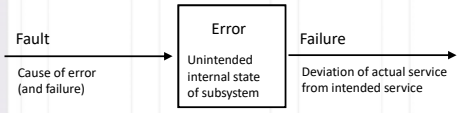
IAF0530 (MSc, 5 ECTS)
IAS0530 (MSc, 6 ECTS)
IAF9530 (PhD, 6 ECTS)

Dependability and fault tolerance

Gert Jervan
Department of Computer Systems
Tallinn University of Technology (TTÜ)

Threats: Faults, Errors & Failures



2

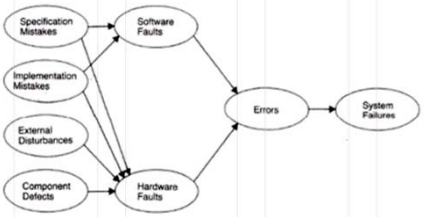
Three-universe model

- Physical universe: where the faults occur
  - Physical entities: semiconductor devices, mechanical elements, displays, printers, power supplies
  - A fault is a physical defect or alteration of some component in the physical universe
- Informational universe: where the error occurs
  - Units of information: bits, data words
  - An error has occurred when some unit of information becomes incorrect
- External (user's universe): where failures occur
  - User sees the effects of faults and errors
  - The failure is any deviation from the desired or expected behavior

3

Causes of faults

- Problems at any stages of the design process can result in faults within the system.



4

Causes of faults, cont.

- Specification mistakes
  - Incorrect algorithms, architectures, hardware or software design specifications
    - Example: the designer of a digital circuit incorrectly specified the timing characteristics of some of the circuit's components
- Implementation mistakes
  - Implementation: process of turning the hardware and software designs into physical hardware and actual code
  - Poor design, poor component selection, poor construction, software coding mistakes
    - Examples: software coding error, a printed circuit board is constructed such that adjacent lines of a circuit are shorted together

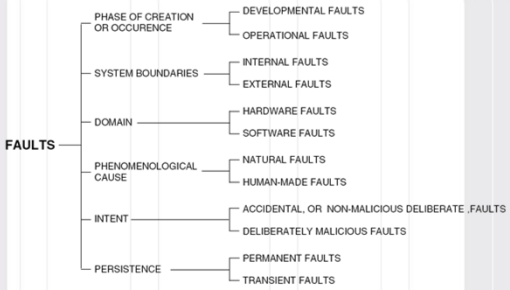
5

Causes of faults, cont.

- Component defects
  - Manufacturing imperfections, random device defects, component wear-out
  - Most commonly considered causes of faults
    - Examples: bonds breaking within the circuit, corrosion of the metal
- External disturbance
  - Radiation, electromagnetic interference, operator mistakes, environmental extremes, battle damage
    - Example: lightning

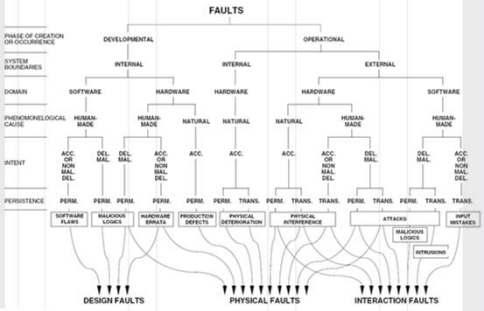
6

Elementary fault classes



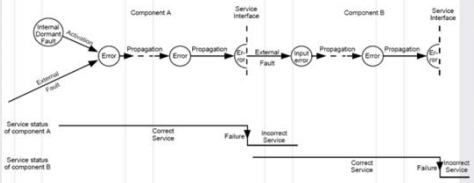
7

Classification of faults



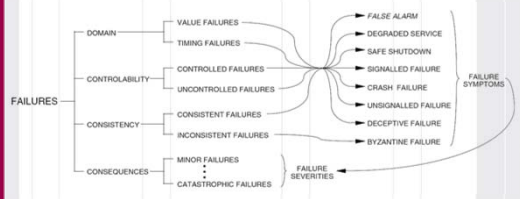
8

The pathology of failure



9

Failure modes



10

Failure modes, cont.

