Dependable Systems

Introduction

Dr. Peter Tröger
Dependable Systems Course

- Extended version of the „Dependable Systems“ course at Humboldt University (www.rok.informatik.hu-berlin.de)
- Recommended readings and slides on course home page
- 2 out of 3 successfully solved assignments as precondition for oral exam
  - Single solutions, no group work
  - (1) Component-based dependability analysis
  - (2) State-model based dependability analysis
  - (3) Dependability analysis for a real-world hardware / software infrastructure
Dependability

- **Umbrella term** for operational requirements on a system

  - IFIP WG 10.4: "[..] the trustworthiness of a computing system which allows reliance to be justifiably placed on the service it delivers [..]"

  - IEC IEV: "dependability (is) the collective term used to describe the availability performance and its influencing factors: reliability performance, maintainability performance and maintenance support performance"

  - Laprie: "Trustworthiness of a computer system such that reliance can be placed on the service it delivers to the user"

- Adds a third dimension to system quality

- General question: How to deal with unexpected events?

- In German: 'Verlässlichkeit' vs. 'Zuverlässigkeit'
## English vs. German

<table>
<thead>
<tr>
<th>Dependability</th>
<th>Verlässlichkeit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability</td>
<td>Zuverlässigkeit</td>
</tr>
<tr>
<td>Availability</td>
<td>Verfügbarkeit</td>
</tr>
<tr>
<td>Safety</td>
<td>Sicherheit</td>
</tr>
<tr>
<td>Security</td>
<td>Schutz</td>
</tr>
<tr>
<td>Maintainability</td>
<td>Wartbarkeit</td>
</tr>
<tr>
<td>Fault</td>
<td>Fehlerursache (Fehler)</td>
</tr>
<tr>
<td>Error</td>
<td>Fehlerzustand (Fehler)</td>
</tr>
<tr>
<td>Failure</td>
<td>Ausfall</td>
</tr>
<tr>
<td>Error Propagation</td>
<td>Fehlerausbreitung</td>
</tr>
<tr>
<td>Fault Prevention</td>
<td>Fehlerursachenverhinderung</td>
</tr>
<tr>
<td>Fault Removal</td>
<td>Fehlerursachenbeseitigung</td>
</tr>
<tr>
<td>Fault Tolerance</td>
<td>Fehlertoleranz</td>
</tr>
<tr>
<td>Fault Forecasting</td>
<td>Fehlervorhersage</td>
</tr>
</tbody>
</table>
Computers in Safety-Critical Systems

- Different levels of critical participation for a computer system
  - Information provisioning to human controller on request
  - Interpretation of data and presentation to the user
  - Issues command on behalf of the human controller
  - Replaces human controller
- Trend to realize critical systems with commercial-of-the-shelf components
  - Driven by budget cuts and performance advantage
  - Puts sole responsibility on software layer, in contrast to early hardware-only redundancy approaches
Dependable Systems Motivation

- Money
- Life saving
- Human users
- Harsh environments
- Complexity
Example: Therac 25

- Radiation therapy engine, involved in 6 accidents between 1985 and 1987
- Two operation modes: High-power spreaded beam, or low power focus beam
- Accident: High-power mode without spreader plate activated
  - Harmful operation not detected due to software flaws
  - Institutional issues: No independent software review, no risk analysis for failure modes, no test of hardware / software assembling
  - Engineering issues: No hardware interlock, software re-use from older models, no sensors for hardware check, arithmetic overflow in the check routines
  - No investigation of single modules
Example: Patriot Missile Launcher

- Mobile missile launcher, designed for a few hours of operation
- Used for Scud defense operation, never designed for it
- February 1991 - Battery in Dharan (Saudi Arabia) failed to intercept Scud missile, hit army barrack
  - Software aging problem in system’s weapon control computer
  - Target velocity and time demanded as real values, stored as 24-bit integer
  - Inaccurate tracking computation due to overlong operation (> 100 hours)
  - Modified software reached the base one day after the accident
Example: Ariane 5

- June 4 1996, self-destruction firework for $500 million
- Initiated because boosters were ripping from the rocket
  - Reasoned by on-board computer reaction on inertia system data
    - Was not data, but a diagnostic bit pattern due to an overflow error
    - Identical inertia backup system had shut down milliseconds before
- Inertia system from Ariane 4, numbers were never expected to get that high
  - 64-bit FP to 16-bit signed integer, data conversion in Ada was not protected
  - No test of full system due to budget limits
- System was running after lift off for no special purpose
  - Possibility for fast restart with brief hold in the countdown, only for Ariane 4
Example: Ariane 5

declare

v_veloc_sensor: float;     -- 64 bit floating point
h_veloc_sensor: float;
v_veloc_bias: integer;     -- 16 bit signed integer
h_veloc_bias: integer;

begin

sensor_get (v_veloc_sensor);
sensor_get (h_veloc_sensor);

v_veloc_bias := integer (v_veloc_sensor);
h_veloc_bias := integer (h_veloc_bias);

exception

  when others => use_irs1();

end;
Example: Mars Pathfinder

- Successful landing on Mars at July 4th, 1997

- Spontaneous resets after landing
  - VxWorks operating system, real-time scheduling of data streams and commands
  - Classical priority inversion - long-running medium-communication task, low-priority meteorological task, reset triggered by watchdog
  - Antennas performed too good, unexpected timing behavior in data transmission
  - Problem reproduced with identical Pathfinder on earth
  - Semaphore had priority inheritance protection, which was switched off for performance optimization
Unbounded Priority Inversion

