Lecture 6 – Dependability

SE 217 - Principles of Software Engineering

Dependability

- IFIP 10.4 Working Group on Dependable Computing and Fault Tolerance:
  - "[..] the trustworthiness of a computing system which allows reliance to be justifiably placed on the service it delivers [..]"

Dependability

- IEC IEV 191-02-03:
  - "dependability (is) the collective term used to describe the availability performance and its influencing factors : reliability performance, maintainability performance and maintenance support performance"

Importance of Dependability

- System failures may have widespread effects with large numbers of people affected by the failure
- Systems that are not dependable and are unreliable, unsafe or insecure may be rejected by their users

From: Software Engineering, Sommerville, 9th ed

Importance of Dependability

- The costs of system failure may be very high if the failure leads to economic losses or physical damage
- Undependable systems may cause information loss with a high consequent recovery cost

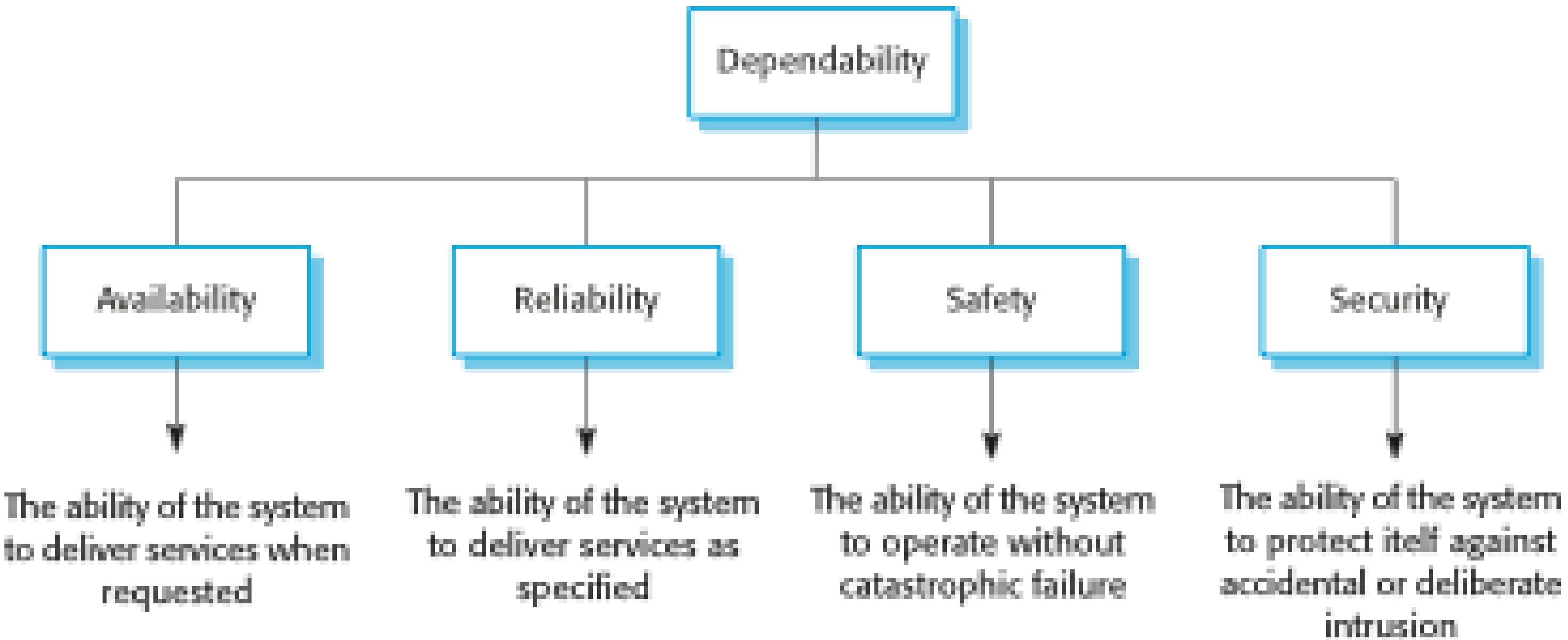
From: Software Engineering, Sommerville, 9th ed

