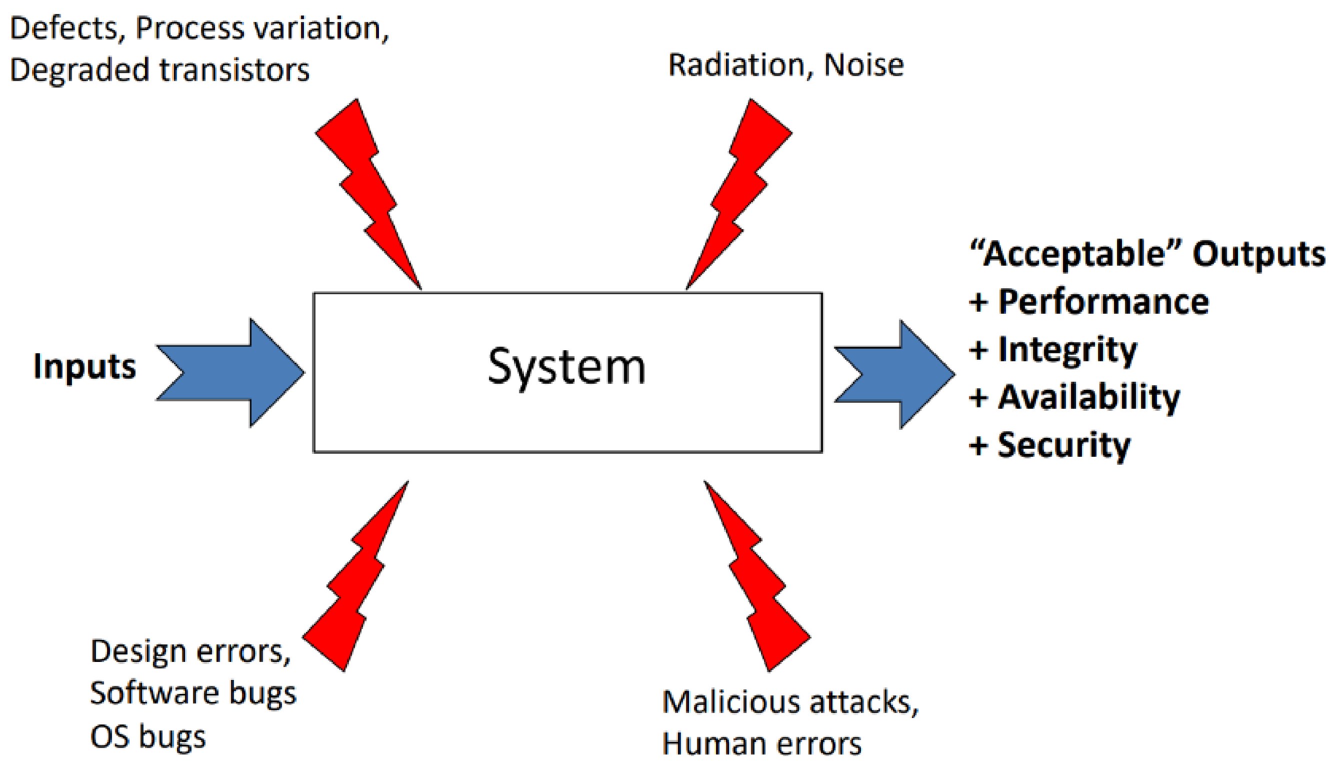
Dependability

Dependability: a measure of how much we trust a system. The ability of a system to perform its functionality while exposing:

- Reliability: continuity of correct service
- Availability: readiness for correct service
- Maintainability: ability for easy maintainance
- Safety:absence of catastrophic consequences
- Security: confidentiality and integrity of data

Why dependability

A lot of effort is devoted to make sure the implementation matches specifications, fulfill requirements, meets constraints, optimizes selected parameters (performance, energy, …) Nevertheless, even if all above aspects are satisfied … things may go wrong


Defects, Process variation, Degraded transistors
Radiation, Noise
Inputs
System
"Acceptable" Outputs
+ Performance
+ Integrity
+ Availability
+ Security
Design errors, Software bugs OS bugs
Malicious attacks, Human errors

Failure effects:

- A single system failure may affect a large number of people
- A failure may have high costs if it impacts economic losses or physical damage
- Systems that are not dependable are likely not be used or adopted
- Undependable systems may cause information loss with a high consequent recovery cost

When to think about dependability?

