TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Department of computer Engineering
ati.ttu.ee

IAF0030

Arvutitehnika erikursus I

Süsteemide usaldusväärsus ja veakindlus
Dependability and fault tolerance

Gert Jervan

Tallinn University of Technology
Department of Computer Engineering
Estonia

General Information

- Contents:
  Dependability and fault tolerance
  www.pld.ttu.ee/IAF0030
- Lecturer & Examiner:
  Gert Jervan
  IT-229 620 2261
  gerje@pld.ttu.ee
  www.pld.ttu.ee/~gerje

Course Plan

- 16 occasions, á 2,5 hours
  Tuesdays 9:00-11:30
- 8 Lectures (maybe more)
  (every Tuesday, except February 24)
- Case Studies
  - Topic presentation
  - 20 min. presentation of the final report
  - Written report (6 pages, using predefined template)
- Exam

Grading

- Case study presentation – 30%
- Case study report – 30%

Prerequisites

- Exam – 40%

Total: 2,5 points

Reading

- **Various papers (on the course homepage)**

www.pld.ttu.ee/IAF0030

- Textbooks:
  - Safety-critical Computer Systems,
    Neil Storey, Addison Wesley, 1996.
    An introductory text which provides overview of safety related aspects and methods in computer systems development.
  - Reliability Engineering: Theory and Practice. 5th Revised edition, Alessandro Birolini, Springer, 2007
    This book shows how to build in, evaluate, and demonstrate reliability & availability of components, equipment, systems. It presents the state-of-the-art of reliability engineering, both in theory and practice
- Web pages, incident reports

Case Studies

- Topic categories:
  - Accident analysis
  - System safety analysis
  - Literature survey
  - Something else (implementation, tool study, etc.) – requires prior ack.

Literature and topics on the webpage

Case Studies

- Some examples (2007):
  - Redundant Arrays of Inexpensive Disks (RAID)
  - Fly-by-wire
  - Ariane V
  - London Ambulance Service Computer Aided Despatch System – Failure 1992
  - THERAC-25
  - Fault-tolerant features of modern processors
  - Triple-Triple Redundant 777 Primary Flight Computer
  - Redundancy Management Technique for Space Shuttle Computers
  - Cambridge'i ülikooli raamatupidamissüsteemi CAPSA
  - CAN (Controller Area Network) protocol

Case Studies

- Topic selection:
  - March 10 (via e-mail)
- Draft of the report (incl. introductory presentation of the topic):
  - April 7
- Presentations:
  - April 28 – May 19
- Final Report:
  - One week before exam
  - The best reports will be published in A&A (selected topics only)

Course Overview

- Reliability: increasing concern
  - Historical
    - High reliability in computers was needed in critical applications: space missions, telephone switching, process control etc.
  - Contemporary
    - Extraordinary dependence on computers: on-line banking, commerce, cars, planes, communications etc.
    - Hardware is increasingly more fault-prone (complexity, technology, environment)
    - Software is increasingly more complex
    - Things simply will not work without special reliability measures

Software complexity is a challenge

- Exponential increase in software complexity
- In some areas code size is doubling every 9 months [ST Microelectronics, Medea Workshop, Fall 2003]
- *... > 70% of the development cost for complex systems such as automotive electronics and communication systems are due to software development* [A. Sangiovanni-Vincentelli, 1999]

Rob van Ommering, COPA Tutorial, as cited by: Gerrit Müller: Opportunities and challenges in embedded systems, Eindhoven Embedded Systems Institute, 2004

Course Overview

- To get an insight into the broad area of system safety
- We cover techniques for high availability, fault tolerance, monitoring, detection, diagnosis, and confinement of failure, ways to improve availability through fast recovery and graceful service degradation, and techniques for using redundancy and replication.
- We also discuss the utopia of flawless software, the impact of scale on availability, ways to cope with human operator error, and metrics for evaluating dependability.

Contents

- Fault tolerance
- System reliability
- Hardware redundancy
- Error detection techniques
- Coding techniques
- Processor-level detection and recovery
- Disk arrays
- Checkpointing and recovery
- Software fault tolerance
- Testing distributed real-time systems
- ...

