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

Loeng 3
Risks, Safety, Fault Tolerance, Software Testing

gert.jervan@pld.ttu.ee

Tallinn University of Technology
Department of Computer Engineering
Estonia

TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

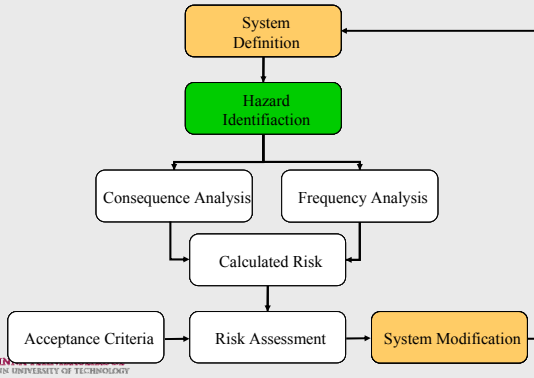
Department of computer Engineering
ati.ttu.ee

Risk Analysis

Risk Analysis

- The purpose
  - Associate risk with given hazards
    - Consequence of malfunction - severity
    - Probability of malfunction – frequency
  - Ensure nature of risks is well understood
  - Ensure safety targets can be set and evaluated
- Techniques
  - Quantitative
  - Qualitative, risk classification
  - Integrity classification
  - Safety Integrity Levels (SILs)
  - ALARP
- Standards
  - IEC 1508, IEC 61508

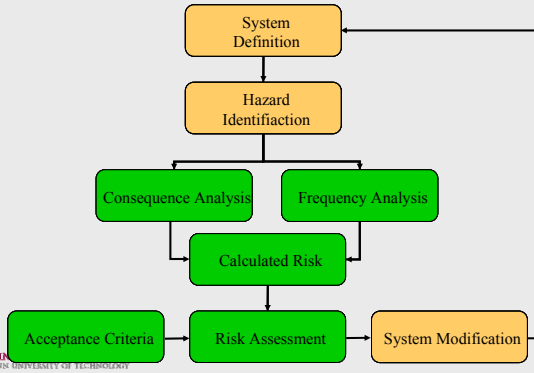
Hazard and Risk Analysis Process



Flashback

- A Hazard is a system state that could lead to:
  - Loss of life
  - Loss of property
  - Release of energy
  - Release of dangerous materials
- Hazards are the *states* we have to avoid
- An accident is a loss event:
  - System in hazard state, ***and***
  - Change in the operating environment
- Classification
  - Severity
  - Nature

Hazard and Risk Analysis Process



Introduction

- Risk is associated with every hazard
  - Hazard is a potential danger
    - i.e. possibility of being struck by lightning
  - Associated risk
- *Accident is an unintended event or sequence of events that causes death, injury, environmental or material damage*

Storey 1996

Introduction

- Hazard analysis identifies accident scenarios: sequences of events that lead to an accident
- *Risk is a combination of the **severity** of a specified hazardous event with its **probability** of occurence over a specified **duration***
  - Qualitative or quantitative

Risk Calculation

- Quantify probability/frequency of occurence:
  - number of events per hour/year of operation
  - number of events per lifetime
  - number of failures on demand
- Ex 1:
  - Failure of a particular component results in explosion that could kill 100 people. Estimate that component will fail once every 10,000 years

    1 failure per 10,000 years = 0.0001 failures per year

    Risk = penalty x (probability per year)

    = 100 x (0.0001)

    = 0.01 deaths per year

Risk Calculation

- Ex 2:
  - Country with population of 50,000,000
  - Approx. 25 people are each year killed by lightning i.e. $25/50{,}000{,}000=5\times10^{-7}$
  - Risk:
    - every individual has proabability of 5×10^{-7} to be killed by lightning at any given year
    - Population is exposed to risk of 5×10^{-7} deaths per person year
- Qualitative:
  - intolerable, undesirable, tolerable

Levels of Fatal Risk

Risk	Chance per million
Risk of being killed by a falling aircraft	0.02 cpm
Risk of death by lightening	0.1 cpm
Risk of being killed by an insect or snake bite	0.1 cpm
Risk of death in a fire caused by a cooking appliance in the home	1 cpm
Risk of death in an accident at work in the very safest parts of industry	10 cpm
General risk of death in a traffic accident	100 cpm
Risk of death in high risk groups within relatively risky industries such as mining	1,000 cpm
Risk of fatality from smoking 20 cigarettes per day	5,000 cpm
Risk of death from 5 hours of solo rock climbing every weekend	10,000 cpm