- Failure domain
  - Value failures : incorrect value delivered at interface
  - Timing failures : right result at the wrong time (usually late)
- Failure consistency
  - Consistent failures : all nodes see the same, possibly wrong, result
  - Inconsistent failures : different nodes see different results
- Failure consequences
  - Benign failures : essentially loss of utility of the system
  - Malign failures : significantly more than loss of utility of the system; catastrophic, e.g. airplane crash
- Failure oftenness (failure frequency and persistency)
  - Permanent failure : system ceases operation until it is repaired
  - Transient failure : system continues to operate
    - Frequently occurring transient failures are called intermittent

11

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.

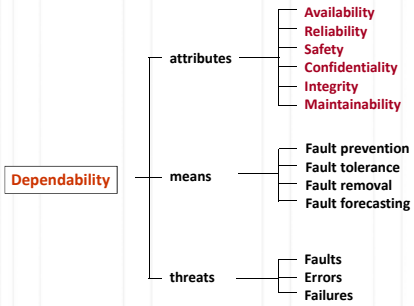
12

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.

13

Dependability



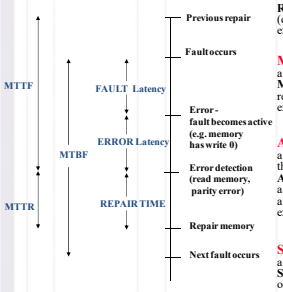
14

Dependability attributes

- **Availability**: readiness for correct service
- **Reliability**: continuity of correct service
- **Safety**: absence of catastrophic consequences on the user(s) and the environment
- **Confidentiality**: absence of unauthorized disclosure of information
- **Integrity**: absence of improper system alterations
- **Maintainability**: ability to undergo, modifications, and repairs
- **Security**: the concurrent existence of (a) availability for authorized users only, (b) confidentiality, and (c) integrity with 'improper' taken as meaning 'unauthorized'.

15

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)***

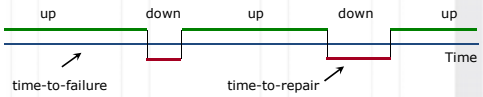
Safety:
a measure of the time to catastrophic failure
S(t) is the probability that no catastrophic failures occur during [0, t];
expected value:
MTTCF(Mean Time To Catastrophic Failure)

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 109 hours (114 000 years) of operation)
- Time To Failure (TTF)
- Mean Time To Failure (MTTF)

17

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)

18

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

19

Graceful Degradation

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

20

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 network switch

21

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, 375 hours → 750 hours (IBM AP-101)
    - 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...

22

Ultra-Reliable Systems

- Airbus A320 family fly-by-wire system (originaal):
  - computer controls all actuators
  - no control rods, cables in the middle
  - 7 central flight control computers
    - 3 Motorola 68000
    - 2 Intel 80C86
    - 2 Intel 80C286
  - software for hardware written by different software houses (C, ASM, dedicated one, specifically developed)
  - all error checking & debugging performed separately
  - computer allows pilot to fly craft up to certain limits (flight envelope)
    - beyond: computer takes over

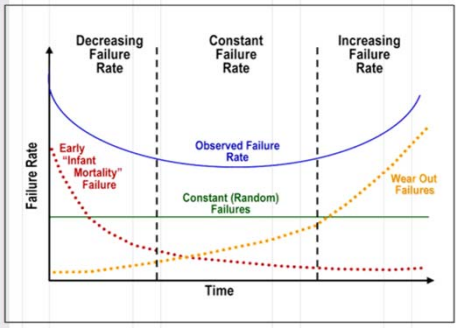
23

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, ...

24

Bathtub Curve



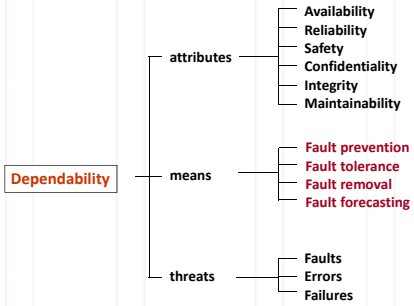
25

Device Reliability Trends

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

© Gert Jervan

Dependability



© Gert Jervan

27

Fault Handling

- Fault avoidance/prevention: 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

© Gert Jervan

28

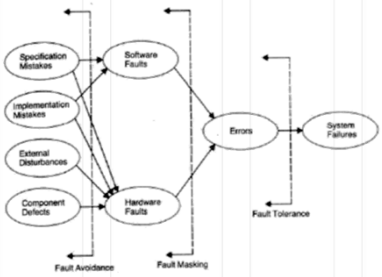
Means to achieve dependability

- **Fault-prevention:** how to prevent, by construction, fault occurrence.
- **Fault-tolerance:** how to provide, by redundancy, service complying with the specification in spite of faults having occurred or occurring.
- **Fault-removal:** how to minimize, by verification and validation, the presence of latent faults.
- **Fault-forecasting:** how to minimize, by evaluation, the presence, the creation and the consequences of faults.

