Design and Analysis of Distributed Interacting Systems

Organization

Prof. Dr. Joel Greenyer

April 4, 2013
Organization

• Lecture held by: Joel Greenyer
  – office: g322
  – email: greenyer@inf.uni-hannover.de
  – office hours: please make appointment via email
• Lecture will be held in English

• Tutorials: Raphael Pham
  – email: raphael.pham@inf.uni-hannover.de
About Me

• I am the new Software Engineering Assistant Professor (Juniorprofessor)
• Before (until last week) I was a researcher at the Politecnico di Milano, Italy
• Research Interests: What I will tell you about in this lecture...
  – ...and other things, you are welcome to ask!
Organization

- Lecture: Thursdays, 10:15 – 11:45, F 128
- Tutorial: Thursdays, 14:00 – 14:45, G 323
  - first tutorial: April 11
- No lecture on May 23. (Widsun Break, Pfingsten)
  - and we will be at the ICSE conference in San Francisco

- Script: Lecture Slides, will be available after each lecture
- Grades:
  - Oral Exam, in English or German
  - Dates: t.b.d., probably some time in August
  - Prerequisite: 50% of total points in exercises and have presented an exercise solution during tutorial / lecture
  - Grade +0.3 with 80% of points in exercises
Design and Analysis of Distributed Interacting Systems

Lecture 1 – Introduction

Prof. Dr. Joel Greenyer

April 4, 2013
Overview

➔ Examples of distributed and interacting systems
  • Accidents caused by software flaws in history

• What this lecture will be about
Distributed Interacting Systems

- Multiple components interact
  - to fulfill increasingly complex functions
  - sometimes distributed physically
  - in diverse and sometimes safety-critical situations
Modern Railway Systems

• They run more and more autonomously today
  – during normal operation, the drivers of today's high-speed trains rarely have to intervene

• Research Project *Neue Bahntechnik Paderborn (RailCab)*:
  – concept for the future rail traffic
  – small shuttles transport goods and passengers on demand
  – they form convoys in order to reduce the wind resistance
  – many diverse situations, changing communication relationships

see also: http://www.youtube.com/watch?v=JPtHYNvumoc
“It takes dozens of microprocessors running 100 million lines of code to get a premium car out of the driveway, and this software is only going to get more complex”


• Autonomous driving on highways or even in cities may become reality in twenty years.

see Volvo SARTRE road trains video:
https://www.youtube.com/watch?v=-jQ1U9KZfWg
Production Systems, Other Flow Systems

- Often controlled by many processors
- Many diverse sensors and actuators
- Software interacts heavily with mechanical components
- Timing aspects of the physical movement of materials becomes relevant

http://www.siemens.com
Mobile Systems

- Green Move (Car sharing project at Politecnico di Milano)

Diagram:
- Ad seller
- Municipal info
- Pollution info
- Traffic info
- Green Move Center
- External Users
Mobile Systems

- e-health systems
(on embedded systems:

In the (recent) past, an embedded system would be either small or simple, or the composition of almost non-interacting imported and assembled components. The trend is that the number and complexity of functions will increase drastically. Increasing complexity is making the present design methodologies rapidly obsolete. Productivity of the order of six (or less!) lines of embedded code per day per person is common in HRT embedded systems. If we do not have a breakthrough in design methodology and tools, the inefficiency of the embedded software development process will prevent novel technology to enter the market in time. The cost of developing a new plane (of the order of several billions of Euros) is about \( \frac{1}{2} \) related to embedded software and electronics subsystems.
Software Engineering Challenges

• Software is around us in more and more areas of our daily life
  – makes our lives more comfortable and safer

• The more we rely on software, the more drastic are the consequences of software failures

• The more complex software becomes, the harder it gets to avoid software failures
Overview

• Examples of distributed and interacting systems
  ➔ Accidents caused by software flaws in history

• What this lecture will be about
(Some) Accidents Caused by Software Flaws in History

- 1978: F16 fighter turns upside-down due to a sign error when flying over the equator (found during simulation)
- 1985-87: Several radio therapy patients are killed or injured due to software faults of the Therac-25
- 1995: Problems in the interplay of hardware and software at Denver Airport causes damage and loss of luggage
- 1999: Mars Polar Lander interprets shock from extending landing gear as ground contact. It crashes onto surface.
- 2005: London Tower Bridge cannot be closed due to a software fault
Therac-25

• Computer-controlled machine for radio-therapy to treat cancer
• produced by AECL (Atomic Energy of Canada Limited)

• Between 1985 and 1987 (at least) six patients received severe overdoses
  – three died as a result from the overdose
  – one patient died shortly after the overdose, but the death was attributed to the cancer
  – the others suffered permanent disabilities
• Some accidents were never fully investigated. Also, no detailed documentation on the software, its development and quality control process is available