The Need for Safety Targets

- Learning from mistakes is not longer acceptable
  - Disaster, review, recommendation
- Probability estimates
  - Are coarse
  - Meaning depends on duration, low/high demand, but often stated without units
- Need rigour and guidance for safety related systems
  - Standards (HSE, IEC)
  - Ensure risk reduction, not cost reduction
  - For risk assessment
  - For evaluation of designs

Quantitative Risk Assessment

- How it works
  - Predict frequency of hardware failures
  - Compare with tolerable risk target
  - If not satisfied, modify the design
- Example
  - The probability that airbag fails when activated
  - The frequency of the interconnecting switch failing per lifetime
- Even if target met by random hardware failure
  - Hardware could have embedded software, potential for systemic failure
  - Engineer's judgment called for in IEC 61508
    (IEC 61508 – Functional Safety – www.iec.ch)

Quantitative risk assessment

- Quantify probability/frequency of occurence:
  - number of events per hour/year of operation
  - number of events per lifetime
  - number of failures on demand
- Example:
  - Failure of a particular component results in explosion that could kill 100 people. Estimate that component will fail once every 10,000 years
    1 failure per 10,000 years = 0.0001 failures per year
    Risk = penalty x (probability per year)
    = 100 x (0.0001)
    = 0.01 deaths per year

Qualitative Risk Assessment

- When cannot estimate the probability
- How it works
  - Classify risk into risk classes
  - Define tolerable/intolerable risks
  - Define tolerable/intolerable frequencies
  - Set standards and processes for evaluation and minimization of risks
- Example
  - Catastrophic, multiple deaths
  - Critical, single death
  - Marginal, single severe injury
  - Negligible, single minor injury
- Aims to deal with systemic failures

Risk Management

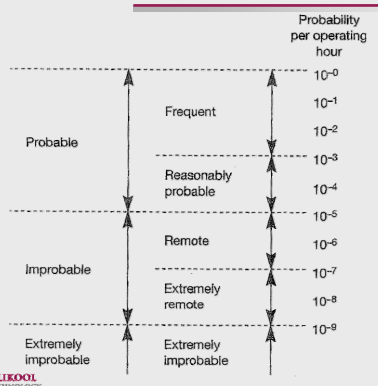
Risk		Probability				
		Very High	**High**	**Medium**	**Low**	**Very Low**
Conse-quence	**Very High**	Very High	Very High	High	High	Medium
	High	Very High	High	Medium	Medium	Low
	Medium	High	Medium	Medium	Low	Low
	Low	High	Medium	Low	Low	Very Low
	Very Low	Medium	Low	Low	Very Low	Very Low

Risk Ranking table

Hazard Severity Categories for Civil Aircraft

Category	*Definition*
Catastrophic	Failure condition which would prevent continued safe flight and landing
Hazardous	Failure conditions which would reduce the capability of the aircraft or the ability of the crew to cope with adverse operating conditions, to the extent that there would be: (1) a large reduction in safety margins or functional capabilities (2) physical distress or higher workload such that the flight crew could not be relied on to perform their tasks accurately or completely (3) adverse effects on occupants, including serious or potentially fatal injuries to a small number of those occupants
Major	Failure conditions which would reduce the capability of the aircraft or the ability of the crew to cope with adverse operating conditions to the extent that there would be, for example, a significant reduction in safety margins or functional capabilities, a significant increase in crew workload or in conditions impairing crew efficiency, or discomfort to occupants, possibly including injuries
Minor	Failure conditions which would not significantly reduce aircraft safety, and which would involve crew actions that are well within their capabilities. Minor failure conditions may include, for example, a slight reduction in safety margins or functional capabilities, a slight increase in crew workload, such as routine flight plan changes, or some inconvenience to occupants
No effect	Failure conditions which do not affect the operational capability of the aircraft or increase crew workload

Hazard Probability Classes for Aircraft Systems



Risk Management Advice

- Identify risks and track them
  - Avoid "unknown" risks at all costs!
- Approaches to risk
  - Mitigate, i.e. perform risk reduction
    - E.g. solve the problem, obtain insurance, etc
  - Avoid
    - Use a less risky approach - not always possible
  - Accept
    - Decide that expected cost is not worth reducing further
    - Often sensible choice
- Ignore
  - Proceed ahead blindly – uninformed acceptance