© Gert Jervan

29

Means to achieve dependability, cont.



© Gert Jervan

30

Fault prevention

- Eliminate problem sources
  - Remove defects: Testing and debugging
  - Robust design: reduce probability of defects
  - Minimize environmental stress: Radiation shielding etc
  - Attained by quality control techniques
    - Software
      - Structured/object oriented programming, information hiding, modularization
    - Hardware
      - Rigorous design rules, shielding, radiation hardening, "foolproof" packaging
- Note:
  - Malicious faults can also be prevented; Example: firewalls

Impossible to avoid faults completely!

31

Fault tolerance

- **Fault tolerance** is the ability of a system to continue to perform its functions (deliver correct service), even when one or more components have failed.
  - **Masking**: the use of sufficient redundancy may allow recovery without explicit error detection (information encoding).
  - **Reconfiguration**: eliminating a faulty entity from a system and restoring the system to some operational condition or state.
    - Error **detection**: recognizing that an error has occurred
    - Error **location**: determining which module produced the error
    - Error **containment**: preventing the errors from propagating
    - Error **recovery**: regaining operational status

32

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.

33

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.

34

Definitions

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

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

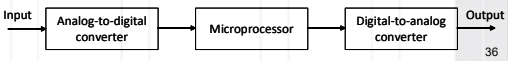
- Mean time to failure (MTTF):
  - Average time between failures

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

35

The concept of redundancy

- Definition
  - **Redundancy** is the addition of information, resources, or time beyond what is needed for normal system operation.
- Digital filter example
  - Software redundancy: lines of software to perform a validity checks
  - Hardware redundancy : if more memory needed for the software checks
  - Time redundancy: each filter calculation performed twice to detect faults
  - Information redundancy: output using with a simple parity bit



36

Error detection

- Two ways to detect errors:
  - a priori knowledge about intended state
  - comparing results of two redundant computational channels
- Notes
  - Errors can happen in the **value domain** and/or in the **time domain**.
  - The probability that an error is detected, provided it is present, is called the **error detection coverage**.
  - The time interval between the start of an error and the detection of an error is the **error detection latency**.

37

A Priori Knowledge flexibility vs. error-detection coverage

- Syntactic knowledge about code space
  - Parity bits, CRC
- Assertions and acceptance tests
  - Valid data values, properties of the controlled object
    - Development of physical processes, plausibility of data sets
- Activation patterns of computation
  - Regularity in execution pattern, e.g., frequency of updates
    - Limited by the update frequency and clock synchronisation
    - Event every second, on the second -> detect missing event
- Worst case execution time of tasks
  - Must be known to calculate real-time schedules
  - A priory information about the execution of a task can be used for detecting task errors

38

Redundant Computations

Type of Redundancy	Implementation	Type of Detected Errors
Time redundancy	Same software executed on the same hardware during two different time-intervals	Errors caused by transient physical faults in hardware with a duration less than one execution time slot
Hardware redundancy	The same software executes on two independent hardware channels	Errors caused by transient and permanent physical hardware errors
Diverse software on the same hardware	Different software versions are executed on the same hardware during two different time intervals	Errors caused by independent software faults and transient physical faults in the hardware with a duration less than one execution time slot
Diverse software on diverse hardware	Two different versions of software are executed on two independent hardware channels	Errors caused by independent software faults and by transient and permanent physical hardware faults

39

Recovery

- Definition
  - **Recovery** transforms a system state that contains one or more errors and (possibly) faults into a state without detected errors and faults that can be activated again.
- Consists of
  - Error handling
    - **Rollback**: returning to a saved state (checkpoint)
    - **Compensation**: enough redundancy to eliminate the error
    - **Rollforward**: the state without errors is a new state
  - Fault handling
    - **Fault diagnosis**: identifies the cause of errors, location and type
    - **Fault isolation**: physical or logical exclusion of the faulty components
    - **System reconfiguration**: switches in spares or re-assigns tasks
    - **System reinitialization**: checks, updates and records the new configuration

