System • Technical and organizational means for the autonomous fulfillment of a task (based on Birolini, ETH) • Generally, a system can consist of hardware, software, people (service and maintenance personnel) and logistic assistance • **Example**: Aviation Hardware= Aircraft, tower, runway Software = Flight control software, software for in-flight entertainment system People= Pilot, co-pilot, stewardess, ground personnel Logistic assistance= Fueling, Baggage handling, ticketing. **Technical System** • System where influences by people and logistics are ignored, **Example**: Avionics Hardware = onboard flight control unit. Software = Flight control software. **Quality** • Degree in which the inherent attributes of an entity fulfill quality requirements /DIN EN ISO 9000 05/ • **Quality Requirement** • Expectation or demand defined (by a customer) that is generally assumed or mandatory /DIN EN ISO 9000 05/ • **Quality Characteristic** • Property of an entity on the basis of which its quality is described and estimated, but which makes no statement about the degree of fulfillment of the characteristic • A quality characteristic can be refined incrementally into partial characteristics • Inherent attribute of a process, product or a system that relates to a quality requirement /DIN EN ISO 9000 05/. • **Quality Measure** • Measure which allows to draw conclusions on the fulfillment of specific quality characteristics. For instance, MTTF (Mean Time To Failure) is a quality measure of the quality characteristic Reliability. **Security** is the concurrent users only a. availability for the authorized users only b. confidentiality c. integrity **Safety** • State where the danger of a personal or property damage is reduced to an acceptable value (DIN EN ISO 8402) • Birolini defines safety as a measure for the ability of an item to endanger neither persons, property nor the environment • A distinction is drawn between the safety of a failure-free system (accident prevention) and the technical safety of a failure afflicted system • Absence of unacceptable risks /IEC 61508 98/ • **Technical Safety** • Measure for the ability of a failure afflicted item to endanger neither persons, property nor the environment, **Correctness** • Correctness has a binary character, i.e., an item is either correct or incorrect • A fault-free realization is correct • An artifact is correct if it is consistent to its specification • If no specification exists for an artifact, correctness is not defined • **Completeness** • A system is functional complete, if all functions required in the specification are implemented. This concerns the treatment of normal cases as well as the interception of failure situations. **Robustness** • Property to deliver an acceptable behavior also in exceptional situations (e.g. ability of a software to detect hardware failures) • A correct system – as measured by the specification – can have a low robustness, actually • Accordingly, robustness is rather a property of the specification than of the implementation • A robust program is the result of the correct implementation of a good and complete specification • Robustness has a gradual character. **Reliability** • Part of the quality with regard to the behavior of an entity during or after given time periods with given working conditions (DIN 40041) • Collective term for the description of the power concerning availability and its influencing factors: power concerning functionality, maintainability and maintainability support (DIN EN ISO 8402) • Property of an entity regarding its qualification to fulfill the reliability requirements during or after given time periods with given application requirements (DIN ISO 9000) • Measure for the ability of an item to remain functional, expressed by the probability that the required function is executed failure-free under given working conditions during a given time period (based on Birolini, ETH) • **Availability(MTBF/MTBF+MTTR)** • Measure for the ability of an item to be functional at a given time,• **Failure**: Inconsistent behavior w.r.t. specified behavior while running a system (happens dynamically during the execution) => Each failure has a time-stamp • **Fault**, defect: Statically existent cause of a failure, i.e. a „bug" (usually the consequence of an error made by the programmer) • **Error**: Basic cause for the fault (e.g., misunderstanding of a particular statement of the programming language). **Accident** is an undesired event that causes death or injury of persons or harm to goods or to the environment • **Hazard** is a state of a system and its environment where the occurrence of an accident depends only on influences that are not controllable by the system • **Risk** is the combination of hazard probability and severity of the resulting accident • **Acceptable Risk** is a level of risk that authorities or other bodies have defined as acceptable according to acceptance criteria. **Hardware failures** • Commonly caused by manufacturing errors or wear (physical degradation) • Traditionally, potential design faults within hardware are disregarded… but the assumption that today's complex hardware is free from design faults cannot be hold anymore! (use of SBLs VHDL, Verilog). • In the ideal case, the system contains no hardware defects in the beginning and therefore shoes no hardware failures. • The reliability of the system does not exceed its initial value through the substitution of components with new components. **Software failures** • It is commonly assumed that the software contains defects, which cannot be detected immediately • Software failures are a result of design errors that are contained in the product from the start and appear accidentally • After error correction the system reliability exceeds its initial value (under the assumption that no additional faults are introduced) • Faults that are introduced during debugging decrease reliability