Course Outline (Preliminary)

- Feb 3: Introduction
- Feb 10: Risks, Hazards, Risk Analysis, Hazard Analysis
- Feb 17: Safety, Design practices, Testing (sw)
- March 3: Testing (sw and systems)
- March 10: Redundancy (hw & sw)
- March 17: Redundancy (information, time, environment)
- March 24: Enemies of dependability
- March 31: Verification, Validation
- April 7: Presentation of the case study topics
- April 14, April 21: Individual work with case studies (no lectures)
- April 28 - May 19: Case study presentations

Lecture Outline

- **Historical perspective and famous incidents/accidents**
- **Basic terminology**

Murphy's Law

- "If something can go wrong, it will go wrong"

  Major Edward A. Murphy, Jr.
  US Air Force, 1949

- "Every component than can be installed backward, eventually will be"

Genesis Space Capsule

- $260 million Genesis capsule was collecting samples of the solar wind over 3 years period
- Crashed in Sept 2004 due to the failure of the parachutes
- Reason: the deceleration sensors — the accelerometers — were all installed backwards. The craft's autopilot never got a clue that it had hit an atmosphere and that hard ground was just ahead.

Mars Orbiter

- One of the Mars Orbiter probes crashed into the planet in 1999.
- It did turn out that engineers who built the Mars Climate Orbiter had provided a data table in "pound-force" rather than newtons, the metric measure of force.
- NASA flight controllers at the Jet Propulsion Laboratory in Pasadena, Calif., had used the faulty table for their navigation calculations during the long coast from Earth to Mars.

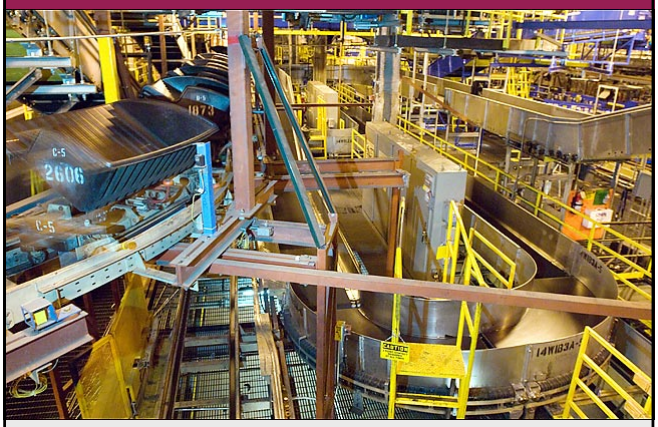
Lockheed Martin Titan 4

- In 1998, a LockMart Titan 4 booster carrying a $1 billion LockMart Vortex-class spy satellite pitched sideways and exploded 40 seconds after liftoff from Cape Canaveral, Fla.
- Reason: frayed wiring that apparently had not been inspected. The guidance systems were without power for a fraction of a second.

A Titan 4 rocket explodes shortly after takeoff in August 1998.

Therac-25

- Therac-25:
  - the most serious computer-related accidents to date (at least nonmilitary and admitted)
  - machine for radiation therapy (treating cancer)
  - between June 1985 and January 1987 (at least) six patients received severe overdoses (two died shortly afterward, two might have died but died because of cancer, the other two had permanent disabilities)
  - scanning magnets are used to spread the beam and vary the beam energy
  - dual-mode: electron beams for surface tumors, X-ray for deep tumors



Denver Airport

- Denver International Airport, Colorado: intelligent luggage transportation system with 4000 "Telecars", 35km rails, controlled by a network of 100 computers with 5000 sensors, 400 radio antennas, and 56 barcode readers.
  Price: $186 million (BAE Automated Systems).
- Due to SW problems about one year delay which costs $1.1 million per day (1993).
- Abondoned in 2005 to save $1 million per month on maintenance

Lecture Outline

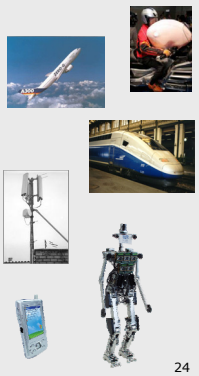
- **Historical perspective and famous incidents/accidents**
- **Basic terminology**

Embedded Systems

- Computing systems are everywhere
- Most of us think of "desktop" computers
  - PC's
  - Laptops
  - Mainframes
  - Servers
- But there's another type of computing system
  - Far more common...