You have to think about dependability both at design time and at runtime

- Design time: analyse the system under design, measure dependability properties, modify the design if required
- Runtime: detect malfunctions, understand causes, react
  Failures occur in development & operation
- Failures in development should be avoided
- Failures in operation cannot be avoided (things break), they must be dealt with
  Design should take failures into account and guarantee that control and safety are achieved when failures occur. Effects of such failures should be predictable and deterministic … not catastrophic

Where to apply dependability?

Initially dependability has been a relevant aspect only for safety-critical and mission-critical application environments as it has huge costs and it was acceptable only when mandatory. However it shouldn't be neglected even in everyday/basic appliance.
Non-critical critical systems: a failure during operation can have economic and reputation effects(Consumer products)
Mission-critical systems: a failure during operation can have serious or irreversible effects on the mission the system is carrying out(Satellites, Automatic Wheather Stations, Surveillance drones, Unmanned vehicles)
Safety-critical systems: a failure during operation can present a direct threat to human life(aircraft control systems, medical instrumentation, railway signaling, nuclear reactor, control systems)

How to provide dependability?

You can use failure avoidance paradigm as:

- Conservative design
- Design validation
- Detailed test on HW/SW
- Infant mortality screen
- Error avoidance
  You can also use failure tolerance paradigm:
- Error detection / error masking during system operation
- On-line monitoring
- Diagnostics
- Self-recovery & self-repair

Key elements

Dependable systems can be achieved by means of a Robust design (error-free), Processes and design practices, Robust operation (fault tolerant), Monitoring, detection and mitigation

Safety-critical systems

Reliability

The ability of a system or component to perform its required functions under stated conditions for a specified period of time.
Definition: R(t): probability that the system will operate correctly in a specified operating environment until time *t*
R(t) = P(not failed during [0, t]) assuming it was operating at time t = 0
1 – R(t): ***unreliability***, also denoted Q(t) is the probability that a system is not working after a period of time equal to *t*
R(t) is a non-increasing function varying from 1 to 0 over $[0, +\infty)$

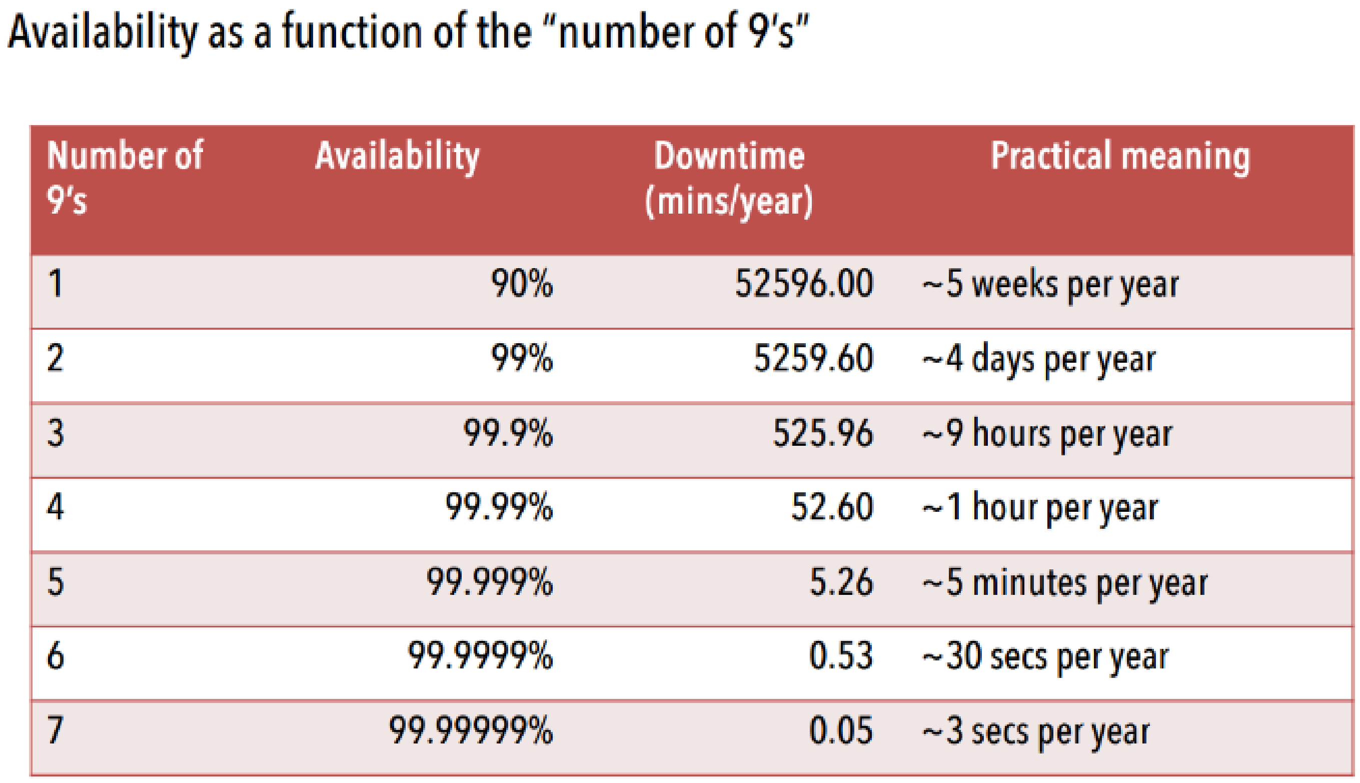
$$\lim_{x \to +\infty} R(t) = 0, \lim_{x \to +\infty} Q(t) = 1$$