Causes of Failure

- Hardware failure
  - Hardware fails because of design and manufacturing errors or because components have reached the end of their natural life
- Software failure
  - Software fails due to errors in its specification, design or implementation
- Operational failure
  - Human operators make mistakes
  - Now perhaps the largest single cause of system failures in socio-technical systems

From: Software Engineering, Sommerville, 9th ed

Principal Dependability Properties



From: Software Engineering, Sommerville, 9th ed

Other Dependability Properties

- Repairability
  - Reflects the extent to which the system can be repaired in the event of a failure
- Maintainability
  - Reflects the extent to which the system can be adapted to new requirements;
- Survivability
  - Reflects the extent to which the system can deliver services whilst under hostile attack;
- Error tolerance
  - Reflects the extent to which user input errors can be avoided and tolerated

From: Software Engineering, Sommerville, 9th ed

Reliability

- The ability of a system or component to perform its required functions under stated conditions for a specified period of time

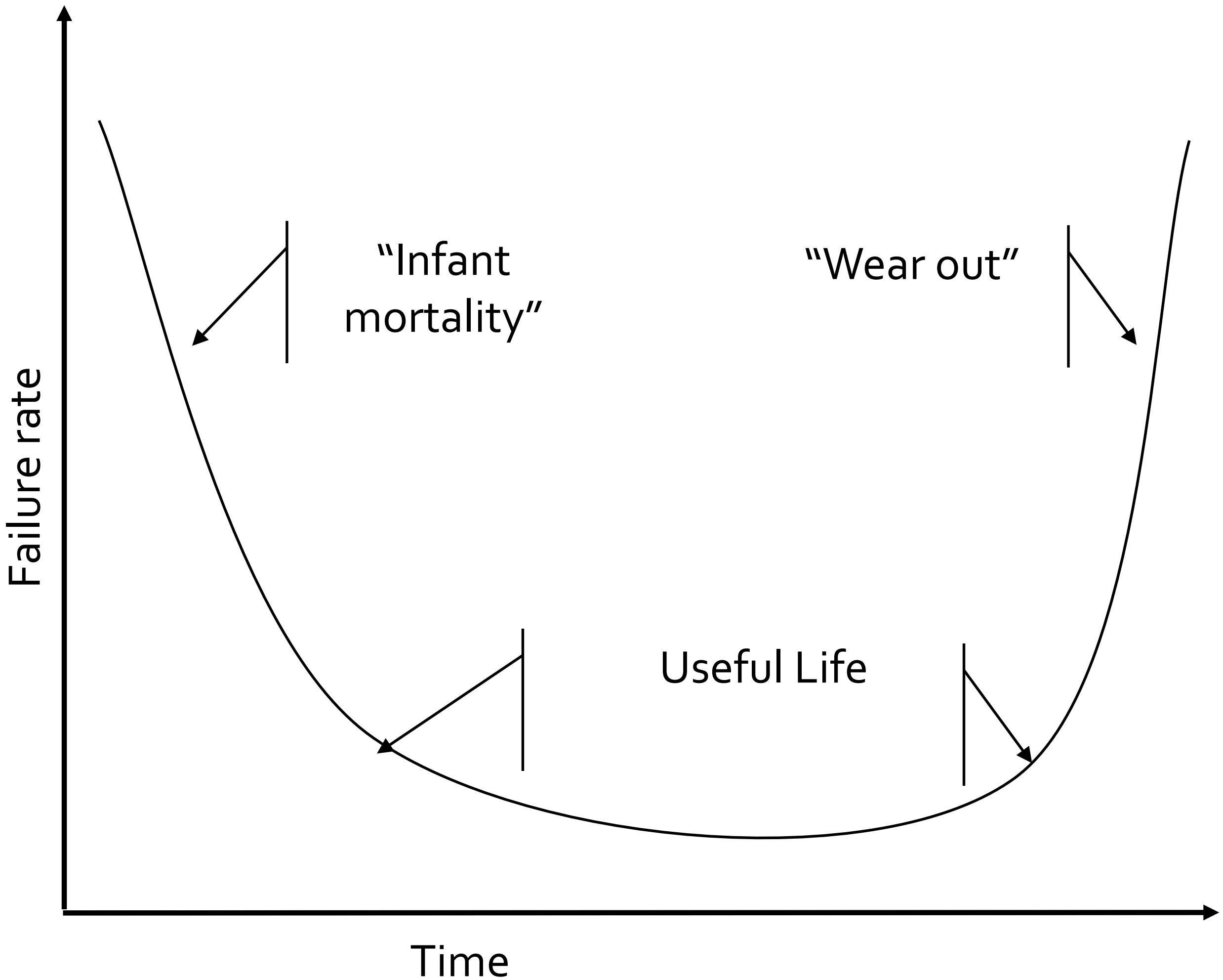
From: wikipedia.org

Reliability

- Closely related influences on overall reliability
  - Hardware reliability
    - What is the probability of a HW component failing and how long does it take to repair that component?
  - Software reliability
    - How likely is it that a SW component will produce an incorrect output?
  - Operator reliability
    - How likely is it that the operator will make an error?