Embedded Systems, cont.

- Embedded computing systems
  - Computing systems embedded within electronic devices
  - Hard to define. Nearly any computing system other than a desktop computer
  - Billions of units produced yearly, versus millions of desktop units
  - Perhaps 50 per household and per automobile



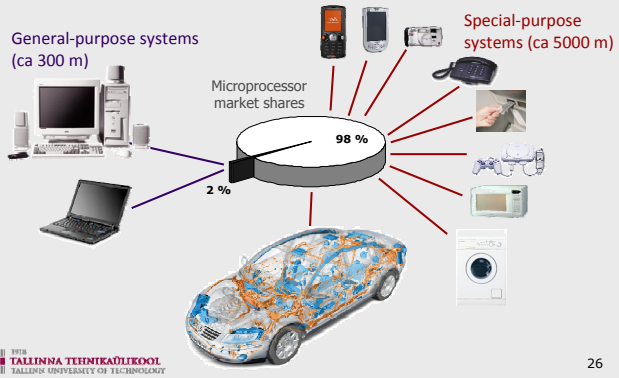
A "Short List" of Embedded Systems

Anti-lock brakes
Auto-focus cameras
Automatic teller machines
Automatic toll systems
Automatic transmission
Avionic systems
Battery chargers
Camcorders
Cell phones
Cell-phone base stations
Cordless phones
Cruise control
Curbside check-in systems
Digital cameras
Disk drives
Electronic card readers
Electronic instruments
Electronic toys/games
Factory control
Fax machines
Fingerprint identifiers
Home security systems
Life-support systems
Medical testing systems

Modems
MPEG decoders
Network cards
Network switches/routers
On-board navigation
Pagers
Photocopiers
Point-of-sale systems
Portable video games
Printers
Satellite phones
Scanners
Smart ovens/dishwashers
Speech recognizers
Stereo systems
Teleconferencing systems
Televisions
Temperature controllers
Theft tracking systems
TV set-top boxes
VCR's, DVD players
Video game consoles
Video phones
Washers and dryers

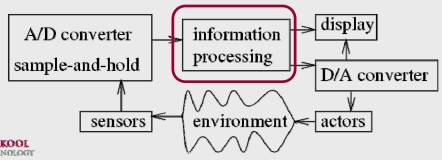
Our daily lives depend on embedded systems

General-Purpose vs. Special-Purpose



What is an Embedded System?

- Definition
  - an **embedded system** special-purpose computer system, part of a larger system which it controls.
- Notes
  - A computer is used in such devices primarily as a means to simplify the system design and to provide flexibility.
  - Often the user of the device is not even aware that a computer is present.



Characteristics of Embedded Systems

- Single-functioned
  - Dedicated to perform a single function
- Complex functionality
  - Often have to run sophisticated algorithms or multiple algorithms.
    - Cell phone, laser printer.
- Tightly-constrained
  - Low cost, low power, small, fast, etc.
- Reactive and real-time
  - Continually reacts to changes in the system's environment
  - Must compute certain results in real-time without delay
- Safety-critical
  - Must not endanger human life and the environment

Real-Time Systems

- **Time**
  - The correctness of the system behavior depends not only on the logical results of the computations, but also on the *time* at which these results are produced.
- **Real**
  - The reaction to the outside events must occur *during* their evolution. The system time must be measured using the same time scale used for measuring the time in the controlled environment.

Hard vs. Soft Real-Time

- Definitions
  - A real-time task is said to be **hard** if missing its deadline may cause catastrophic consequences on the environment under control.
  - A real-time task is said to be **soft** if meeting its deadline is desirable for performance reasons, but missing its deadline does not cause serious damage to the environment and does not jeopardize correct system behaviour.
- Definition
  - A real-time system that is able to handle hard real-time tasks is called a **hard real-time system**.

Hard vs. soft, cont.

- Examples of hard activities
  - Sensory data acquisition
  - Detection of critical conditions
  - Actuator serving
  - Low-level control of critical system components
  - Planning sensory-motor actions that tightly interact with the environment
- Examples of soft activities
  - The command interpreter of the user interface
  - Handling input data from the keyboard
  - Displaying messages on the screen
  - Representation of system state variables
  - Graphical activities
  - Saving report data

Functional vs. Non-Functional Requirements

- Functional requirements
  - output as a function of input
- Non-functional requirements:
  - **Time** required to compute output
  - **Reliability, availability, integrity, maintainability, dependability**
  - Size, weight, power consumption, etc.