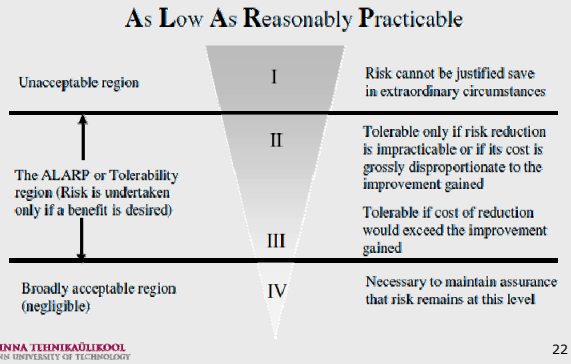
Acceptability of Risk

- Acceptability of risk is a complex issue involving
  - social factors, e.g., value of life and limb
  - legal factors, e.g., responsibility of risk
  - economic factors, e.g., cost of risk reduction
- Ideally these tasks are performed by policy makers, not engineers!
- Engineers provide the information on which such complex decisions can be made
- At beginning of project, accurate estimates of risks and costs are difficult to achieve

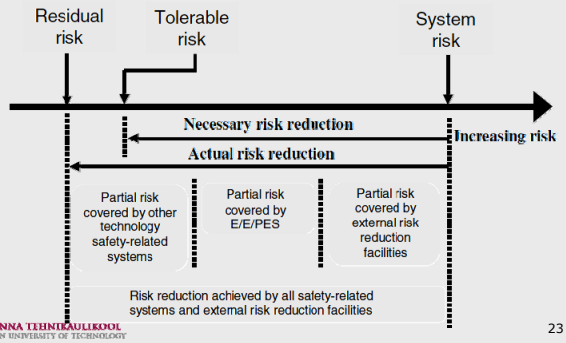
Acceptability of risk

- Ethical considerations
  - Determining risk and its acceptability involves moral judgement
  - Society's view not determined by logical rules
  - Perception that accidents involving large numbers of deaths are perceived as more serious than smaller accidents, though they may occur less frequently

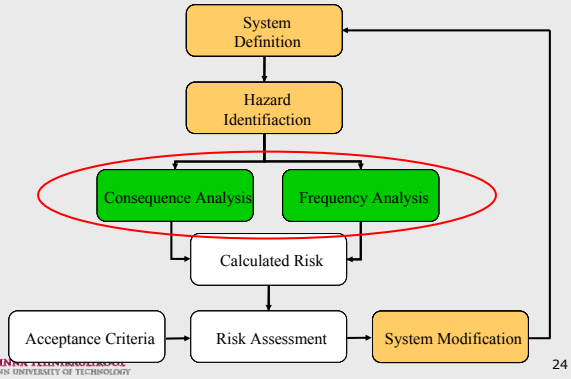
Risk Reduction - ALARP



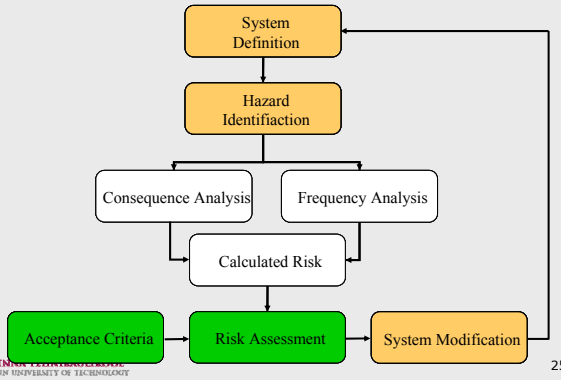
Risk Reduction



Hazard and Risk Analysis Process



Hazard and Risk Analysis Process



Safety Requirements

- Once hazards are identified and assessed, safety requirements are generated to mitigate the risk
- Requirements may be
  - primary: prevent initiation of hazard
    - eliminate hazard
    - reduce hazard
  - secondary: control initiation of hazard
    - detect and protect
    - warn
- Safety requirements form basis for subsequent development