• Summary from lawsuits: Nancy Leveson: Medical Devices: The Therac-25 (http://sunnyday.mit.edu/papers/therac.pdf)
  – The material on the following slides is taken (mainly) from this paper
Background

• In radio-theraphy, cancer tissue (a tumor) is destroyed using high-energy radiation
  – typical radiation: X-Rays, electrons, protons
  – X-Rays: can reach deeper tissue
  – Electrons: beam concentration falls off rapidly, thus suited for more shallow tissue, e.g. skin cancer
• the surrounding (healthy) tissue should not be damaged
• correct dosage and focus is very important

• The Therac-25 could produce an X-Ray and electron beam
  – “dual mode”, more economic
Figure from Nancy Leveson:
Medical Devices: The Therac-25
Turntable Assembly

- Monitors turntable position
- Reflects light to see where beam will strike
- To spread electron beam
- Lock turntable into position

Figure from Nancy Leveson: Medical Devices: The Therac-25
• For X-Ray therapy a high-energy electron beam hits a tungsten target, which create a beam of X-Rays
• For electron therapy, the electron beam is used directly
• In X-Ray mode, the electron beam energy is “some 100 times greater than that for electron therapy”
Background

• Earlier models (Therac-6 and -20) were developed in collaboration with CGR, a French company.
• The software of the Therac-20 was reused in the Therac-25, but the collaboration between AECL and CGR ended after the development of the Therac-20.

• The earlier models had microcomputers, but could also be operated without them.
• In the Therac-25, some hardware safety-features of the turntable positioning present in earlier models (extra control circuits, mechanical interlocks) were replaced by software (run on the microcomputer).
  – Cheaper, “software does not wear-out”
### Operator Interface

- Operator manually sets parameters on the machine
- Then leaves the room, and enters data in terminal
- The treatment is permitted only if values match

```
<table>
<thead>
<tr>
<th>PATIENT NAME</th>
<th>TEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>TREATMENT MODE</td>
<td>FIX</td>
</tr>
<tr>
<td>BEAM TYPE: X</td>
<td></td>
</tr>
<tr>
<td>ENERGY (MeV): 25</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ACTUAL</th>
<th>PRESCRIBED</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIT RATE/MINUTE</td>
<td>0</td>
</tr>
<tr>
<td>MONITOR UNITS</td>
<td>50</td>
</tr>
<tr>
<td>TIME (MIN)</td>
<td>0.27</td>
</tr>
</tbody>
</table>

| GANTRY ROTATION (DEG)   | 0.0       | 0          | VERIFIED |
| COLLIMATOR ROTATION (DEG) | 359.2     | 359        | VERIFIED |
| COLLIMATOR X (CM)       | 14.2      | 14.3       | VERIFIED |
| COLLIMATOR Y (CM)       | 27.2      | 27.3       | VERIFIED |
| WEDGE NUMBER            | 1         | 1          | VERIFIED |
| ACCESSORY NUMBER        | 0         | 0          | VERIFIED |

| DATE       | 84-OCT-26   |
| TIME       | 12:55:8     |
| OPR ID     | T25V02-R03  |
| SYSTEM     | BEAM READY  |
| TREAT      | TREAT PAUSE |
| OP. MODE   | TREAT      |
| AUTO X-RAY | 173777     |
```
Operator Interface

• The operators complained that it took too long to enter data
  – a function was implemented which allowed operators to use old values by just hitting a series of carriage returns

• The system could shut down when it detected an error condition
  – operators could continue the treatment by hitting “proceed”
  – this could be done five times before a complete system reset was necessary

• Error messages were cryptic and not documented
  – “An operator involved in one of the accidents testified that she had become insensitive to machine malfunctions”
“It was not out of the ordinary for something to stop the machine. . . . It would often give a low dose rate in which you would turn the machine back on. . . . They would give messages of low dose rate, V-tilt, H-tilt, and other things; I can’t remember all the reasons it would stop, but there was a lot of them.”

Operator testimony, from Nancy Leveson: *Medical Devices: The Therac-25*
Initial Hazard Analysis

• An initial hazard analysis was performed using a fault tree
  – top-down analysis of what can cause an undesired system state

• The following assumptions were made
  – Programming errors have been reduced by extensive testing---
    no programming errors were included in the analysis!
  – The software does not wear out
  – Computer error are only caused by random hardware errors (bit
    flips) caused by radiation

• The following probabilities were selected, without justification
  – “Computer selects wrong energy”: $10^{-11}$
  – “Computer selects wrong mode”: $4 \times 10^{-9}$
1. Kennestone Regional Oncological Center, June 1985
A patient was set up for treatment of tumor in the clavicle area (dt: Schlüsselbein).