Fault Tolerance

- A fault-tolerant system is one that can continue to correctly perform its specified tasks in the presence of failures:
  - hardware
  - software
  - user errors
  - environmental, input, ...
- Fault tolerance is the attribute that enables a system to achieve fault tolerant operation.

Basic Concepts

- *Fault Tolerance* is closely related to the notion of "Dependability". This is characterized under a number of headings:
  - *Availability* – the system is ready to be used immediately.
  - *Reliability* – the system can run continuously without failure.
  - *Safety* – if a system fails, nothing catastrophic will happen.
  - *Maintainability* – when a system fails, it can be repaired easily and quickly (and, sometimes, without its users noticing the failure).

Faults, Errors & Failures

- Fault: a defect within the system or a situation that can lead to the failure
- Error: manifestation of the fault – an unexpected behavior
- Failure: system not performing its intended function

Fault → Error → Failure

Fault Examples

- Bit flips in hardware due to cosmic radiation
  - A person on an airplane over the Atlantic at 35,000 ft working on a laptop with 256 Mbytes (2 Gbits) of memory. At this altitude, the SER of 600 FITs per megabit becomes 100,000 FITs per megabit, resulting in a potential error every five hours.
  - 1 FIT (failures in time), is the number of failures in 1 billion device-operation hours. A measurement of 1000 FITs corresponds to a MTTF (mean time to failure) of approximately 114 years.

Fault Examples

- ✓ Year 2000 bug
- ✓ Loose wire
- ✓ Aircraft retracting its landing gear while on ground
- ✓ Effects in time:
  - Permanent
  - Transient
  - Intermittent

Permanent

- ✓ A permanent fault or failure is one which is stable and continuous.
- ✓ Permanent hardware failures require some component to be replaced or repaired.
- ✓ An example of a permanent fault would be a VLSI chip with a manufacturing defect, causing one input pin to be stuck high (stuck-at-1).

Transient

- ✓ A transient fault is one which results from a temporary environmental condition.
- ✓ For example, a voltage spike might cause a sensor to report an incorrect value for a few milliseconds before reporting correctly.

Transient faults

- Happen for a short time
- **Corruptions of data, miscalculation in logic**
- Do not cause a permanent damage of circuits
- Causes are outside system boundaries

Radiation

Electromagnetic interference (EMI)

Lightning storms

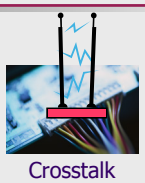
Intermittent

- ✓ An intermittent fault is one which only manifests occasionally, due to unstable hardware or certain system states.
- ✓ A loose contact on a connector will often cause an intermittent fault.
- ✓ Intermittent electrical faults, as a rule, are notoriously difficult to detect. Typically, whenever the fault doctor shows up, the system works fine.

Intermittent faults

- Manifest similar as transient faults
- Happen repeatedly
- Causes are inside system boundaries

Internal EMI

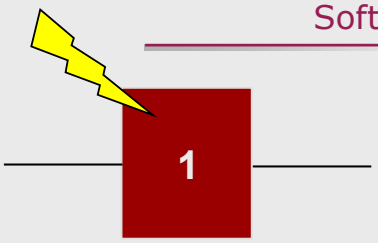


Crosstalk

Power supply fluctuations

Software errors (Heisenbugs)

Soft Errors



- Transient bit-flip (soft memory error)
  - Random event
  - Corrupts the value but not the cell
  - Can be corrected (in contrast to hard errors caused by faults in the hardware itself)
  - Happen continuously during system lifetime (*i.e.*, can not be screened by burn-in tests)

Sources

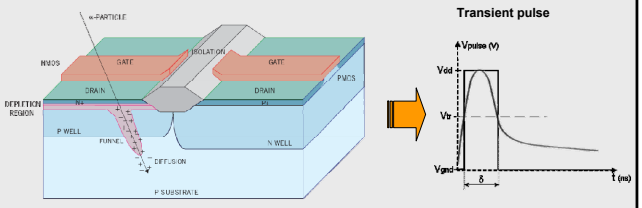
- First traced to alpha particlce emissions from chip packaging materials
  - Most sources removed (pure materials, different designs, shielding)
- Today's main problem: cosmic radiation
  - Cosmic particles from deep space (actually 5th- or 6th-hand collision particles)
    - At ground level ca 95% neutrons, 5% protons
  - Radioactive material in manufacturing process

Sources (cont.)