Often used to characterize systems in which even small periods of incorrect behavior are unacceptable:

- Performance requirements
- Timing requirements
- Extreme safety requirements
- Impossibility or difficulty to repair

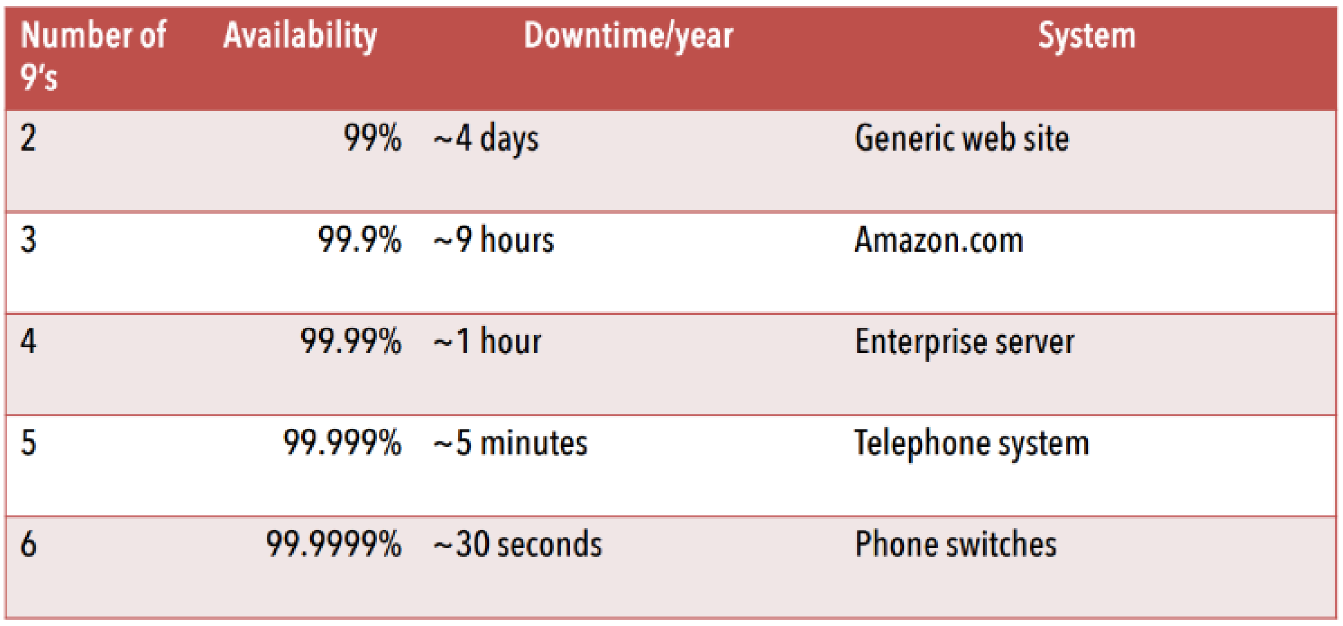
Availability

The degree to which a system or component is operational and accessible when required for use: Availability = Uptime / (Uptime + Downtime)
Definition: A(t): probability that the system will be operational at time *t*
A(t) = P(not failed at time t), it is a constant probability in time
1 – A(t): unavailability
Reliability and Availability are 0 if a system is broken.
In general (repairable systems): A(t) ≥ R(t) [A system can be available also when it is not reliable]