From: Software Engineering, Sommerville, 5th ed

Hardware Failure Characteristics

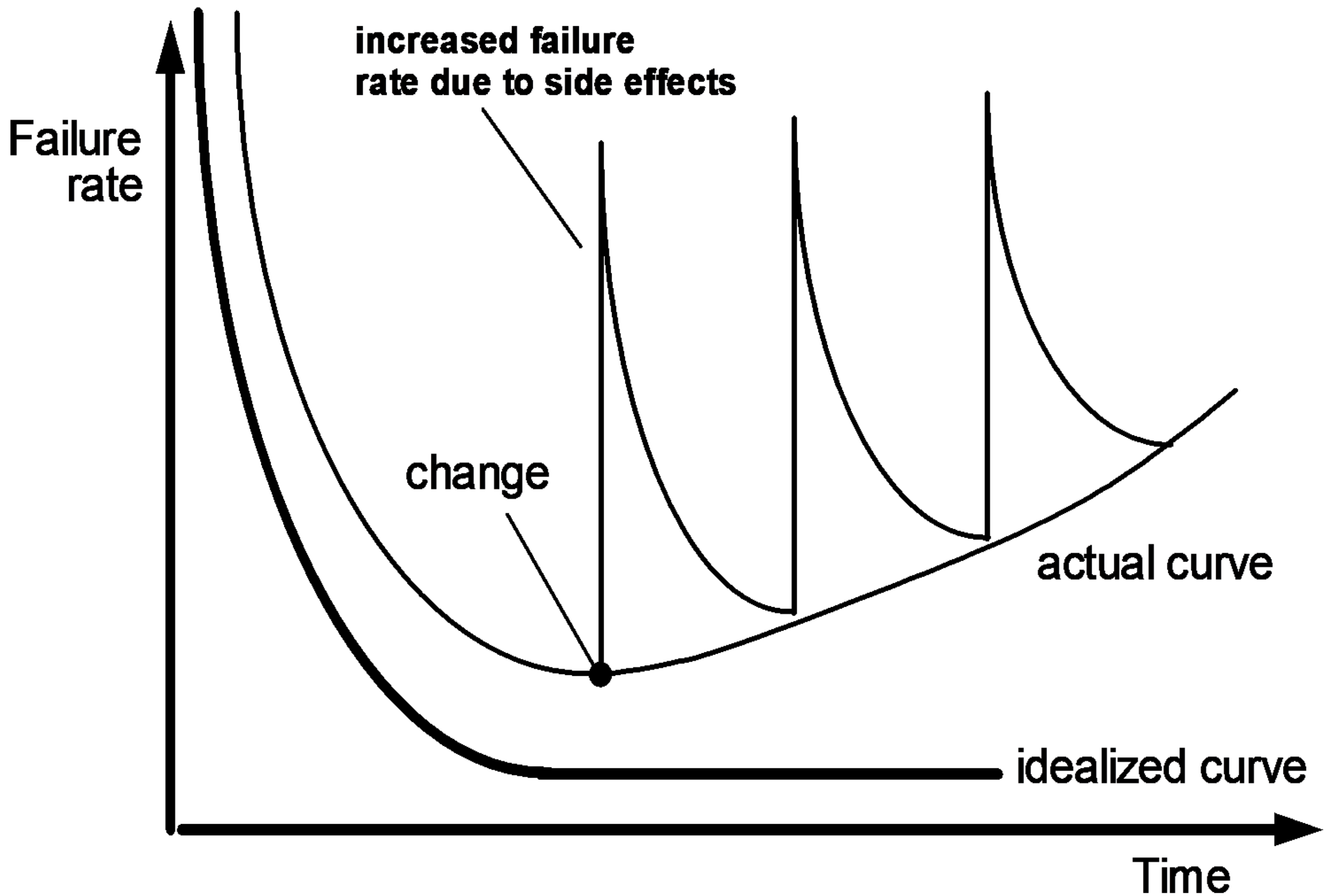


Hardware Failure Causes

- Design failures
  - Flaws in design
- Infant mortality
  - Manufacturing problems
- Random failures
  - Can occur during entire life
- Wear out

From: http://www.eventhelix.com/RealtimeMantra/FaultHandling/reliability_availability_basics.htm

Software Failure Characteristics



Software Failure Causes

- Process used to develop the design and code
- Complexity of SW
- Size of SW
- Experience of the team developing the software

From: http://www.eventhelix.com/RealtimeMantra/FaultHandling/reliability_availability_basics.htm

Software Failure Causes

- Percentage of code reused from a previous stable project
- Rigor and depth of testing before product is shipped

From: http://www.eventhelix.com/RealtimeMantra/FaultHandling/reliability_availability_basics.htm

Reliability Metrics

- MTBF (Mean Time Between Failures)
  - average time between failure of hardware modules
- FITS
  - Total number of failures of the module in a billion hours
- MTTR (Mean Time To Repair)
  - The time taken to repair a failed hardware module

From: http://www.eventhelix.com/RealtimeMantra/FaultHandling/reliability_availability_basics.htm

Availability

- The proportion of time a system is in a functioning condition

From: wikipedia.org

Availability

- A = MTBF / (MTBF+MTTR)
- Downtime
  - Time per year the system is not available