Safety Integrity

- Safety integrity, defined by
  - Likelihood of a safety-related system satisfactorily performing the required safety functions under all stated conditions within a stated period of time
  - Hardware integrity, relating to random faults
  - Systematic integrity, relating to dangerous systematic faults
- Expressed
  - Quantitatively, or
  - As Safety Integrity Levels (SILs)
- Standards, IEC 1508, 61508
  - Define target failure rates for each level
  - Define processes to manage design & development
- Aims to deal with systemic failures

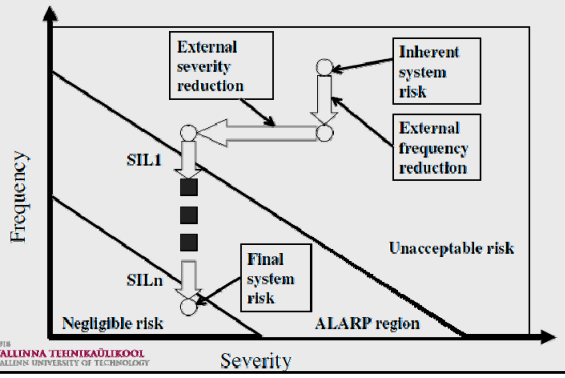
Safety Integrity Levels (SILs)

- Tolerable failure frequency are often characterised by Safety Integrity Levels rather than likelihoods
  - SILs are a qualitative measure of the required protection against failure
- SILs are assigned to the safety requirements in accordance with target risk reduction
- Once defined, SILs are used to determine what methods and techniques should be applied (or not applied) in order to achieve the required integrity level
- Point of translation from failure frequencies to SILs may vary

Automotive SIL

- Uncontrollable (SIL 4), critical failure
  - No driver expected to recover (e.g. both brakes fail), extremely severe outcomes (multiple crash)
- Difficult to control (SIL 3), critical failure
  - Good driver can recover (e.g. one brake works, severe outcomes (fatal crash)
- Debilitating (SIL 2)
  - Ordinary driver can recover most of the time, usually no severe outcome
- Distracting (SIL 1)
  - Operational limitations, but minor problem
- Nuisance (SIL 0)
  - Safety is not an issue, customer satisfaction is

Risk & SILs



IEC 61508 Standard

- New main standard for software safety
- Can be tailored to different domains (automotive, chemical, etc)
- Comprehensive
- Includes SILs, including failure rates
- Covers recommended techniques

- IEC = International Electrotechnical Commission

- E/E/PES = electrical/electronic/programmable electronic safety related systems

Safety-Integrity Table of IEC 61508

Safety Integrity Level	Low demand mode of operation (Average probability of failure to perform its design function on demand)	
4	$\geq 10^{-5}$ to $< 10^{-4}$	(> 99.99 % reliable)
3	$\geq 10^{-4}$ to $< 10^{-3}$	(> 99.9 % reliable)
2	$\geq 10^{-3}$ to $< 10^{-2}$	(> 99% reliable)
1	$\geq 10^{-2}$ to $< 10^{-1}$	(> 90% reliable)

Safety Integrity Level	High demand mode or continuous mode of operation (Probability of dangerous failure per hour)
4	$\geq 10^{-9}$ to $< 10^{-8}$
3	$\geq 10^{-8}$ to $< 10^{-7}$
2	$\geq 10^{-7}$ to $< 10^{-6}$
1	$\geq 10^{-6}$ to $< 10^{-5}$

- The higher the SIL, the harder to meet the standard
- High demand for e.g. car brakes, critical boundary SIL 3
- Low demand for e.g. airbag, critical boundary is SIL 3, one failure in 1000 activations

SILs

- SILs 3 and 4 are critical
- SIL activities at lower levels may be needed
- SIL 1
  - Relatively easy to achieve, if ISO 9001 practices apply,
- SIL 2
  - Not dramatically harder than SIL 1, but involves more review and test, and hence cost
- SIL 3
  - Substantial increment of effort and cost
- SIL 4
  - Includes state of the art practices such as formal methods and verification, cost extremely high

Techniques and Measures

Clause 7.7 : Software Safety Validation					
TECHNIQUE/MEASURE	**Ref**	**SIL1**	**SIL2**	**SIL3**	**SIL4**
1. Probabilistic Testing	B.47	--	R	R	HR
2. Simulation/Modelling	D.6	R	R	HR	HR
3. Functional and Black-Box Testing	D.3	HR	HR	HR	HR
NOTE: One or more of these techniques shall be selected to satisfy the safety integrity level being used.					