Availability as a function of the "number of 9's"

Number of 9's	Availability	Downtime (mins/year)	Practical meaning
1	90%	52596.00	~5 weeks per year
2	99%	5259.60	~4 days per year
3	99.9%	525.96	~9 hours per year
4	99.99%	52.60	~1 hour per year
5	99.999%	5.26	~5 minutes per year
6	99.9999%	0.53	~30 secs per year
7	99.99999%	0.05	~3 secs per year



Number of 9's	Availability	Downtime/year	System
2	99%	~4 days	Generic web site
3	99.9%	~9 hours	Amazon.com
4	99.99%	~1 hour	Enterprise server
5	99.999%	~5 minutes	Telephone system
6	99.9999%	~30 seconds	Phone switches

Definition

MTTF (Mean Time To Failure): mean time before any failure will occur(in particular the first failure after a repair)

MTBF (Mean Time Between Failures): mean time between two failures, $MTBF = \frac{totaloperatingtime}{numberoffailures}$

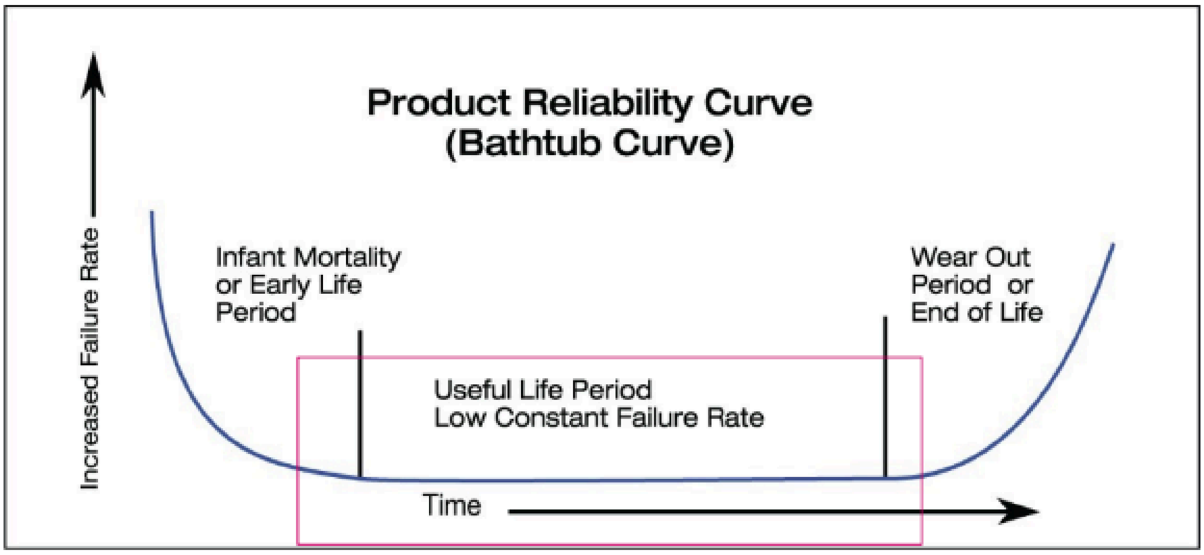
MTTF (Mean Time To Failure): mean time before any failure will occur $MTTF = \int_0^\infty R(t)dt$

FIT: failures in time

- another way of reporting MTBF
- the number of expected failures per one billion hours (10^9) of operation for a device
- MTBF (in h) $=10^9/\text{FIT}$

$$\text{Failure Rate } \lambda = \frac{\text{number of failures}}{\text{total operating time}}$$

$$\text{MTBF} = \frac{1}{\lambda}$$



- *Infant Mortality*: failures showing up in new systems. Usually this category is present during the testing phases, and not during production phases.
- *Random Failures*: showing up randomly during the entire life of a system.
  - Our main focus
- *Wear Out*: at the end of its life, some components can cause the failure of a system. Pre-emptive mainteinance can reduce the number of this type of failures.

Of course reliability and availability are related:

- if a system is unavailable it is not delivering the specified system services
  It is possible to have systems with low reliability that must be
  available:
- system failures can be repaired quickly and do not damage data, low reliability may not be a problem (for example a database management system)
  The opposite is generally more difficult

Reliability terminology

Fault: a defect within the system, an incorrect system failure, it can be quiet, not detected by the system
Error: a deviation from the required operation of the system or subsytem

Failure: an error that produce a different behaviour from the expected one
Not always the chain fault-error-failure close as there could be some error/fault not resulting in failure or sometimes the fault is innoque.