From: http://www.eventhelix.com/RealtimeMantra/FaultHandling/reliability_availability_basics.htm

Availability as percentage of uptime

Availability %	Downtime per year	Downtime per month	Downtime per week
90% ("one nine")	36.5 days	72 hours	16.8 hours
95%	18.25 days	36 hours	8.4 hours
98%	7.30 days	14.4 hours	3.36 hours
99% ("two nines")	3.65 days	7.20 hours	1.68 hours
99.5%	1.83 days	3.60 hours	50.4 minutes
99.8%	17.52 hours	86.23 minutes	20.16 minutes
99.9% ("three nines")	8.76 hours	43.2 minutes	10.1 minutes
99.95%	4.38 hours	21.56 minutes	5.04 minutes
99.99% ("four nines")	52.56 minutes	4.32 minutes	1.01 minutes
99.999% ("five nines")	5.26 minutes	25.9 seconds	6.05 seconds
99.9999% ("six nines")	31.5 seconds	2.59 seconds	0.605 seconds

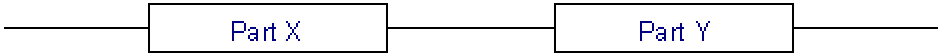
From: wikipedia.org

System Availability

- Calculated by modeling the system as an interconnection of parts in series and parallel

From: http://www.eventhelix.com/RealtimeMantra/FaultHandling/reliability_availability_basics.htm

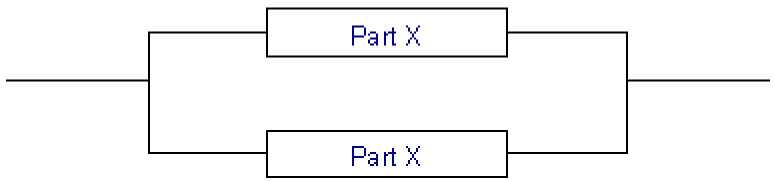
System Availability in Series



- $A = A_x * A_y$

From: http://www.eventhelix.com/RealtimeMantra

System Availability in Parallel



- $A = 1 - (1 - A_X)^2$

From: http://www.eventhelix.com/RealtimeMantra

Fault-tolerant (fail-safe) Design

- Enables a system to continue operation, possibly at a reduced level rather than failing completely, when some part of the system fails