- Implementing the recommended techniques and measures should result in software of the associated integrity level.
- For example, if the software was required to be validated to be of Integrity level 3, Simulation and Modelling are Highly Recommended Practices, as is Functional and Black-Box Testing.

Detailed Techniques and Measures

- Related to certain entries in these tables are additional, more detailed sets of recommendations structured in the same manner. These address techniques and measures for:
  - Design and Coding Standards
  - Dynamic analysis and testing
  - Approaches to functional or black-box testing
  - Hazard Analysis
  - Choice of programming language
  - Modelling
  - Performance testing
  - Semi-formal methods
  - Static analysis
  - Modular approaches

Modeling

D.6 : Modelling Referenced by Clauses 7.6					
TECHNIQUE/MEASURE	**Ref**	**SIL1**	**SIL2**	**SIL3**	**SIL4**
1. Data Flow Diagrams	B.12	R	R	R	R
2. Finite State Machines	B.29	--	HR	HR	HR
3. Formal Methods	B.30	--	R	R	HR
4. Performance Modelling	B.45	R	R	R	HR
5. Time Petri Nets	B.64	--	HR	HR	HR
6. Prototyping/Animation	B.49	R	R	R	R
7. Structure Diagrams	B.59	R	R	R	HR
NOTE: One or more of the above techniques should be used.					

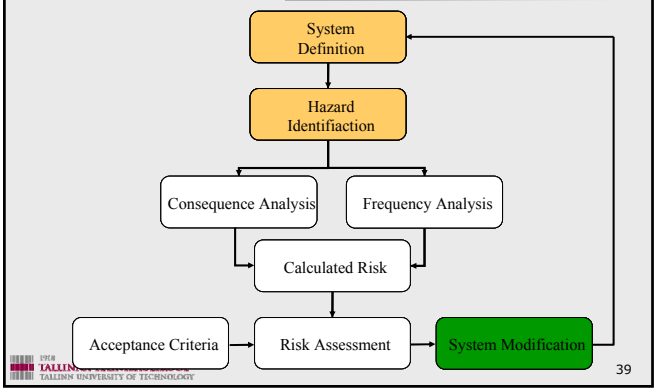
SILs

- What does it all mean?
  - SIL 4 system should have a duration of about 10^9 hours between critical failures
  - If established SIL 4 needed, used all the techniques…
  - But there is no measurement that the results actually achieves the target
  - Standard assumes that you are competent in all methods and apply everything possible
  - Except that these may be insufficient or not affordable

The Engineering Council's Code of Practice on Risk Issues

1	Professional responsibility	Exercise reasonable professional skill and care
2	Law	Know about and comply with the law
3	Conduct	Act in accordance with the codes of conduct
4	Approach	Take a systematic approach to risk issues
5	Judgement	Use professional judgement and experience
6	Communication	Communicate within your organization
7	Management	Contribute effectively to corporate risk management
8	Evaluation	Assess the risk implications of alternatives
9	Professional development	Keep up to date by seeking education and training
10	Public awareness	Encourage public understanding of risk issues

Hazard and Risk Analysis Process



Risk Reduction Procedures

- Four main categories of risk reduction strategies, given in the order that they should be applied:
  - Hazard Elimination
  - Hazard Reduction
  - Hazard Control
  - Damage Limitation
- Only an approximate categorisation, since many strategies belong in more than one category

Hazard Ellimination

- Before considering safety devices, attempt to eliminate hazards altogether
  - use of different materials, e.g., non-toxic
  - use of different process, e.g., endothermic reaction
  - use of simple design
  - reduction of inventory, e.g., stockpiles in Bhopal
  - segregation, e.g., no level crossings
  - eliminate human errors, e.g., for assembly of system use colour coded connections

Design Principles

- Familiar
  - use tried and trusted technologies, materials techniques
- Simple
  - testable (including controllable and observable)
  - portable (no use of sole manufacturer components compiler dependent features)
  - understandable (behaviour can easily be from implementation)
  - deterministic (use of resources is not random)
  - predictable (use of resources can be predicted)
  - minimal (extra features not provided)

Design Principles (cont.)