- Four main sources:
  - Low-energy alpha particles
  - High-energy cosmic particles
  - Thermal neutrons
  - Poor system design

SER type	Source	Mechanism	Trend
Alpha	Thorium and uranium contamination in-mold compound, silicon, or lead bumps	2- to 9-MeV alpha particle creating electron-hole funnel traveling 25 microns in silicon	Exponential increase with scaling
Cosmic	Intergalactic sources modulated by solar flares	High-energy neutrons/protons (10 MeV to 1 GeV) colliding with silicon nuclei	Decrease in failures in time per megabit
Thermal neutron	Boron present in BPSG25-meV neutrons	Collision with B10 in BPSG	Highest, always dominates if present

Soft Errors



The electric field in the depletion region directly generates electron-hole pairs in its wake, causing the charges to drift so that the transistor sees a current disturbance

Evidence of Cosmic Ray Strikes

- Documented strikes in large servers found in error logs
  - Normand, "Single Event Upset at Ground Level," IEEE Transactions on Nuclear Science, Vol. 43, No. 6, December 1996.
- Sun Microsystems, 2000 (R. Baumann, Workshop talk)
  - Cosmic ray strikes on L2 cache with defective error protection
    - caused Sun's flagship servers to suddenly and mysteriously crash!
  - Companies affected
    - Baby Bell (Atlanta), America Online, Ebay, & dozens of other corporations
    - Verisign moved to IBM Unix servers (for the most part)

Failure Classification

- Domain/Nature
  - Value failure
  - Timing failure
- Perception
  - Consistent failure
  - Inconsistent failure
- Effect
  - Benign failure
  - Malign/catastrophic failure
- Frequency
  - Single failure
  - Repeated failure

Failures

- **Crash** Failure: After an error has been detected, the component stops silently.
- **Omission** Failure: Sometimes a result is missing; when result is available, it is correct.
- **Consistent** Failure: If there are multiple receivers, all see the same erroneous result.
- **Byzantine** (Malicious, Asymmetric) Failure: Different receivers see differing results.

Failures (cont.)

- **Timing** Failure: A server's response lies outside the specified time interval.
- **Response** Failure: The server's response is incorrect (value of the response is wrong, server deviates from the correct flow of control).
- **Arbitrary** Failure: A server may produce arbitrary responses at arbitrary times.

Fault Handling

- Fault avoidance: eliminate problem sources
  - Remove defects: Testing and debugging
  - Robust design: reduce probability of defects
  - Minimize environmental stress: Radiation shielding etc

Impossible to avoid faults completely

- Fault tolerance: add redundancy to mask effect
  - Additional resources needed (more later)
  - Examples:
    - Error correction coding, voting and masking, checksums, ...
    - Backup storage, replication, ...
    - Spare tire, etc

Fault Tolerance

- **Fault detection** is the process of recognizing that a fault has occurred. Fault detection is often required before any recovery procedure can be initiated. The techniques include error detection codes, self-checking/failsafe logic, watchdog timers, and others.
- **Fault location** is the process of determining where a fault has occurred so that an appropriate recovery can be initiated.

Fault Tolerance (cont.)

- **Fault containment** is the process of isolating a fault and preventing the effects of that fault from propagating throughout the system.
- **Fault recovery** is the process of remaining operational or regaining operational status via reconfiguration even in the presence of faults. A few basic approaches are fault masking, retry, and rollback.