---

Definition of risk: R = H * S • H: expected frequency of the occurrence of an event that leads to a particular harm • S: expected severity of the harm, **Goals**: The aim of risk acceptance is to bring about a decision in a systematic and founded fashion whether the risk under consideration can be accepted or not. In the latter case, the system causing the risk cannot be put operational • In particular for safety-critical systems, admission offices follow such a procedure as a prerequisite for putting the system in operation (e.g., for railway transportation systems), • The costs for risk reduction do not increase linearly with reducing residual risks. Merely, they are disproportionately high. Therefore, there exists an economically optimal trade-off between the costs of a system and its residual risks. This tradeoff could be acceptable, but it can also be the case that the residual risks are still too high and further risk reduction is demanded. **Influencing Factors:** Deciding, which risks are acceptable, is also subjective and depends among other things on the following factors • Degree of benefit? Great distances in aviation: Is the exposure to this particular risk related to travel distance or time spent in the aircraft? • Who is at risk? Astronauts, sick persons, railway travelers, service personnel, uninvolved public • Degree of self-determination? – Driving a car vs. taking an elevator • How many people are at risk? – Car vs. nuclear power plant • Severity? Death or injuries?, **Important risk acceptance methods** • **MEM** (Minimal Endogenous Mortality) • **GAMAB** (Globalement Au Moins Aussi Bon) • **ALARP** (As Low as Reasonably Practicable). **MEM** – **(13 year-old 2×10-4 deaths per person** and year, **10^-5 deaths per person for adults)** Minimal Endogenous Mortality • The Minimal Endogenous Mortality method is based upon the fact that there exist different mortality rates in society, depending on age and gender. These deaths are partly caused by technical systems. MEM now compares the risks due to a new system with already existing risks caused by „natural" mortality. MEM demands that the new system does not significantly contribute to the existing mortality caused by technical systems. The MEM method can also be used in such cases, where the comparison between a novel system and similar pre-existing systems is not feasible. **GAMAB** – Globalement Au Moins Aussi Bon • Unlike MEM, GAMAB requires the existence of a reference system with accepted residual risks • According to GAMAB, residual risks caused by a new system must not exceed those of the reference system. **ALARP** – As Low as Reasonably Practicable • ALARP aims to minimize risks under consideration of economic and social aspects. ALARP tries to assess what is technically feasible within the context of financial feasibility and acceptance in society • The overall risk can fall into one of three possible ranges (**1**. The risk is negligible and can be accepted without further measures 2. The risk is higher than commonly accepted but falls below the upper limit of tolerability 3. The risk is unacceptably high. If the risk is irrelevant, ALARP does not demand any further measures • If the risk is unacceptably high**)**, measures to reduce this risk must be taken in either case. **Safety Integrity** "probability of an E/E/PE* safety-related system satisfactorily performing the required safety functions under all the stated conditions within a stated period of time" (DIN EN 61508-4:2010) • **Safety Integrity Level** (SIL) "discrete level (one out of a possible four) corresponding to a range of safety integrity values where safety integrity level 4 has the highest level of safety integrity and safety integrity level 1 has the lowest" (DIN EN 61508-4:2010)

Hazard and Operability Study (HAZOP) • From chemical industry • Find potential hazards at early process stage • Check every "flow" in preliminary design scheme for deviations • Manual search using guide-words (more, less, no, reverse...) • **Preliminary Hazard Analysis (PHA)** • During requirements analysis or early design phase • Coarse identification, classification and counter-measures for potential hazards • Table representations. **The Failure Mode, Effects and Criticality Analysis (FMECA)** is a preventive method for the identification of problems, their risks and effects • **Goals** • **(**Detection of hazards and problems • Identification of potential risk • Quantification of risks • Determination of corrective measures**)** • FMECA can be performed as component FMECA (e.g. for a subsystem), as system FMECA (a complete system) or as process FMECA (e.g. for a development process), **Accomplishment** • Formulate proposed actions **(**• Gear proposed solutions towards fault prevention • High occurrence probabilities of faults: An improvement is definitely necessary (also in the case of low severity and high detection probability) • High severity: In this case corrective measures are also required because of the consequences • High non-detection probability: Improvement of detection probability by suitable analytical instruments **)** • Decide for actions • Analyze residual risk (recalculate RPN) • Conduct cost-benefit analysis • Comparison of RPN before and after the improvement • Relate obtained improvement to invested effort. • **FMECA is done in the following steps** • Fault analysis: Collection of possible faults including available information about the type, causes and consequences • Risk evaluation with the aid of the risk priority number RPN = occurrence probability * severity of consequences * probability of non-detection • If for the three influencing factors a value between 1 and 10 is used (1= no risk, minor occurrence; 10 = high risk, high occurrence), the RPN is a value between 1 and 1000 • The risk priority number generates a ranking for the causes of faults • Causes of faults with a high risk priority number are to be handled with priority, **RBD:** Interconnection of all components of a system which are involved in performing the required function; represented as a flow chart • RBDs distinguish only two states (intact/failed) • Reliability function R(t) • F(t) gives the probability that at time t at least one failure has occurred; thus R(t) = 1 - F(t) is the probability that at time t no failure has occurred yet, **Markov models** are based on a description of the system behavior with state machines • Common assumption of all Markov Models: The probability of the next state depends on the current state; it is independent from previous states, i.e. Markov models do not take into account the history • Various Model types, e.g.: • Discrete time models (Markov chain) • Continuous time models; also called Markov processes. **Markov chains** assume that state changes occur at discrete points in time. **Markov processes** are continuous time models, **Petri Nets** • Condition/Event Petri nets • State/Transition Petri nets • Predicate/Transistion Petri Nets / Coloured Petri Nets • Timed Petri Net Types**(** • SPN • GSPN • DSPN**)**, A Petri Net N contains at least places (P), transitions (T) and a flow relation (F) as well as an initial marking (M0): N = (P, T, F, M0): • $P \cap T = \emptyset$ • $F \subseteq (P \times T) \cup (T \times P)$ • $M0 : P \rightarrow IN0$,