- Structured design techniques
  - defined notation for describing behaviour
  - identification of system boundary and environment
  - problem decomposition
  - ease of review
- Design standards
  - limit complexity
  - increase modularity
- Implementation standards
  - presentation and naming conventions
  - semantic and syntactic restrictions in software

Classes of System Failure

- Random (physical) failures
  - due to physical faults
  - e.g., wear-out, aging, corrosion
  - can be assigned quantitative failure probabilities
- Systematic (design) failures
  - due to faults in design and/or requirements
  - inevitably due to human error
  - usually measured by integrity levels
- Operator failures
  - due to human error
  - mix of random and systematic failures

Nature of Random Failures

- Arise from random events generated during operation or manufacture
- Governed by the laws of physics and cannot be eliminated
- Modes of failure are limited and can be anticipated
- Failures occur independently in different components
- Failure rates are often predictable by statistical methods
- Sometimes exhibit graceful degradation
- Treatment is well understood

Treating Random Failures

- Random failures cannot be eliminated and must be reduced or controlled
- Random failures can be mitigated by:
  - predicting failure modes and rates of components
  - applying redundancy to achieve overall reliability
  - performing preventative maintenance to replace components before faults arise
  - executing on-line or off-line diagnostic checks

Nature of Systematic Failures

- Ultimately caused by human error during development, installation or maintenance
- Appear transient and random since they are triggered under unusual, random circumstances
- Systematic and will occur again if the required circumstances arise
- Failures of different components are *not* independent
- Difficult to predict mode of failure since the possible deviations in behaviour are large
- Difficult to predict the likelihood of occurrence

Treating Systematic Failures

- In theory, design failures can be eliminated
- In practice, perfect design may be too costly
- Focus the effort on critical areas
  - identify safety requirements using hazard analysis
  - assess risk in system and operational context
- Eliminate or reduce errors using quality development processes
  - verify compliance with safety requirements
  - integrate and test against safety requirements

Design Faults

- Design faults are much more difficult to deal with than random (degradation) faults because:
  - They are hard to anticipate
  - Their effects are hard to predict
  - Component failure semantics tend to be undefined
- This makes all forms difficult to tolerate, especially software faults

Common Design Faults

- All forms of software:
  - System software
  - Application software
  - Embedded software (firmware)
- All forms of computing hardware:
  - Hardware design faults now dominate
  - Degradation faults used to dominate
- Power supply systems
- Component interconnection wiring

Design Diversity

- Idea:
  - Design faults are "aspects" of design
  - Different designs, different faults
  - Produce multiple designs—independent level.
  - Operate in parallel at execution time
- Applies to all types of design fault
- Can be configured using many system architectures, like NMR, TMR, etc.

Hazard Reduction

- Reduce the likelihood of hazards
- Use of barriers, physical or logical
  - Lock-ins
  - Lock-outs
  - Interlocks
- Failure minimization
  - Redundancy
  - Recovery

Forms of Redundancy

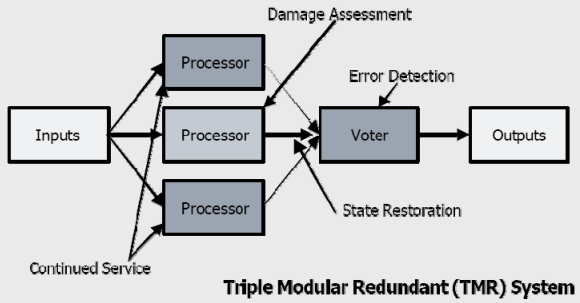
- Hardware redundancy
- Software redundancy
- Information redundancy
- Temporal (time) redundancy

- Design diversity, for hardware/software
  - Develop different implementations of the same hardware/software component
  - Called N-version programming
  - Then apply static or dynamic redundancy

Hardware Redundancy

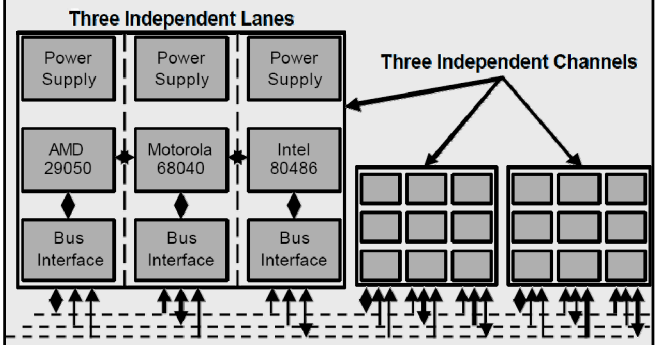
- Static redundancy
  - Component (at least) triplicated
    - Triple Modular Redundancy (TMR), N-Modular Redundancy (NMR)
  - Voting element used to remove effects of single failure
  - Loss Of Unit Implies:
    - Removal Or Containment
    - Service Provided By Those That Remain
- Dynamic redundancy
  - Component has a mirror that is invoked when fault occurs
  - Cold or Hot Standby, spares
  - Loss Of Unit Implies:
    - Removal Or Containment
    - Introducing Standby Unit