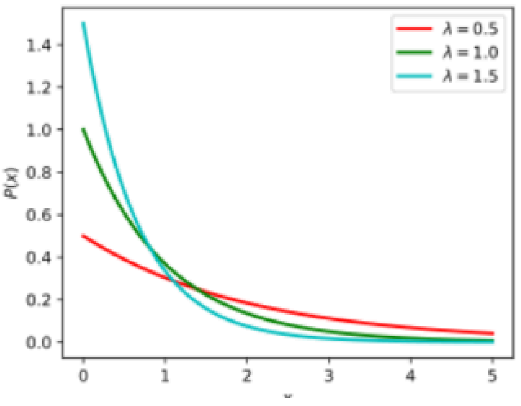
Reliability Block Diagrams

We have to compute as fast as possible the reliability and a good model to do so is the Reliability Blocks Diagrams. It doesn't represent your system but how it has to work.
An inductive model where a system is divided into blocks that represent distinct elements such as components or subsystems.
Every element in the RBD has its own reliability (previously calculated or modelled)
Blocks are then combined together to model all the possible success paths

Assuming that a failures occurs according to a Poisson model, it models the time between two successive failures:

- Probability density function: $f(t;\lambda) = \lambda e^{-\lambda t},\ t \geq 0,\ \lambda > 0$
- Cumulative density function: $P(T \leq t) = \int_0^t f(s;\lambda)\, ds = 1 - e^{-\lambda t}$
- Expected value: $E[T] = \frac{1}{\lambda}$
- Variance: $\sigma^2(T) = \frac{1}{\lambda^2}$

Reliability: $R(t) = P(T \geq t) = e^{-\lambda t}$
$\lambda(t)$: failure rate



To simplify things we will consider λ fixed, constant.
We can represent the components in series(all the components have to be up to have a functioning system) and in parallel(at least one components has to be up to have a functioning system). The RBD is a graph, a success path.
For components in series the failure rate is $R_s(t) = e^{-\lambda_s t}$ where $\lambda_s = \sum_{i=1}^{n} \lambda_i$ and

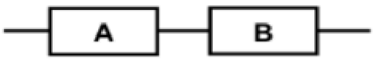
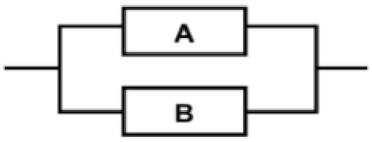
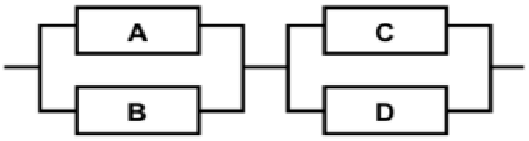
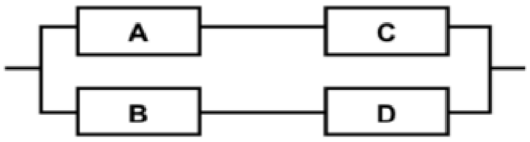
$$MTTF_s = \frac{1}{\lambda_s} = \frac{1}{\sum_{i=1}^{n} \lambda_i} = \frac{1}{\sum_{i=1}^{n} \frac{1}{MTTF_i}}$$

$$R_s = \prod_{i}^{n} R_i$$

$$R_s = 1 - \prod_{i}^{n} (1 - R_i)$$

Component redundancy

System redundancy

Type	Block Diagram Representation	System Reliability (R_S)
Series		$R_S = R_A R_B$ R_A = reliability, component A R_B = reliability, component B
Parallel		$R_S = 1-(1-R_A)(1-R_B)$
Series-Parallel		$R_S = [1-(1-R_A)(1-R_B)]^*$ $[1-(1-R_C)(1-R_D)]$ R_C = reliability, component C R_D = reliability, component D
Parallel-Series		$R_S = 1-(1-R_A R_C)^*$ $(1-R_B R_D)$

A system may be composed of two parallel replicas:

- The primary replica working all time, and
- The redundant replica (generally disable) that is activated when the primary replica fails Obviously we need:
- A mechanism to determine whether the primary replica is working properly or not (on-line self check)
- A dynamic switching mechanism to disable the primary replica and activate the redundant one