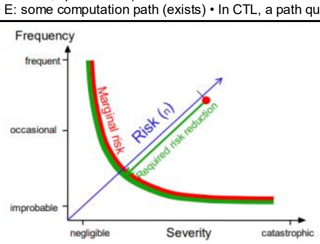
---

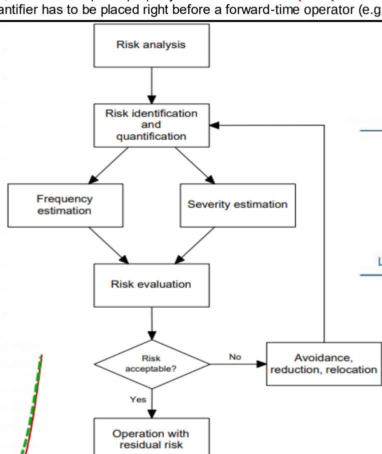
Fault Tree Analysis method for the qualitative and quantitative evaluation of a specific failure of a system • (Deductive (backward searching) • Graphical and intuitive technique • Based on Boolean logic and combinatorics • Widely accepted, captured in standards / handbooks • Has been used and extended since 1961), Causes for the effect can be defective system components • FTA is applied particularly in complex systems in order to analyze safety-critical effects of failures **Fault trees** trace back influences to a given hazard or failure • Help to find all influences • Graphically explain causal chains leading to the hazard • Find event combinations that are sufficient to cause hazard (qualitative analysis) • Calculate hazard probability from influence probabilities (quantitative analysis), **Root**: "**Top-Event**" The hazard or failed state (or the accident or failure event) • Leaves: "**Basic Events**" The causes that cannot or shall not be refined any further • **Gates**: Logical connectives, **In probability theory, "event" means everything that can happen with a given probability**, events can be • Sudden events ("Bolt breaks") • States or conditions ("Valve is blocked") • (Informal) propositions ("Fire is not detected by supervisor") **Exclusive OR (XOR):** Output occurs when exactly one of the input events is true 3. **N-out-of-M Voter alias Combination Gate**: Output occurs if at least n of the m input events occur 4. **Priority AND**: Output occurs when all input events occur in the specified order, **FTs** are useful even without any analysis • Help understanding the system • Reveal problem areas immediately • Build up awareness for safety and reliability issues, **Qualitative Analysis**: Check, if the top-event is reachable • Find minimal cut sets • e.g. list all cut sets with order 1 or 2 • e.g. list all cut sets with total probability > 0.01 (requires quantitative analysis) • Find minimal path sets, **Quantitative analysis** produces numerical results • Probability or rate of top-event / of a given cut set • Importance of basic events / cut sets • (How much impact has an event on the total failure probability? • By how much is the total failure probability influenced by changes / uncertainties regarding a particular event?), **Safety analysis** aims at proving that the actual risk is below the acceptable risk, A **Cut Set** is a set of basic events, which in conjunction cause the top event • **A Minimal Cut Set (MCS)** is a cut set that no longer is a cut set if any of its basic events is removed • A **Path Set** is a set of basic events that, if they are false, inhibit the top event from occurring • A **Minimal Path Set (MPS)** is a path set that no longer is a path set if any of its basic events is removed, **BDD** = Binary Decision Diagram • OBDD = Ordered BDD (defined variable order) • **ROBDD** = Reduced Ordered BDD (after elimination of redundancies), **Importance measures quantify the significance of FT** events in terms of their contribution to the top-event probability • To know the importance of part of the FT is important for • (Robustness Estimation: How much will my result change if input values are roughly estimated or change during operation? • Work planning: You should rather spend your time on system changes that have significant impact on overall failure probability), **Fussell-Vesely Importance** • Absolute or relative (= percentage) contribution to the top-event probability • **Risk Reduction Worth or Top Decrease Sensitivity** • Decrease of top-event probability if a given event is assured not to occur • **Risk Achievement Worth** • Increase of top-event probability if a given event occurs • **Binbaum's Importance Measure** • Rate of change of top-event probability in relation to rate of change of a given event, **Issues in Quantitative Analysis: Uncertainty Quantification** • **(**Event data is taken from samples or from other environment • Sensitivity analysis or formal uncertainty analysis (assigning a probability distribution)**)** • **Coverage Factors** • Take into account that some failures do not lead to catastrophic results • **Time or Phase Dependent Analysis** • Use different models or rates for different time intervals according to mission phases.