- Pathfinder scenario

- T1: Periodically checks health of systems

- T2: Processes image data

- T3: Occasional test on equipment status

- T1 and T3 share a protected data structure

- T1 must indirectly wait for T2 suspend, which has no relation to its own purpose

- T1 timeout lead to full system reset and check
Example: Air France AF 447 (May 2009)

- Airbus A330 flight at 10,000m through turbulent weather conditions
- Ice crystals trigger malfunction in Pitot air pressure sensors
  - Stream of air normally used for speed computation
  - Flight speed instruments shut down, followed by auto pilot
  - Demanded manual angle and thrust level computation by pilots
  - Flight computer switches to emergency mode (‘alternate law 2‘)
    - Still operational (‘fly-by-wire‘), but different flight feeling
  - Pilots lost physical control over machine, restart of flight computer not successful
- Specification for sensors not feasible (written in 1947)
  - Sensor must be operational till -40 degrees / 9000m height
Importance of Dependability for Business

- Average costs per hour of downtime (Gartner 1998)
  - Brokerage operations in finance: $6.5 million
  - Credit card authorization: $2.6 million
  - Home catalog sales: $90,000
  - Airline reservation: $89,500
- 22-hour service outage of eBay on June 6th 1999
  - Interruption of around 2.3 million auctions
  - 9.2% stock value drop, $3-5 billion of lost revenues
  - Problems blamed on Sun server software
- Banking, telephone system, online games, cloud operations, ...
Example: New York Stock Exchange

• Credit Suisse trading desk, Nov 14th, 2007
  • Automated trading algorithm went mad
  • Reasoned by misplaced double click in the trading software UI
  • Hundreds of thousands of bogus order cancellations
  • DoS nature of fault caused NYSE to freeze up - $150,000 fine
Example: Amazon Cloud Outages

- 2006 - Elevated levels of authenticated requests from multiple users
  - Request volumes are monitored, but cryptographic overhead was not considered
  - Overload on authentication services - performs request processing and validation
  - S3 service (cloud storage service) became inaccessible, but is the only storage facility supported for Amazon-hosted virtual machines
  - Took 3.25 hours to solve the problem - live fixing, started at 3am
- No SLA promises ever, but trust in cloud-only hosted web sites dropped dramatically
- 2009 - 19 hours of outage for bitbucket.org
  - Code hosting service, relies on EC2 and block storage
  - UDP / TCP flooding attack influenced also access to storage facility
Hello,

A few days ago we sent you an email letting you know that we were working on recovering an inconsistent data snapshot of one or more of your Amazon EBS volumes. We are very sorry, but ultimately our efforts to manually recover your volume were unsuccessful. The hardware failed in such a way that we could not forensically restore the data.

What we were able to recover has been made available via a snapshot, although the data is in such a state that it may have little to no utility...

If you have no need for this snapshot, please delete it to avoid incurring storage charges.

We apologize for this volume loss and any impact to your business.

Sincerely,
Amazon Web Services, EBS Support
Example: IBM S/390 Outage Sources

• Examination of 16 managed mainframe systems over 6 months (Pfister, 1990)

• 92% planned outages, 8% unplanned outages
  • 32% database backup
  • 24% database reorganization
  • 14% system software maintenance
  • 12% network
  • 8% applications
  • 8% hardware
  • 2% other
Tradeoffs

Ultra Reliable Systems

Commercial Fault-Tolerant Systems

Massively Parallel/Distributed Systems

Availability

0.99999
0.9999
0.999
0.99
0.9

Throughput (MIPS)
Example: AT&T Electronic Switching System ESS1A

- Downtime for entire system not more than 2 hours over 40 years of life
  - System outage < 3 minutes / year
  - 100% availability from user perspective
- Goal achieved, two minutes of down time [Malek]
  - 24 sec hardware faults, 18 sec software problems, 36 sec procedural errors, 42 sec recovery problems
- Full duplication of all critical components
- Automated error detection on hardware and software level
  - Replication checks - duplex system with comparison on every cycle
  - Timing checks, coding checks, internal checks with comparators
Example: AT&T Electronic Switching System ESS1A

The architecture of the AT&T 3B20 Duplex system

(C) Malek
Example: Boeing 777 Flight Controller

- Fly-by-wire system, requirements
  - Probability $1 \times 10^{-10}$ for degradation from 'MIN-OP configuration'
  - 'Never give up' redundancy management strategy
- Mean time between maintenance actions: 25,000 operating hours
- TMR for all hardware resources, fail-passive electronics
- Candidate architectures were evaluated according to many trade aspects
Example: Boeing 777 Flight Controller (Buus et al.)