After the treatment, the patient said to the technician “You burned me”, “she felt a tremendous force of heat…”

The patient's skin reddened and her shoulder froze.

Her back was also red, as if the burn went through her body.
• Physisists later estimated that she received 15,000 – 20,000 rads (a single treatment is usually only ~200 rads)

• The patient's breast had to be removed because of the burns
• Her shoulder remained paralyzed, she was in constant pain

• The manufacturer refused to believe that burns were caused by the Therac-25.
• The lawsuit was settled out of court
• No investigation took place
Incidents

2. Ontario Cancer Foundation, Hamilton, Ontario, Canada, July 1985
Ontario Cancer Foundation, July 1985

• A patient was treated for a cancer in the cervix (dt: Gebärmutterhals)
• After activation, the machine shut down with an “H-Tilt” error message
• The operator pressed “P” (proceed), the display showed “no dose” every time.
• The patient complained about a burning sensation
• She died four months later
• The autopsy revealed that she died of cancer, but that radiation damaged her hip so that she would have needed a total hip replacement
• The AECL was informed and found that a bit-error in a 3-bit signal about the turntable position could have been the reason

• They changed the system to tolerate 1-bit errors
  – and some other changes

• AECL then claimed that the system safety was increased by five magnitudes (100,000 times)

• Canadian RPB (Radiation Protection Bureau) suggested to remove the “P” (proceed) feature
  – AECL reduced times an operator could hit “P” to three times

• Also an independent turntable interlock should be installed
  – AECL did not comply
Incidents

3. Yakima Valley Memorial Hospital, Yakima, WA, December 1985
A patient treated for skin cancer had a reddening of the skin.

The tissue under the skin was damaged, causing pain.

The damage was surgically repaired.

The patient survived.

A malfunction of the Therac-25 was ruled out, since it was believed to be 10.000.000% more safe than before.
Incidents

4. East Texas Cancer Center, Tyler, TA, March 1986
A patient received electron therapy of cancer in upper back.

After entering the treatment data, the operator noticed that the beam type was set to “x” (x-ray) instead of “e” (electrons).
• The operator changed the setting ("x" to "e") and confirmed all other previously entered values with a quick series of carriage returns

• Then she hit “B” to begin
• The machine shut down, indicating an under-dosage
• She hit “P” to proceed

• On that day, the video and audio monitoring was out of operation
East Texas Cancer Center, March 1986

- After the first treatment, the patient felt a burn on his back
- He then got up, because he suspected something was wrong
- When the operator hit “P”, the patient's hand was in the beam
- “He felt his arm was shocked with electricity and his hand was leaving his body”
- Physicists suspected an electric shock
- Instead, the patient received a massive overdose
- The patient was in pain and partly paralyzed, died five months later from the overdose
- AECL engineers could not reproduce the malfunction
5. East Texas Cancer Center, Tyler, TA, April 1986 (same hospital, month later)
A patient was set up for electron therapy of skin cancer on the side of his face.

Again, the (same) operator, after entering the data, noticed an error in the mode ("x" instead of "e")

Again, she fixed the error and confirmed all other values by a series of carriage returns.

When the machine was turned on, she heard a loud noise over the (now working) intercom.

The patient reported he felt "fire" on his face.

The patient died three weeks later, from a high-dose radiation injury of parts of the brain.
The Software Bug

- A physisist at the Tyler clinic tried to reproduce the error
- It only occurred if the data entry was performed rapidly
- Software design & code not available
- Information about the bug were reproduced from lawsuit documents

- The problem was a race condition that only occurred if the data entry was performed fast enough
  - race condition: in concurrent programs, the result of a computation changes depending on the speed of the execution of constituent threads
Software Design

Figure from Nancy Leveson: Medical Devices: The Therac-25

controls turntable

process entered data
The Datent Process

Datent:

if mode/energy specified then
begin
  calculate table index
  repeat
  fetch parameter
  output parameter
  point to next parameter
  until all parameters set
  call Magnet
  if mode/energy changed then return
end
if data entry is complete then set Tphase to 3
if data entry is not complete then
  if reset command entered then set Tphase to 0
return

Magnet:

Set bending magnet flag
repeat
  Set next magnet
  call Ptime
  if mode/energy has changed, then exit
until all magnets are set
return

Ptime:

repeat
  if bending magnet flag is set then
    if editing taking place then
      if mode/energy has changed then exit
  until hysteresis delay has expired
Clear bending magnet flag
return
The Datent Process

Datent:

if mode/energy specified then
begin
    calculate table index
    repeat
        fetch parameter
        output parameter
        point to next parameter
    until all parameters set
    call Magnet
    if mode/energy changed then return
end

if data entry is complete then set Tphase to 3
if data entry is not complete then
    if reset command entered then set Tphase to 0
return

Magnet:

Set bending magnet flag
repeat
    Set next magnet
    call Ptime
    if mode/energy has changed, then exit
until all magnets are set
return

Ptime:

repeat
    if bending magnet flag is set then
        if editing taking place then
            if mode/energy has changed then exit
    until hysteresis delay has expired
Clear bending magnet flag
return

keyboard handler set mode and energy?