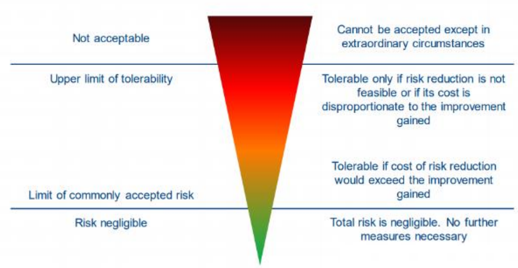
FTA is one technique for probabilistic risk assessment • It must be embedded in a safety respecting process, assuring • (construction of correct, reliable and safe hardware and software • analysis and validation of safety and reliability of the whole system in its operation environment throughout all process phases) • It should be accompanied / preceded by • (Preliminary Hazard Analysis • FMEA • Event Tree Analysis) **FTA Procedure** 1. Identify the objective 2. Get familiar with operation and success criteria of the system 3. Define the top-event 4. Define the scope 5. Define resolution 6. Define ground rules 7. Construct the FT 8. Evaluate the FT 9. Interpret and present the results, **Objective** • All stakeholders should agree on what is to be examined • The objective should be stated in written • The objective should refer to a failure of the system in application domain vocabulary • The objective determines the top-event, the scope, the resolution, **A FT without NOT is called "coherent"** • "A system can never get better if more components fail" • Some NOTs are virtual (e.g. in transcriptions of voter, XOR, ...) • If FT events are sudden events • How can their occurrence be negated? • In this case, NOT makes no sense • If FT events are states / conditions / predicates / logical propositions • Negation makes sense • BDD algorithm can handle negated variables • Minimal cut sets → Prime implicants (may contain negated events), **Generalization vs. Causation Gates** • **Sometimes gates suggest causality** • Electrical short circuit OR defective gas tube ⇒ fire • **Sometimes gates suggest generalization / decomposition** • Engine defective OR tire defective = car defective • **In original FT standards no distinction, some researchers do distinction** • Sometimes, two pairs of AND / OR are proposed [Gorski] • Some say that AND means causation and OR means decomposition. **Modules are independent sub-trees But: Technical components often influence each other. Temporal Functions** • Events can be understood as propositions that are true or false at each given point of time • Supplying time functions instead of constant probabilities allows to plot reliability/availability function for the complete system • (Events can be exponentially distributed, Weibull distributed etc. • Useful in combination with Markov analysis • Mission time for each event or sub-component specifies time) • Important special case: Exponential distribution, **By restructuring the Boolean formula, repeated events can be avoided. Directed Acyclic Graphs (instead of trees) eliminate repeated events, A namespace concept is desirable (implemented in Component Fault Trees), Many attempts have been made to bring special cases of dependency into FTA. At a certain point, state-based models (e.g. Markov chains) are a better fit (but analysis is usually much slower!), Formal integration with software / systems modeling is desirable, but not yet achieved to a satisfactory degree, Applying FTA to software controlled systems is still an issue. Formal semantics** FTs are intuitive, but unclear semantics is an issue when • (constructing FTs automatically • integrating FTs with other kinds of models • using dynamic extensions) • Some recent research work about formalizing FTA • (formalizing the meaning of gates • proving that FT is complete and consistent) • Different formalisms used • (Z (algebraic specification) • Interval Temporal Logic, Duration Calculus (temporal/real time logic) • Translation into Markov chains and different kinds of probabilistic Petri nets)