Fault Tolerant System Example



Risk of single-point failure

B777 Primary Flight Computer Architecture



N Modular Redundancy

- Independent development of modules
- This is what Boeing did with N = 3 for processors
- Operation:
  - Parallel—forward error recovery
  - Serial—backward error recovery
- In software with forward error recovery, referred to as N-version programming
- In software with backward error recovery, referred to as recovery block

B777 PFC CPUs

- Problem:
  - Processors often (essentially always) contain design faults, need to deal with them
  - 777 channel is a TMR system
- Three manufacturers, three designs
- Are these designs different?
- How would you measure the difference?
- What metric is there for design diversity?

Redundancy

- Software redundancy, e.g. N-version programming
- Information redundancy, e.g., checksums, cyclic redundancy codes, error correcting codes
- Hybrid redundancy

N-Version Programming

- NMR for software
- Practical issues:
  - Cost of development, team separation
  - Resources during execution
  - Different execution times for different versions
  - Different but similar output values
  - Different but valid output values (multiple correct solutions)

N-Version Programming

- Performance:
  - Assumed statistical independence
  - If not independent, then no lower bound
  - Common specification defects
  - Common implementation (design) faults
- Problem compounded by comparison checking during testing

Hybrid Redundancy

- N-S modular redundancy with "S" spares
- As members of the N-S fail, spares switched in
- Able to tolerate up to N-2 failures
- Spares may be unpowered:
  - Saves power
  - Unpowered units much more reliable than powered
  - Attention required to infant mortality
- Clearly applicable to:
  - Long-duration systems
  - Systems with no repair opportunity

Space Shuttle Computer System

- Uses combination of
  - Redundancy, fault detection and design diversity
- Hardware voting on sensors and actuators
- Five identical computers
  - During critical stages, four computers work in NMR with voting for fault detection
  - Fifth computer performs non-critical functions, e.g. comm.
- Fault tolerance
  - Tolerates failure of two computers
  - In case of third failure, crew/ground control decide which computer wins
  - Fifth computer can take over control, uses different software

Recovery

- Can reduce failures by recovering after error detected but before component or system failure occurs
- Recovery can only take place after detection of error
  - Backward recovery
  - Forward recovery

Error Detection

- Based on check that is independent of implementation of the system
  - coding - parity checks and checksums
  - reasonableness - range and invariants
  - reversal - calculate square of square root
  - diagnostic - hardware built-in tests
  - timing - timeouts or watchdogs

Error Detection (cont.)

- Timing of error detection important
  - early error detection can be used to prevent propagation
  - late error detection requires a check of the entire activity of system
- Checking may be in several forms
  - monitor, acting after a system function, checking outputs after production but before use
  - kernel, encapsulating (safety-critical) functions in a subsystem that allows all inputs to and outputs from the kernel to be checked

Backward Recovery

- Corrects errors through reversing previous operations
- Return system to a previous known safe state
- Allows retry
- Requires checkpoints or saved states (and the expenses involved with producing them)
- Rollback usually impossible with real-time system

Forward Recovery

- Corrects errors without reversing previous operations, finding safe (but possibly degraded) state for system
  - data repair, use redundancy in data to perform repairs
  - reconfiguration, use redundancy such as backup or alternate systems
  - coasting, continue operations ignoring (hopefully transient) errors
  - exception processing, only continue with selection of (safetycritical) functions
  - failsafe, achieve safe state and cease processing
    - use passive devices (e.g., deadman switch) instead of active devices (e.g., motor holding weight up)

Hazard Control

- Detect and control hazard before damage occurs
- Reduce the level or duration of the hazard
- Hazard control mechanisms include:
  - Limiting exposure: reduce the amount of time that a system is in an unsafe state (e.g. don't leave rocket in armed state)
  - Isolation and containment
  - Fail safe design