“started phase of setting bending magnets”

delay, ~8 sec.

abort if mode/energy values have changed

abort if mode/energy values have changed
The Datent Process

Datent:

if mode/energy specified then begin
  calculate table index
  repeat
    fetch parameter
    output parameter
    point to next parameter
  until all parameters set
  call Magnet
if mode/energy changed then return
end
if data entry is complete then set Tphase to 3
if data entry is not complete then
  if reset command entered then set Tphase to 0
return

Magnet:

Set bending magnet flag
repeat
  Set next magnet
  call Ptime
  if mode/energy has changed, then exit
until all magnets are set
return

Ptime:

called several times
repeat
  if bending magnet flag is set then
    if editing taking place then
      if mode/energy has changed then exit
    until hysteresis delay has expired
Clear bending magnet flag
return

Problem: does not abort if values are changed during subsequent calls of Ptime!

“ended phase of setting bending magnets”
6. Yakima Valley Memorial Hospital, Yakima, WA, January 1987
• Patient was set up for x-ray therapy of cancer in the chest

• The patient received an overdose, because the electron beam was switched on while the turntable was in the field light position

• The patient died three months later

• The software bug was an overflow of an 1-byte variable
  – The collimator setting was checked if the variable was non-zero
  – in rare cases, the variable could overflow “255+1 = 0”
After the Sixth Accident

• All Therac-25 systems were shut down

• A meeting took place with AECL and US and Canadian governmental representatives

• Changes were decided and implemented, among others:
  – Operators not allowed to restart the machine without re-entering all parameters
  – More meaningful error messages, reporting specific dose-rates
  – Hardware interlocks
  – Multiple software changes to handle hardware faults

• No accidents were reported after the changes were made
(Some) Lessons Learned

• Bad software design
• Unacceptable software engineering practices (testing, documentation, no defined development process)
• Unrealistic risk assessment, overconfidence in software
• Effects of concurrency were not taken into account
• Accidents usually happen for more than one reason
• Irresponsible reaction to accidents, inadequate investigation and reports
Overview

• Examples of distributed and interacting systems
• Accidents caused by software flaws in history

➡ What this lecture will be about
Looking at the Development Process

http://en.wikipedia.org/wiki/V-Model
Looking at the Development Process

Informal Requirements:
- correct?
- consistent?

Not what the stakeholder wanted → Redesign

automated tests? which specification?

ignore requirements
Improving the Development

formal requirements, adequate notation

incremental development, monitoring, …, updates, self-X, …

systematic (or partly automated) implementation

automated validation, test case generation

verification and validation

project definition
Improving the Development

- formal requirements, adequate notation
- incremental development, monitoring, ..., updates, self-X, ...
- systematic (or partly automated) implementation
- Problem with tests:
  1. Can only be performed relatively late, means long iterations if bugs are found
  2. "Testing can only show the presence of errors never their absence." (E. Dijkstra)

But:
They are nevertheless very important
Improving the Development

formal requirements, adequate notation

analyze a model of the system and its software

systematic (or partly automated) implementation

incremental development, monitoring, ..., updates, self-X, ...

Problem with tests:
1. Can only be performed relatively late, means long iterations if bugs are found
2. "Testing can only show the presence of errors never their absence." (E. Dijkstra)

But:
They are nevertheless very important
Improving the Development

focus of this lecture:

- formal requirements, adequate notation
- analyze a model of the system and its software
- systematic (or partly automated) implementation
- incremental development, monitoring, ..., updates, self-X, ...

- automated validation, test case generation
Model-Checking

Model

Specification

Model Checking

modify

false

ture

+ counter example (a possible sequence of events in the system that violates the specification)
Further Topics

• Scenario-Based Design
  – intuitive techniques for the design of interaction behavior

• Graph Transformation Systems
  – modeling and analyzing the reconfiguration behavior of dynamic systems

• Real-Time Systems
  – representation and efficient analysis of time aspects
Course Outline

- Introduction
- Modeling with Automata and State Charts
- Design Methodology
- LTL Model Checking
- CTL Model Checking
- Scenario-Based Design
- Graph Transformation Systems
- Real-Time Systems