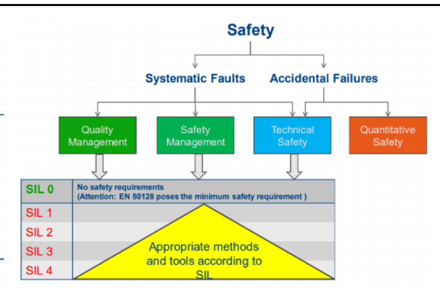
Hardware Reliability• Failures are a result of physical degradation • When the faulty component is substituted, the reliability becomes the initial value of this component • The reliability of the system does not exceed the initial value of the system reliability through the substitution of components with new components • Hardware reliability is determined by fairly constant parameters**, Software Reliability**• Failures are a result of design errors that are contained in the product from the start and appear accidentally • After fault correction the system reliability exceeds its initial value (under the assumption that no additional faults are introduced) • Faults that are introduced during debugging decrease reliability • Reliability parameters are assumed to vary. A relevant measure for reliability is the **Mean Time To Failure** (MTTF) or Mean Time Between Failure (MTBF) • The MTTF resp. **MTBF** defines the mean value of the lifetime resp. the mean value for the time interval between two successive failures**, Reliability function R(t)** • F(t) gives the probability that at time t at least one failure has occurred; thus R(t) = 1 - F(t) is the probability that at time t no failure has occurred yet, The **probability density** f(t) describes the modification of the probability that a system fails over time, The **failure rate** is the relative boundary value of failed entities at time t in a time interval that approximates zero, referring to the entities still functional at the beginning of the time interval. **Musa's Execution Time Model** • A software system fails due to errors in the software randomly at t1, t2, (t here refers to execution time, i. e. CPU-seconds), **Least squares** • Target: Define parameters in such a way that the sum of the squares of the deviations between the calculated and the observed values becomes minimal. If Fi refers to the value of the empirical distribution function at point t, **Maximum-Likelihood-Method** • Target: Choose parameters in such a way that the probability is maximized to produce a "similar" observation to the present observation. The probability density has to be known, **Model Selection based on Failure Observations:** U-Plot-Method • Prequential-Likelihood-Method • Holdout-Evaluation. **U-Plot** • Graphic method that tests whether a distribution function can be accepted with regard to the present observation. The **Prequential-Likelihood-Method** compares the suitability of two distribution functions under consideration with regard to a given failure observation. **Holdout Evaluation:** Only parts of the failure data are used for model calibration. The remaining data are used to judge the prediction quality of the calibrated model. The **prediction quality** of the Weibull distribution is however worse than that of the exponential distribution. The sum of the deviation squares for the failure times T7 to T10 is 0,00446 for the Weibull distribution; for the exponential distribution is only 0,00210. We might prefer to use the exponential distribution in order to avoid „over-calibration", **Software Reliability** can be adequately measured and predicted using appropriate models • The use of stochastic reliability models requires some knowledge w.r.t. the underlying mathematics • Appropriate tools are a precondition for the successful use of reliability models.

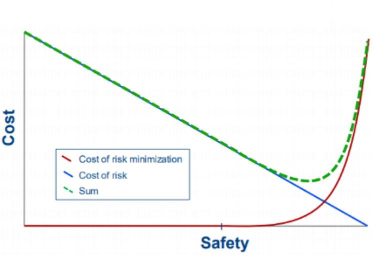
Symbolic model checking is a formal verification technique • It is mathematically founded • Formal documents are required as a starting point • It yields complete statements • The results are reliable **• Aims •** Verification of defined properties of the item under consideration (software, specification) or ... • Generation of information, where the required property does not hold. **CTL (Computation Tree Logic)** • Forward-time operators • G: globally, invariantly • F: sometime in the future • X: next time • U: Until • Path quantifiers • A: all computation paths • E: some computation path (exists) • In CTL, a path quantifier has to be placed right before a forward-time operator (e.g. AG f or EF f).









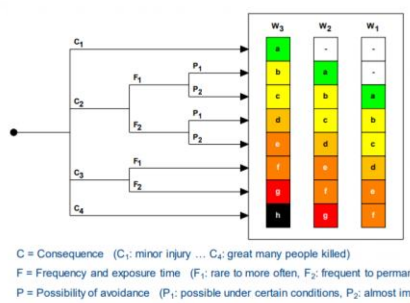


AND-Gate: $P_{out} = \prod_{i=1}^{n} P_i$

OR-Gate: $P_{out} = 1 - \prod_{i=1}^{n} (1 - P_i)$

NOT-Gate: $P_{out} = 1 - P_{in}$ (only

XOR-Gate: $P_{out} = \sum_{i=1}^{n} P_i \cdot \prod_{\substack{j=1 \\ j \neq i}}^{n} (1 - P_j)$



Necessary minimal risk reduction	Safety integrity level
-	No safety requirements
a	No special safety requirements
b, c	1
d	2
e, f	3
g	4
h	An E/E/PE SRS* is not sufficient

* Electrical/Electronic/Programmable Electronic safety-related system

C = Consequence (C_1: minor injury … C_4: great many people killed)