Damage Limitation

- In addition to eliminating hazards or employing safety devices, consider
  - warning devices
  - procedures
  - training
  - emergency planning
  - maintenance scheduling
  - protective measures

Architectural Design

- Suitable architectures may allow a high integrity system to be built from lower integrity components
  - combinations of components must implement a safety function independently
  - overall likelihood of failure should be the same or less
  - be wary of common failure causes
- Apportionment approaches can be quantitative and/or qualitative
  - quantitative: numerical calculations
  - qualitative: judgement or rules of thumb

TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

Department of computer Engineering
ati.ttu.ee

Fault Tolerance

Basics

- Computing systems are characterized by five fundamental properties:
  - functionality
  - usability
  - performance
  - cost
  - dependability

Faults

- Faults are there!
- Either prevent, **tolerate**, remove or forecast
- We need redundancy
  - System that is more complex than needed for performing the required task

Means to Achieve Dependability

- Fault prevention
  - Good design processes, avoid design flaws
  - Good procedures for runtime faults
- Fault tolerance
  - Fault detection
  - Redundancy
  - Diversity
- Fault removal
  - Verification and validation during design
  - Corrective/preventive action during maintenance
- Fault forecasting
  - Simulation, modelling, prediction
  - Analysis based on history statistics

Fault Tolerance

- Automobile:
  - Spare Tires
  - Dual Braking Systems
- Power Supplies:
  - UPS/battery backup
  - Power-fail interrupts
- Multiple engines on aircraft
- Emergency lighting in buildings
- Tape backups of disk files
- Checkpoint/restart of long-running programs
- Parity and SECDED in computer memories

Faults

- Random faults (Degradation faults)
  - Arise during operation
  - Usually hardware component failure
- Systematic faults (Design Faults)
  - mistakes in the spec
  - mistakes in the hardware
  - mistakes in the software

Faults

- Faults are either permanent, transient or intermittent
- Design faults are always permanent
- Dealing with faults:
  - During development: fault avoidance & removal
  - During operation: fault tolerance & detection

Hardware Faults

- Use of fault models
- Decomposition into modules
  - Gates, transistors, etc
- Connection faults
  - Single stuck-at model, bridging model (shorts), stuck-open
- Used to model hardware faults
  - Design testing schemes for digital circuits
  - Fault removal coverage usually less than 100%
  - Guard against physical defects, not design faults
- In safety critical systems
  - Combined with Failure Modes and Effects Analysis (FMEA)
  - Need fault avoidance by verification…

Other Faults

- Hardware design and specification faults
  - Few fault models available
  - Many faults cannot be modelled
  - System must meet the spec, but spec might be incorrect as well
  - Spec errors may manifest as either hardware or software failures
  - Use of formal methods (formal spec. languages, automata theory, formal verification, model checking, etc.)

Software Faults

- Bugs:
  - Software spec faults
  - Coding faults
  - Logical errors within calculations
  - Stack overflows or underflows
  - Uninitialized variables
- No random failures and it does not degrade with age
- Always systematic
- Exhaustive testing almost impossible
- Must be tolerated

SW Testing - i.e. Verification

- Verification:
  - SW testing
  - formal verification
- Functional and structural testing
- Path testing, transaction flow testing, data-flow testing, domain testing, mutation testing etc.

Fault Detection Techniques

- Functionality checking
  - march test
- Consistency checking
  - range checking, overflow
- Signal comparison
- Information redundancy
  - checksums, cyclic redundancy codes, error correcting codes
- Monitoring techniques
  - Loopback testing
  - Power supply monitoring

Watchdog Timer

- An inexpensive method of error detection
- Process being watched must reset the timer before the timer expires, otherwise the watched process is assumed as faulty
- Watchdog timers only detect errors which manifest themselves as a control-flow error such that the system does not continue to reset the timer
- Only processes with relatively deterministic runtimes can be checked, since the error detection is based entirely on the time between timer resets

Heartbeats

- A common approach to detecting process and node failures in a distributed (networked) computing environment.
- Periodically, a monitoring entity sends a message (a heartbeat) to a monitored node or process and waits for a reply.
- If the monitored node does not respond within a predefined timeout interval, the node is declared as failed and appropriate recovery action is initiated.
- Adaptive or smart

System Testing

HW Testing SW Testing

HW/SW Testing
(system testing)