From: wikipedia.org

Fault-tolerant (fail-safe) Design

- Most commonly used to describe computer-based systems designed to continue more or less fully operational with a reduction in throughput or an increase in response time in the event of some partial failure

From: wikipedia.org

Fault-tolerant (fail-safe) Design

- The system as a whole is not stopped due to problems either in the hardware or the software
- Providing fault-tolerant design for every component is normally not an option

From: wikipedia.org

Fault-tolerant Design - When to Use

- The following criteria may be used to determine which components should be fault-tolerant:
  - **How critical is the component?** In a car, the radio is not critical, so this component has less need for fault-tolerance.
  - **How likely is the component to fail?** Some components, like the drive shaft in a car, are not likely to fail, so no fault-tolerance is needed.
  - **How expensive is it to make the component fault-tolerant?** Requiring a redundant car engine, for example, would likely be too expensive both economically and in terms of weight and space, to be considered

From: wikipedia.org

Fault-tolerant System

- First fundamental characteristic of fault-tolerance is no single point of failure
- This is achieved in different ways
  - **Replication**
    - Multiple instances of the same system, direct tasks to all of them in parallel and choose the correct result based on a quorum
  - **Redundancy**
    - Multiple instances of the same system and switch to one of the remaining components in case of failure (failover)
  - **Diversity**
    - Multiple different implementations of the same specification, and use them like replicated systems to cope with errors in a specific implementation

From: wikipedia.org

Replication

- Sharing information so as to ensure consistency between redundant resources, to improve reliability, fault-tolerance, or accessibility

From: wikipedia.org

Replication

- Data replication
  - Same data is stored on multiple storage devices (RAID, etc)
- Computation replication
  - Same computing task is executed many times

From: wikipedia.org

Redundancy

- Duplication of critical components of a system with the intention of increasing reliability of the system, usually in the case of a backup or fail-safe

From: wikipedia.org

Redundancy

- In many safety-critical systems, such as fly-by-wire and hydraulic systems in aircraft, some parts of the control system may be triplicated
  - An error in one component may then be out-voted by the other two

From: wikipedia.org

Forms of Redundancy

- Major forms of redundancy
  - Hardware redundancy, such as DMR (dual modular redundancy) , TMR (triple MR) and QMR (quadruple MR)
  - Information redundancy, such as Error detection and correction methods
  - Time redundancy, including transient fault detection methods such as Alternate Logic
  - Software redundancy such as N-version programming

From: wikipedia.org

Hardware Fault Tolerance

- Redundancy Schemes
  - One for One Redundancy
    - Each hardware module has a redundant hardware module
    - The hardware module that performs the functions under normal conditions is called Active and the redundant unit is called Standby
    - Standby module takeovers and becomes active if the active unit fails

From: http://www.eventhelix.com/RealtimeMantra

Hardware Fault Tolerance

- Redundancy Schemes
  - N + X Redundancy
    - If N hardware modules are required to perform system functions, the system is configured with N + X hardware modules; typically X is much smaller than N
    - Whenever any of the N modules fails, one of the X modules takes over its functions

From: http://www.eventhelix.com/RealtimeMantra

Hardware Fault Tolerance

- Redundancy Schemes
  - Load Sharing
    - Under zero fault conditions, all the hardware modules that are equipped to perform system functions, share the load
    - If one of the load sharing module fails, the load is distributed among the rest of the units

From: http://www.eventhelix.com/RealtimeMantra

Hardware Fault Tolerance

- Computer Cluster
  - A group of linked computers, working together closely thus in many respects forming a single computer

From: wikipedia.org

Hardware Fault Tolerance

- Cluster Categories
  - High-availability clusters (Failover Clusters)
    - Implemented primarily for the purpose of improving the availability of services that the cluster provides
  - Load-balancing clusters
    - Multiple computers are linked together to share computational workload or function as a single virtual computer
  - Compute clusters
    - Used primarily for computational purposes, rather than handling IO-oriented operations such as web service or databases