F = Frequency and exposure time (F_1: rare to more often, F_2: frequent to permanent)

P = Possibility of avoidance (P_1: possible under certain conditions, P_2: almost impossible)

W = Probability of unwanted occurrence (W_1: very slight probability, W_2: slight probability, W_3: relatively high probability)

Note: Risk graph concept used to be defined in DIN 19250, which has been withdrawn in favor of DIN EN 61508-5:2010

Risk parameter	Classification
Consequence C	C_1: **Minor injury** C_2: **Serious permanent injury** to one or more persons; **death of one person** C_3: **Death of several people** C_4: **Great many people killed**
Frequency and time of exposure to the hazardous zone F	F_1: **Rare to more often** exposure to the hazardous zone F_2: **Frequent to permanent** exposure to the hazardous zone
Possibility of avoiding the hazardous event P	P_1: **Possible** under certain conditions P_2: **Almost impossible**
Probability of the unwanted occurrence W	W_1: A **very slight probability** that the unwanted occurrences will happen and **only a few unwanted occurrences** are likely W_2: A **slight probability** that the unwanted occurrences will happen and **few unwanted occurrences** are likely W_3: A **relatively high probability** that the unwanted occurrences will happen and **frequent unwanted occurrences** are likely

$$F_S(t) = F_{K_1}(t)\,F_{K_2}(t)\,F_{K_3}(t)\,\dots\,F_{K_n}(t) = \prod_{i=1}^{n} F_{K_i}(t)$$

$$R_S(t) = 1 - F_S(t) = 1 - \prod_{i=1}^{n} F_{K_i}(t) = 1 - \prod_{i=1}^{n} (1 - R_{K_i}(t))$$

Markov-Diffrential Equation

$$\frac{dP_1(t)}{dt} = -(\lambda_{11}+\lambda_{21}+\lambda_{31})\cdot P_1(t) + \mu_1\cdot P_2(t) + \mu_2\cdot P_3(t) + \mu_3\cdot P_4(t)$$

$$\left(10^{-5}\frac{deaths}{person\cdot year}\right)\cdot 1person = \frac{10^{-5}deaths}{8760h} \approx 1{,}142\cdot 10^{-9}\frac{deaths}{h}$$

The Exponential Distribution

- Life distribution: $F(t) = 1 - e^{-\lambda t}$
- Density function: $f(t) = \lambda\, e^{-\lambda t}$
- Reliability function: $R(t) = 1 - F(t) = e^{-\lambda t}$
- Failure rate: $\lambda(t) = \lambda$
- MTTF: $\bar{T} = \dfrac{1}{\lambda}$

$$RF_{total} = \sum_{hazard_j} A_i \cdot F_i \cdot \frac{N_{end,i}}{N_{all}} \cdot HR_i$$

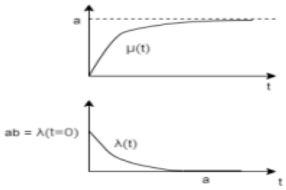
HR_i	[1/t]	Rate, with which hazard i occurs
$S = A_i \cdot F_i$	[1]	Extent of damage (Cost of hazard)
A_i	[1]	Probability that hazard i will result in an accident (typically from event trees, fault trees)
$F_i = N_{end} \cdot P$	[Persons]	Probability P that death or injury is caused by an accident multiplied by the number of endangered persons
N_{end}	[Persons]	Number of the actually endangered persons in danger area of hazard i
N_{all}	[Persons]	Total number of system users

$$IRF_i = \sum_{hazard_j} NP_i \cdot \left| HR_j \cdot (D_j + E_{ij}) \cdot \sum_{accidents_k} A_{jk} \cdot F_{jk} \right|$$

NP_i	[1/t]	Usage profile (number of usages per time)
HR_j	[1/t]	Rate, with which hazard j occurs
D_j	[t]	Duration of hazard j
E_{ij}	[t]	Time during which individual i is exposed to hazard j
A_{jk}	[1]	Probability that hazard j will result in accident k (typically from event trees, fault trees)
$F_{jk} = N_{end} \cdot P_k$	[Persons]	Probability P_k that death or injury is caused by accident k multiplied by the number of endangered persons

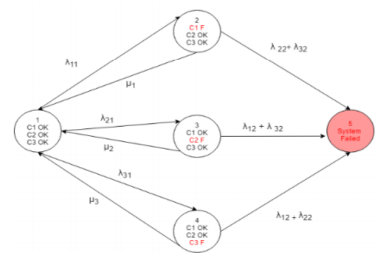
Weibull

- Life Distribution : $F(t) = 1 - e^{-(\lambda t)^\beta}$; $\lambda, \beta > 0$
- or:
  $F(t) = 1 - e^{-\frac{1}{\alpha} t^\beta}$; $\alpha, \beta > 0$, d. h. $\frac{1}{\alpha} = \lambda^\beta$
- Density:
  $f(t) = \dfrac{dF(t)}{dt} = \lambda\, \beta\, (\lambda\, t)^{\beta - 1}\, e^{-(\lambda t)^\beta}$
- Reliability: $R(t) = e^{-(\lambda t)^\beta}$
- Failure rate:
  $\lambda(t) = \dfrac{f(t)}{R(t)} = \lambda\, \beta\, (\lambda\, t)^{\beta - 1}$