Definitions

- Failure rate (λ):
  - Average frequency with which something fails.

$$\frac{6\ failures}{7502\ hrs} = 0.0007998\ failures/hr = 799.8 \times 10^{-6}\ failures/hr$$

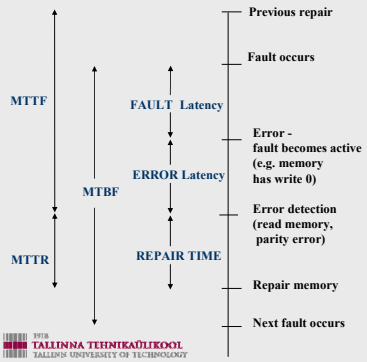
- Mean time to failure (MTTF):
  - Average time between failures

$$MTTF = \frac{1}{\lambda}$$

Dependability

- Property of a computing system which allows reliance to be justifiably placed on the service it delivers
- Dependability = reliability + availability + safety + security + ...
- Reliability → continuity of correct service
- Availability → readiness of usage
- Safety → no catastrophic consequences
- Security → prevention of unauthorized access

Dependability Concepts



Reliability:
a measure of the continuous delivery of service;
R(t) is the probability that the system survives (does not fail) throughout [0, t];
expected value: *MTTF(Mean Time To Failure)*

Maintainability:
a measure of the service interruption
M(t) is the probability that the system will be repaired within a time less than t;
expected value: *MTTR (Mean Time To Repair)*

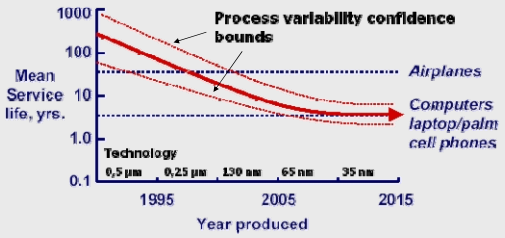
Availability:
a measure of the service delivery with respect to the alternation of the delivery and interruptions
A(t) is the probability that the system delivers a proper (conforming to specification)service at a given time t.
expected value: *EA = MTTF / (MTTF + MTTR)*

Safety:
a measure of the time to catastrophic failure
S(t) is the probability that no catastrophic failures occur during [0, t];
expected value:
MTTCF(Mean Time To Catastrophic Failure)

Reliability

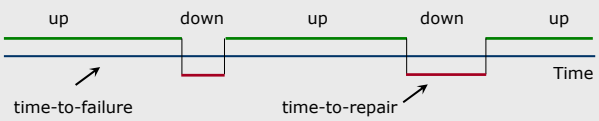
- A measure of an it performing its intended function satisfactorily for a prescribed time and under given environment conditions.
- Probability that system will survive to time *t*
  - In aerospace industry the requirement is that failure probability is 10^{-9} (one failure over 10^9 hours (114 000 years) of operation)
- Time To Failure (TTF)
- Mean Time To Failure (MTTF)

Commercial Chip Reliability Estimation



*Extrapolated from ITRS roadmap, MRQW-2002, Bernstein

Availability



$$Availabili\ ty = \frac{MTTF}{MTTF + MTTR}$$

- Availability:
  - Probability that system is operational at time *t*
- High availability:
  - MTTF → infinity (high reliability)
  - MTTR → zero (fast recovery)

Maintainability

- *M(t)* is the probability that a failed system will be restored within a specified period of time *t*.
- Restoration process:
  - locating problem, e.g. via diagnostics
  - physically repairing system
  - bringing system back to its operational condition

Graceful Degradation

- The ability of system to automatically decrease its level of performance to compensate for hardware failure and software errors.

The Myth of the Nines

Nines	Availability	Downtime per year	Downtime per week	Example
2 nines	99%	3.65 days	1.7 hours	General web site
3 nines	99.9%	8.75 hours	10.1 min	E-commerce site
4 nines	99.99%	52.5 min	1.0 min	Enterprise mail server
5 nines	99.999%	5.25 min	6.0 s	Telephone system
6 nines	99.9999%	31.5 s	0.6 s	Carrier-grade phone switch

Historical Evaluation

- Mean Time Between Failures:

$$MTBF = MTTR + MTTF$$

  - ENIAC. MTBF: 7 minutes (18000 vacum tubes)
    - ENIAC → TX-2 interactive computer (MIT) → web
  - F-8 Crusader – first fly-by-wire
    - MD-11
    - A320 family
  - Patriot missile defence system
    - 1/3 sec in 100 hours, targeting error: 600 m
    - Needed reboot after 8 hours, was learned in hard way...

