- **Poor human-computer interface** at the control panel.
  - Typical for lots of control rooms
  - (problem in nuclear power stations
  - criticality of such a design is not yet sufficiently acknowledged
  - (see problems with the UK airtraffic control system, which is often difficult to read.)
• The railway controller was **overworked**.

• **Overconfidence in ICT.**
  
  – Otherwise one wouldn’t have used such a badly designed software.
  – Otherwise one wouldn’t have simply relied on the mobile phones; a special agreement with the mobile phone companies should have been set up.

• **Flaws in management practices.**
  
  – No protocol for dealing with mobile phone numbers.
  – No mechanism for dealing with incidents.
    * A mechanism should have been established to thoroughly investigate them.
    * Most accidents are preceded by incidents, which are not taken seriously enough.
Lessons to be Learned

- Safety-critical systems are very complex – **System aspect**.
  - **Software**, which includes parallelism.
  - **Hardware**.
    * Might fail (light bulb of a signal might burn through).
    * Have to operate under **adverse conditions** (low temperature, snow).
  - **Human-computer interaction**.
  - **Protocols** the operators have to follow.
  - **Training** of operators.
  - **Cultural habits**.
A *sequence of events* had to happen in order for the accident to take place.

- **Preliminary events.**
  - events which influence the initiating event.
  - without them the accident cannot advance to the next step (initiating event).

In the main example:
- Express train is late. Therefore crossing of trains moved from Rudstad to Rena.
- Delay of the express train reduced. Therefore crossing of trains moved back to Rudstad.

- **Initiating event, trigger event.**
  - Mechanism that causes the accident to occur.

In the main example:
- Both trains leave their stations on crash cause, maybe caused by both trains having green signals.
Lessons to be Learned (Cont.)

- **Intermediate events.**
  Events that may propagate or ameliorate the accident.
  * Ameliorating events can prevent the accident or reduce its impact.
  * Propagating events have the opposite effect.
- When designing safety critical systems, **one should**
  * avoid triggering events
    · if possible by using several independent safeguards,
  * add additional safeguards, which **prevent a triggering event causing an accident** or reduces its impact.
Lessons to be Learned (Cont.)

- Usually at the end of an investigation conclusion “human error”.
  - The architecture of the software was investigated but no detailed search for a bug was done.

- Afterwards, concentration on trigger events, but not much attention to preliminary and intermediate events – root cause often not fixed.

- Most failures of safety-critical systems were caused by multiple failures.
(b) Two Aspects of Critical Systems

• (1) Software engineering aspect.
  – System aspect.
    * Computer system.
    * Hardware connected (hardware failure in sensors, relays and computers).
    * Interaction with environment.
    * Human-machine interaction.
  – Methods for identifying hazards and measuring risk (HAZOP, FMEA, etc.)
  – Standards.
  – Documentation (requirements, specification etc.)
  – Validation and verification.
  – Based on techniques used in other engineering disciplines (chemical industry, nuclear power industry, aviation).
Two aspects (Cont.)

- **(2) Tools for writing correct software.**
  - Software bugs can not be avoided by careful design.
    - Especially with distributed algorithms.
  - Need for verification techniques using formal methods.
  - **Different levels of rigour:**
    1. Application of formal methods by hand, without machine assistance.
    2. Use of formalized specification languages with some mechanized support tools.
    3. Use of fully formal specification languages with machine assisted or fully automated theorem proving.
  - However, such methods don’t replace software engineering.
    - Formal methods idealize a system and ignore aspects like hardware failures.
Two Streams in this Module

- Stream A
  - **Software engineering aspect** + general overview over formal methods.
  - **Industrial practices.**
Stream B (interleaved with Stream A):

- Closer look at **one prototype example** of a tool which allows to write 100% correct software.
- **Programming with dependent types**, based on Martin-Löf type theory.
- **Still area of research**, a few successful industrial applications (using the theorem prover Coq).
- Use of the theorem prover **Agda**.
  * This part will be heavily **machine based**.
  * **Experimental system**.
  * Interesting: use of a **theorem prover**, which is used like a **programming language**.
  * Application of its ideas not limited to safety critical software.
    - Dependent types will sooner or later be used in ordinary programming languages.
      (**Templates** in C++ or soon in Java (?) is one approximation.)
- Goal is that students have been in contact with one proof assistant.
Assessment

- 80% Exam.
  - One question concerning stream A.
  - One question mixture of stream A and B.
  - One question concerning stream B.

- 20% coursework:
  - 4 small assignments. Each counts 5% (Plan, might be changed).
  * Handed out approx. every 2nd week.
  * Due two weeks later.
  * Mainly associated with stream B.
Timetable, Course Material

• Two lectures per week.
  – Monday, 13:00, Robert Recorde Room.
  – Thursday, 12:00, Robert Recorde Room.

• Web page contains overhead slides from the lectures. Course material will be continually updated.
• Learning outcome:
  – Familiarity with issues surrounding safety-critical systems.
  – Understanding of techniques for specifying and verifying software.
  – Experience with one proof-assistant.
A0. **Introduction, overview.**
A2. Hazard and risk analysis.
A4. Fault tolerance.
A5. The development cycle of safety-critical systems.
A6. Design methods and tools.
A7. Formal methods.
A8. Verification, validation, testing.

- Probably not all topics covered (last year only A3 was reached).
Plan Stream B

B1. Introduction.
B2. The logical framework.
B3. Data types.
B4. Interactive programs in dependent type theory.
B5. Case studies.
In general, the module is self-contained.

In the following a list of books, which might be of interest to study critical systems more intensively.
Books Relevant for Stream A

- **Main course book:**

- **Supplementary books on software-engineering aspects:**
    Intended as a short book for engineers of many disciplines.
    Concentrates mainly on human and sociological aspects.
    A report on a lot of (100?) accidents and incidents of software.
• Some books on general formal methods:


  – John Barnes: *High integrity Ada. The SPARK approach.* Use of a subset of Ada with proof annotations in order to develop secure software. Developed and used in industry.

Literature Relevant for Stream B


- Aarne Ranta: *Type-theoretic grammar*. Clarendon Press, 1995. Use of type theory in linguistics and for translation between languages. Supposed to have a good and simple introduction into type theory.