1. Correctness has a binary character – True 2. An artefact is not consistent to its spec, if it's not correct - True 3. If there are no defects the program is correct – True 4. System is affected by human – True 5. The environment and people can be part of a system - True 6. A system can have a low robustness even if its correct – True 7. A correct system can have low robustness – True 8. Robustness is depending on the specification – True 9. Safety allows the presence of risks – True 10. It can always be decided whether an artefact is correct/not – True 11. A system with a low technical safety can be a tech sys – True 12. Equipment may be available but not reliable – True 13. High efficiency achieves high safety – True 14. Safety can be measured – True 15. A/Each Fault leads (always) to a Failure – True 16. Every failure should have a timestamp – True 17. An error can cause a failure lead – True 18. Errors can lead to faults – True 19. Every failure is caused by faults – True 20. MTTR is a reliability measure – True 21. High reliability always leads to high availability - True 22. Reliability says its highly available, but availability may or may not be reliable – True 23. If a system is reliable then it is available - True 24. Low robustness means it doesn't handle failure condition – True 25. Robust system need not be correct – True 26. Correctness is property of code – True	1. Robustness has binary character. (Gradual character) - False 2. Robustness is a property only of the implementation (property of specification) - False 3. Correct system is always safe – False 4. Robust system is always safe – False 5. Safe system is always reliable – False 6. Safe system is always available - False 7. Available system is safe – False 8. Technical system cannot influence environment or ppl – False 9. Technical system is only defined for technical system – False 10. Safety is absence of Risk – False 11. Hazard always leads to accident (may or may not) – False 12. Correctness is always defined – False 13. Can a reliable system be incorrect – False 14. When analysing a system, people are never taken into account – False 15. A system with correct specification is always available – False 16. High reliability always leads to high availability – False 17. High availability always leads to high reliability – False 18. A/Every failure leads (always) to a Fault 19. A failure is always the cause of a fault. 20. Correct SW always guarantees a safe sys - False 21. Can a reliable sys be incorrect? - False

C/E Net	P/T Nets (*P/T net, Place/Transition Net*)	Pr/T Nets (*Predicate/Transition Net*)	Timed Petri Net
Objects, respectively tokens, are **Boolean** data type	Places can obtain **more than one** token (in C/E nets only one token)	Tokens are **colored**: They can be individualized, they do have a description (unlike other tokens)	Similar as in C/E net
Transitions are interpreted as **Events**	Transitions must release or add as many tokens when firing as the **weights** that are given on arcs	Transitions have a **firing condition** and a **firing effect.**	There is **firing delay associated** with each **transition.** This delay specifies the time that the transition has to be enable, before it can actually fire
Places are denoted as **conditions.** **Binary condition**: Each place is allowed to receive exactly one or no token	Places have a **capacity** (maximum number of tokens that may lie in one place): if it is to be bigger than 1, this will be denoted as »K = ...« at the place	Places contain a number of tokens that can be integers or variables	Similar as in C/E nets



$$\frac{dS_1(t)}{dt} = -(\lambda_{11}+\lambda_{21}+\lambda_{31}).S_1(t) + \mu_1.S_2(t) + \mu_2.S_3(t) + \mu_3.S_4(t)$$

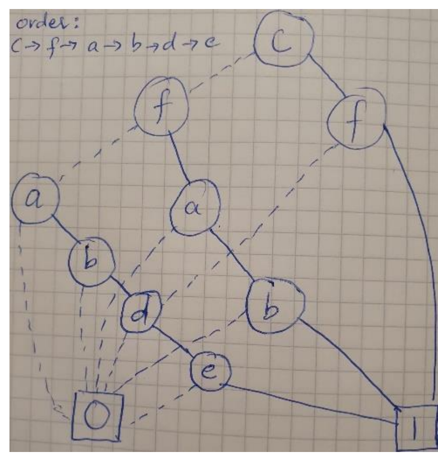
$$\frac{dS_2(t)}{dt} = \lambda_{11}.S_1(t) - (\mu_1+\lambda_{22}+\lambda_{32}).S_2(t)$$