From: wikipedia.org

Software Fault Tolerance

- Redundancy Models
  - Timeouts
    - Use timers to keep track of feature execution
    - A timeout generally signals that some entity involved in the feature has misbehaved and a corrective action is required
    - Retry: When the application times out for a response, it can retry the message interaction.
    - Abort: In this case timeout for a response leads to aborting of the feature

From: wikipedia.org

Software Fault Tolerance

- Redundancy Models
  - Audits
    - Generally software running across multiple processors.
      - This implies that data is also distributed
    - Audit is a program that checks the consistency of data structures across processors by performing predefined checks

From: wikipedia.org

Software Fault Tolerance

- Redundancy Models
  - Exception Handling
    - Whenever a task receives a message, it performs a series of defensive checks before processing it
    - The defensive checks should verify the consistency of the message as well as the internal state of the task
    - Exception handler should be invoked on defensive check failure

From: wikipedia.org

Software Fault Tolerance

- Redundancy Models
  - Task Rollback
    - A software bug in one task leading to processor reboot may not be acceptable
    - A better option in such cases is to isolate the erroneous task and handle the failure at the task level
    - The task in turn may decide to rollback i.e. start operation from a known or previously saved state

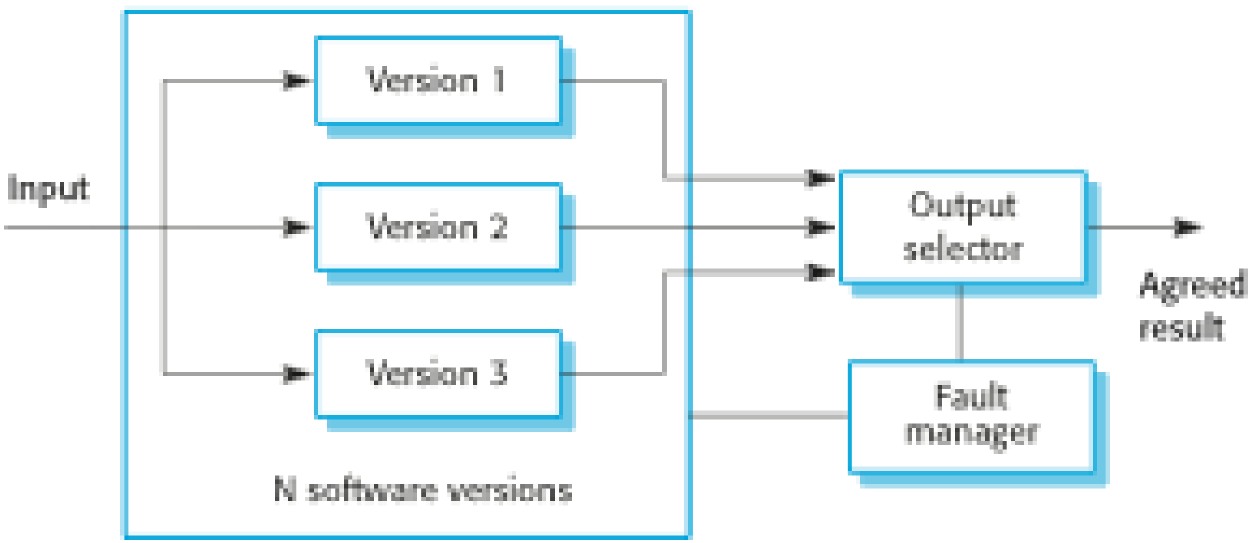
From: wikipedia.org

Software Fault Tolerance

- Redundancy Models
  - Voting
    - This is a technique that is used in mission critical systems where software failure may lead to loss of human life .e.g. aircraft navigation software.
    - System software is developed by at least three distinct teams, independently
    - All the three implementations are run simultaneously
    - All the inputs are fed to the three versions of software and their outputs are voted to determine the actual system response

From: wikipedia.org

N-version programming



From: Software Engineering, Sommerville, 9th ed

Disaster Recovery

- The process, policies and procedures related to preparing for recovery or continuation of technology infrastructure critical to an organization after a natural or human-induced disaster