Fig. 2. System architecture design drivers and selection process.
Example: VAX Hardware Redundancy

- VMS operating system on VAX hardware, since mid-1970‘s, outgrow of PDP-11
- VAXft in the early 1990‘s
IBM zEnterprise

Manage Risk with System z Resiliency

*Availability built in across the system*

- Single System z
  - Where mean time between failure is measured in decades
- Multiple System z
  - Designed for *application* availability of 99.999%
- Multiple Data Centers
  - Industry-leading solution for disaster recovery

- Avoiding the cost of downtime
- Ensuring access to critical applications
- Maintaining productivity of users
- Open to clients 24/7
Sources of Outages
Pre z9 - Hrs/Year/Syst-

Impact of Outage

Unscheduled Outages

Scheduled Outages

Planned Outages

Preplanning requirements

Power & Thermal Management

Temperature = Silicon Reliability Worst Enemy
Wearout = Mechanical Components Reliability Worst Enemy.

© IBM
**IBM zEnterprise**

**Extreme Virtualization in zEnterprise**
- Built into the architecture not an “add on” feature

*System z*
- Deploy virtual servers in seconds
- Highly **granular** resource sharing (<1%)  
- Add physical resources without taking system down, scale out to **1000s** of virtual servers
- **Do more with less:**  
  - More virtual servers per core,  
  - Share more physical resources across servers
- Extensive built-in facilities for virtual server **life-cycle management**
- **Hardware-enforced isolation**

**Distributed Platforms**
- Limited per-core virtual server scalability  
- Physical server sprawl is needed to scale  
- Operational complexity increases as virtual server images grow  
- VMware tools only support VMware hypervisor (ESX)

---

The potential performance impact of the Linux server farm is isolated from the other Logical partitions (LPAR)

Each LPAR is one Virtual Mainframe

**z/VM** – 100s of virtual servers – Shared Memory

**LPAR** – Up to 60 Logical partitions

**PR/SM**

Hypervisor in firmware

Hardware support: 10% of circuits are used for virtualization
Dependability Examples

• A segmentation fault can lead to a system failure.

• A faulty controller in an RAID array lowers the reliability of the storage facility.

• The measurable availability of a web server depends on the fault tolerance capabilities in all its components.

• A database fault creates an erroneous state in the authentication component, which can lead to a system failure influencing the system confidentiality.

• The integrity of electronic voting systems depends on the complete absence of failures in the voting and the calculation procedures.

• Fault forecasting can support the maintainability of a system.

• Testing can only prove the presence of faults, not their absence.

• Our service level agreement guarantees an availability of 99.99%.
Consequences of Failures [Knight]

• Human injury or loss of life
• Damage to the environment
• Damage to or loss of equipment
• Damage to or loss of data
• Financial loss by theft
• Financial loss through production of useless or defective products
• Financial loss through reduced capacity for production or service
• Loss of business reputation
• Loss of customer base
• Loss of jobs
Dependability Concerns [Knight]

• Do I know all available techniques to prevent some damage in / by my system?

• I don‘t know how the system is used, so how can I decide for an appropriate dependability mechanism?

• How do I get an accurate and complete set of term definitions to communicate reliably with the other engineers?

• What is a technology to tackle all weak points in my system architecture, not only the weakest or the obvious ones?

• How can I measure the effectiveness of my dependability techniques?

• Software is special ...
  • Definition of the required function is harder than in other engineering disciplines
  • Software needs to take action when hardware fails
  • Software may have to operate on platforms whose design was influenced by dependability goals
Course Topics

• Definitions and metrics
  • Fault, error, failure, fault models, fault tolerance, fault forecasting, ...
  • Reliability, availability, maintainability, error mitigation, damage confinement, ...
  • Reliability engineering, reliability testing, MTTF, MTBF, failure rate, ...

• Fault tolerance patterns
  • N-out-of-M, voting, heartbeat, ...

• Analytical evaluation
  • Reliability block diagrams, fault tree analysis, ...
  • Petri nets, Markov chains
  • Root cause analysis, risk analysis and assessment
Course Topics

• Hardware-Implemented Dependability
  • Failure rates, testing, hardware redundancy approaches

• Software-Implemented Dependability
  • Fault-tolerant programming: simplex, recovery blocks, checkpointing, roll-back, ...
  • Testing, software metrics
  • Fault-tolerant distributed systems: Transactions, consensus, clocks, ...

• Advanced approaches
  • Autonomic computing, rejuvenation, online failure prediction, performability

• Case studies