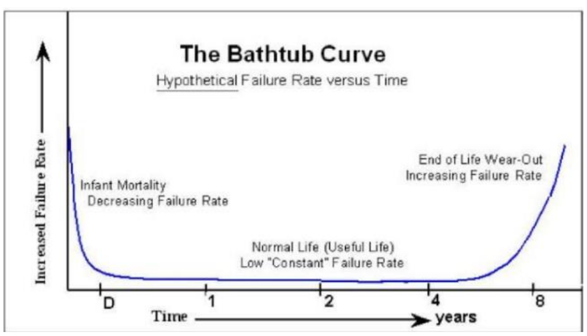
$$\frac{dS_3(t)}{dt} = \lambda_{21}.S_1(t) - (\mu_2+\lambda_{12}+\lambda_{32}).S_3(t)$$

$$\frac{dS_4(t)}{dt} = \lambda_{31}.S_1(t) - (\mu_3+\lambda_{12}+\lambda_{22}).S_4(t)$$

$$\frac{dS_5(t)}{dt} = (\lambda_{22}+\lambda_{32}).S_2(t) + (\lambda_{12}+\lambda_{32}).S_3(t) + (\lambda_{12}+\lambda_{22}).S_4(t)$$

$$S_1(t) + S_2(t) + S_3(t) + S_4(t) + S_5(t) = 1$$

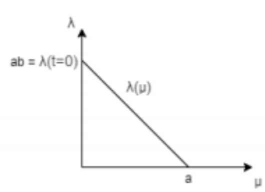




The conditional probability that a system has survived until t and fails within Δt is: $1 - \frac{R(t+\Delta t)}{R(t)}$

$$\frac{F(t+\Delta t) - F(t)}{1 - F(t)} \Rightarrow \frac{F(250+50) - F(250)}{1 - F(250)} = 0.25$$

250h | 300h



a = Total Failures | μ = No of Failures

λ = Rate of Failures

$\lambda(t) = \lambda_0 . e^{-\frac{\lambda_0}{a}t}$; **unit is 1/cpu sec** ;**failure rate after t=x CPU-seconds??**

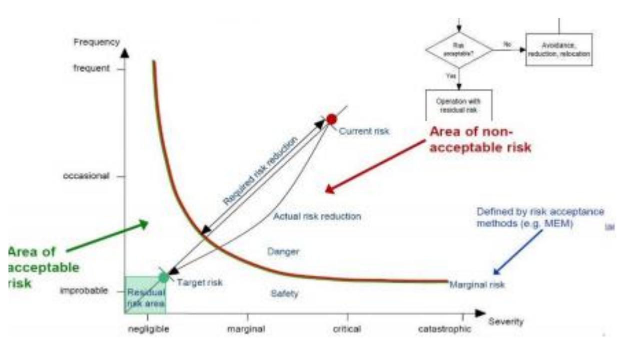
$\mu(t) = a . (1 - e^{-\frac{\lambda_0}{a}t})$; **unit is failures** ;How many **failures will have occurred??**

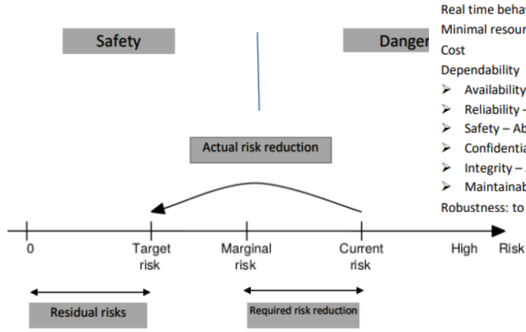
$\Delta t = \frac{a}{\lambda_0} \ln \frac{\lambda_{current}}{\lambda_{target}}$; **unit is CPU sec** ;additional execution time will be necessary?

$\Delta\mu = a . (\frac{\lambda - \lambda_z}{\lambda_0})$; **lambda is current; lambda z is target; lamda 0 is initial ; unit is failures** Additional failures will occur??

$\lambda(\mu) = \lambda_0 . (1 - \frac{\mu}{a})$; **mew is intermediate failures; a is total failures ; unit is 1/cpu sec** ;failure rate after intermediate(x) failures?

$t = a . \frac{\ln(\lambda_0) - \ln(\lambda(t))}{\lambda_0}$; **t=time;** total execution time? unit: cpu sec





Real time behavior: Deadlines must be met as required by the environment.
Minimal resource consumption: Memory, energy, dimensions, weight
Cost
Dependability
- Availability – Readiness for correct service
- Reliability – Continuity of correct service
- Safety – Absence of catastrophic consequences on users and environment
- Confidentiality – Absence of unauthorized disclosure of information
- Integrity – Absence of improper system state alterations
- Maintainability – Ability to undergo repairs and modifications

Robustness: to deliver an acceptable behavior, also in exceptional situations.