From: wikipedia.org

Disaster Recovery Strategies

- Backups made to tape and sent off-site at regular intervals (preferably daily)
- Backups made to disk on-site and automatically copied to off-site disk, or made directly to off-site disk

From: wikipedia.org

Disaster Recovery Strategies

- Replication of data to an off-site location, which overcomes the need to restore the data (only the systems then need to be restored or synced).
  - This generally makes use of storage area network (SAN) technology

From: wikipedia.org

Disaster Recovery Strategies

- High availability systems which keep both the data and system replicated off-site, enabling continuous access to systems and data
- Local mirrors of systems and/or data and use of disk protection technology such as RAID

From: wikipedia.org

Disaster Recovery Precautions

- Surge protectors — to minimize the effect of power surges on delicate electronic equipment

From: wikipedia.org

Disaster Recovery Precautions

- Uninterruptible power supply (UPS) and/or backup generator to keep systems going in the event of a power failure
- Fire preventions — alarms, fire extinguishers
- Anti-virus software and other security measures

From: wikipedia.org

UI Design

- Is critical in avoiding operator error
- For complex systems the principal objective of UI design must be to produce safe and resilient interfaces

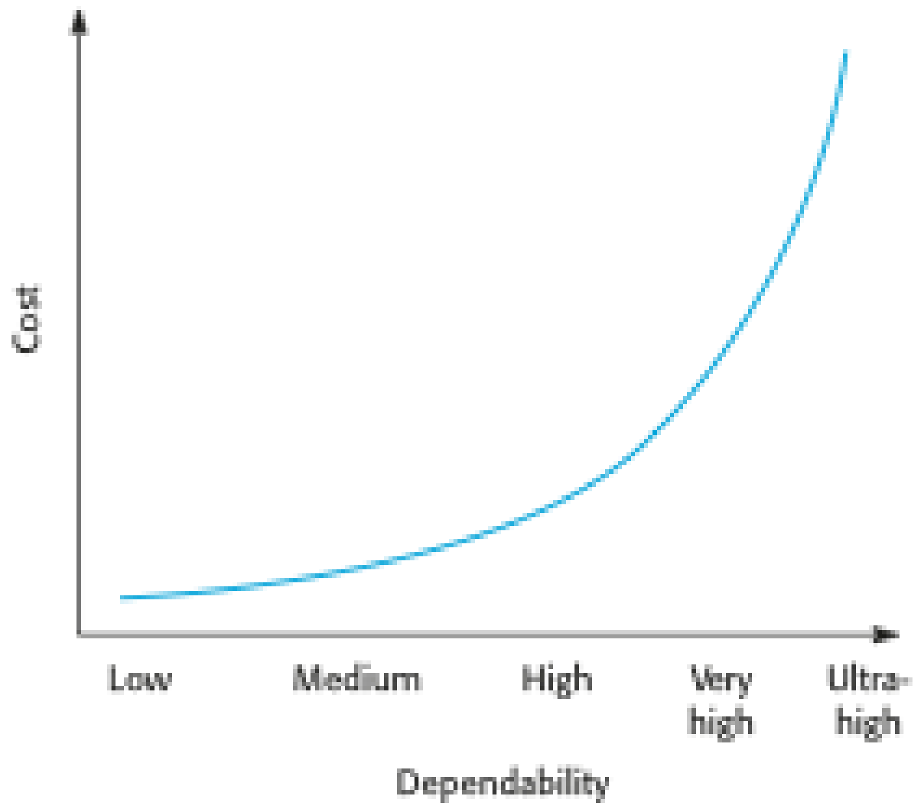
From: Software Engineering, Sommerville, 5th ed

UI Design

- Find answers to
  - What should be considered as an operator error?
  - How can the system be designed so that mistakes are avoided?
  - What information is needed and how can be presented so that operator will not misread under stressful conditions?
  - How can HW and SW errors be detected and reported to the operator?
  - What degree of operator overriding should be allowed?

From: Software Engineering, Sommerville, 5th ed

Cost/dependability curve



From: Software Engineering, Sommerville, 9th ed