40

Fault removal

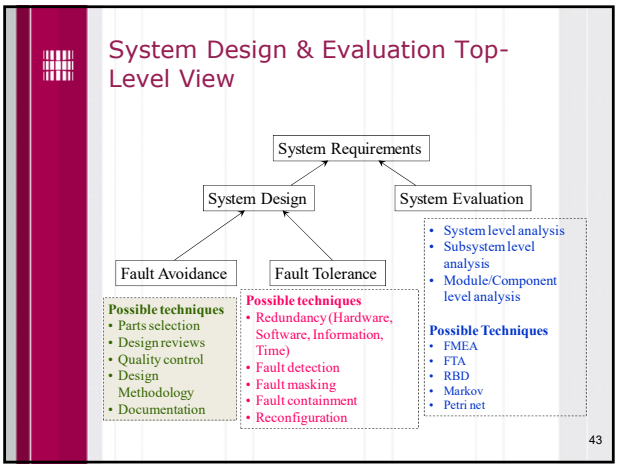
- **Verification**: "Are we building the system right?"
  - Static: does not exercise the system
    - Static analysis: inspections, walkthroughs, model checking
  - Dynamic
    - Symbolic execution: inputs are symbolic
    - Testing: actual inputs
  - Fault injection
- **Validation**: "Are we building the right system?"
  - Checking the specification

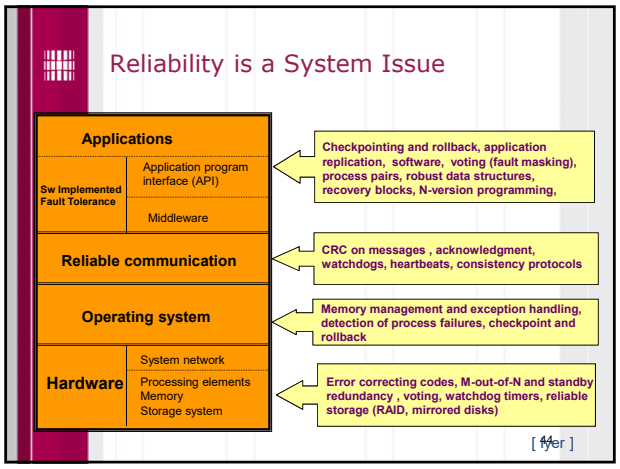
41

Fault Forecasting

- Evaluation of the system behavior with respect to fault occurrence
  - **Qualitative** evaluation
    - Identifies, classifies, ranks the failure modes or the event combinations that lead to system failures
    - Example methods: Failure mode and effect analysis, fault-tree analysis
  - **Quantitative** evaluation
    - Evaluates in terms of probabilities the extent to which some of the dependability are satisfied (measures dependability)
    - Example methods: Markov chains, reliability block diagrams

42


System Design & Evaluation Top-Level View
System Requirements
System Design
System Evaluation
Fault Avoidance
Fault Tolerance
Possible techniques
• Parts selection
• Design reviews
• Quality control
• Design Methodology
• Documentation
Possible techniques
• Redundancy (Hardware, Software, Information, Time)
• Fault detection
• Fault masking
• Fault containment
• Reconfiguration
• System level analysis
• Subsystem level analysis
• Module/Component level analysis
Possible Techniques
• FMEA
• FTA
• RBD
• Markov
• Petri net
43


Reliability is a System Issue
Applications
Sw Implemented Fault Tolerance
Application program interface (API)
Middleware
Reliable communication
Operating system
Hardware
System network
Processing elements
Memory
Storage system
Checkpointing and rollback, application replication, software, voting (fault masking), process pairs, robust data structures, recovery blocks, N-version programming,
CRC on messages , acknowledgment, watchdogs, heartbeats, consistency protocols
Memory management and exception handling, detection of process failures, checkpoint and rollback
Error correcting codes, M-out-of-N and standby redundancy , voting, watchdog timers, reliable storage (RAID, mirrored disks)
[Iyer]
44