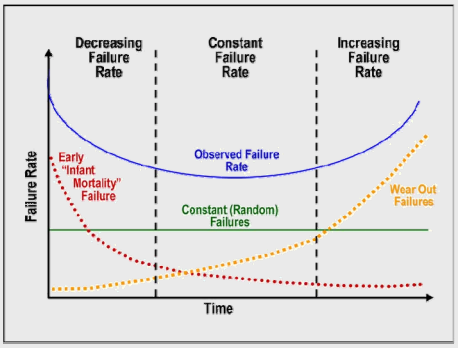
Ultra-Reliable Systems

- Airbus A320 family fly-by-wire system:
  - computer controls all actuators
  - no control rods, cables in the middle
  - 5 central flight control computers
  - different systems used
    - Thomson CSF => 68010
    - SFENA => 80186
  - software for both hardware written by different software houses
  - all error checking & debugging performed separately
  - computer allows pilot to fly craft up to certain limits (flight envelope)
    - beyond: computer takes over

Hardware and Environment Failures

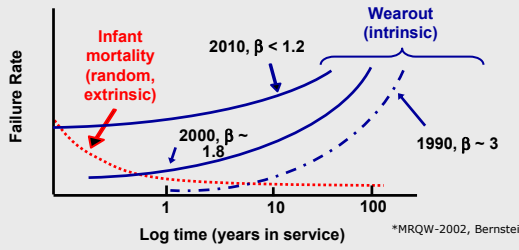
- Moving parts, high speed, low tolerance, high complexity: disks, tape drives/libraries
- Lowest MTBF found in fans and power supplies
- Often fans fail gradually → subtle, sporadic failures in CPU, memory, backplane
- Environment: power, cooling, dehumidifying, cables, fire, collapsing racks, ventilation, earthquakes, ...

Bathtub Curve



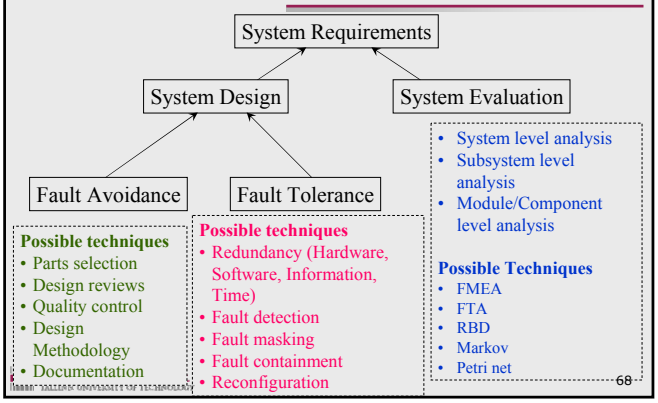
Device Reliability Trends

As technology progresses, wearout failures become statistically indistinguishable from infant mortality failures with the same wearout drivers.



*MRQW-2002, Bernstein

System Design & Evaluation Top-Level View



Safety

- ✓ Attribute of a system which either operates correctly or fails in a safe manner
- ✓ Freedom from expose to danger, or exemption from hurt, injury or loss.
- ✓ "Fail-safe": traffic lights start to blink yellow
- ✓ Degrees of safety
- ✓ Closely related to risk

Risk

- ✓ A combination of the likelihood af an accident and the severity of the potential consequences
- ✓ The harm that can result if a threat is actualised
- ✓ Acceptable/tolerable risk: **The Ford Pinto case** (1968)

BENEFITS
Savings: 180 burn deaths, 180 serious burn injuries, 2,100 burned vehicles.
Unit Cost: $200,000 per death, $67,000 per injury, $700 per vehicle.
Total Benefit: 180 X ($200,000) + 180 X ($67,000) + $2,100 X ($700) = $49.5 million.

COSTS
Sales: 11 million cars, 1.5 million light trucks.
Unit Cost: $11 per car, $11 per truck.
Total Cost: 11,000,000 X ($11) + 1,500,000 X ($11) = $137 million.

System Safety & Hazards

- ✓ Safety:
  - achieved by anticipating accidents and eliminating their causes
- ✓ Hazards are potential causes of accidents
  - Conditions in a system which together with other factors in the environment inevitably cause accidents

TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

Department of computer Engineering
ati.ttu.ee

Questions?

Gert Jervan

Tallinn University of Technology
Department of Computer Engineering
Estonia