Technische Universität Kaiserslautern

0101seda010100

software engineering dependability

Safety and Reliability of Embedded Systems

(Sicherheit und Zuverlässigkeit eingebetteter Systeme)

Safety and Reliability Analysis Models: Overview

Content

- Classification
- Hazard and Operability Study (HAZOP)
- Preliminary Hazard Analysis (PHA)
- Event Tree Analysis
- Failure Modes Effects and Criticality Analysis (FMECA) (DIN 25448, IEC 812)
- Reliability Block Diagrams (IEC 61078)
- Fault Tree Analysis (DIN 25424, IEC 61025)
- Markov Analysis (IEC 61165)
  - Markov Chain
  - Markov Processes
- Petri Nets
  - Condition/Event Petri nets
  - State/Transition Petri nets
  - Predicate/Transistion Petri Nets / Coloured Petri Nets
  - Timed Petri Net Types
    - SPN
    - GSPN
    - DSPN

Classification of Safety / Reliability Analysis Techniques

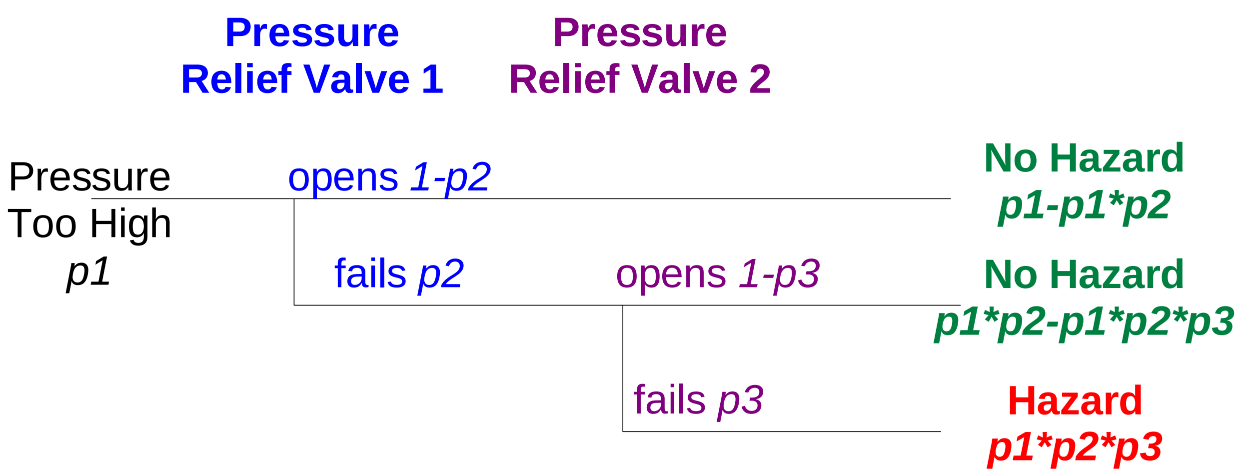
- Focused Property
  - Safety, Reliability, Availability...
- Application Area
- Scope
  - Product / Process, HW / SW, System / Component
- Process Phase
- Search Direction
  - Inductive / Deductive
- Degree of Formality
- Representation
  - Textual, Graphical, Tabular
- Model based: Combinatorial vs. State-Based

HAZOP / PHA

- 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

Event Tree Analysis

- Forward-searching technique with graphical representation
- Search consequences to given hazard, depending on conditions



FMECA

- The Failure Mode, Effects and Criticality Analysis (FMECA) is a preventive method for the identification of problems, their risks and effects
- FMECA has the following 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)

FMECA

- 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

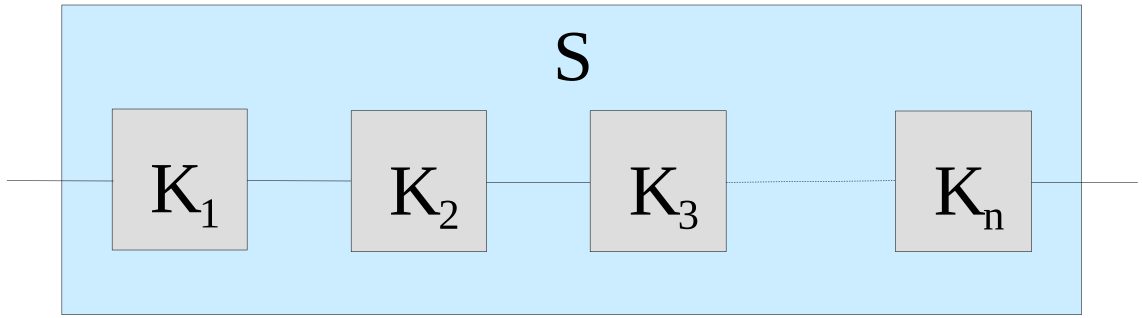
- If for the three influencing factors a value between 1 and 10 is used (1= no risk, minor occurrence; 10 = high risk, high occurrence), the RPN is a value between 1 and 1000
- The risk priority number generates a ranking for the causes of faults
- Causes of faults with a high risk priority number are to be handled with priority

Reliability Block Diagrams

- 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

Reliability Block Diagrams
Serial Connection

- n serial connected components K_i. The system S fails if one of the components fails



$$R_S(t) = R_{K_1}(t)\, R_{K_2}(t)\, R_{K_3}(t) \ldots R_{K_n}(t) = \prod_{i=1}^{n} R_{K_i}(t)$$

- Example:

  Two components with $R_1 = R_2 = 0{,}8$: $\underline{R_S = 0{,}64}$

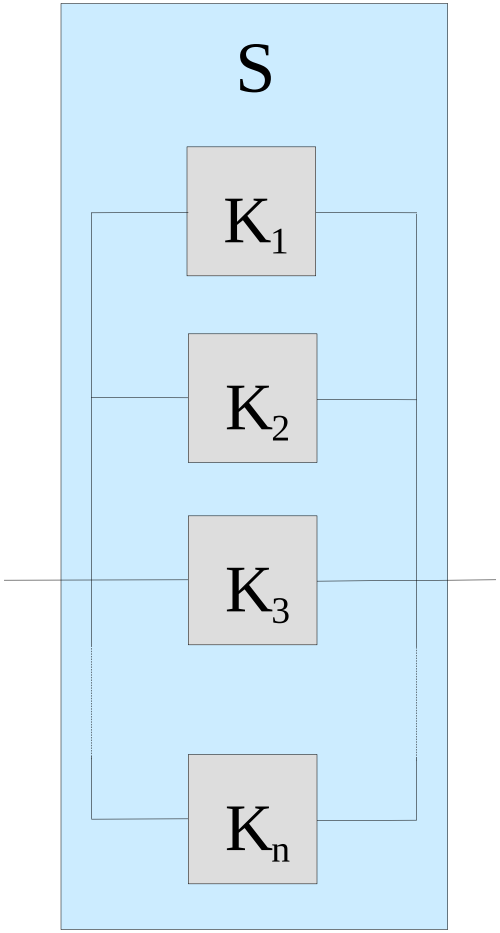
- n parallel connected components K_i. The system S fails if all components fail

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

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

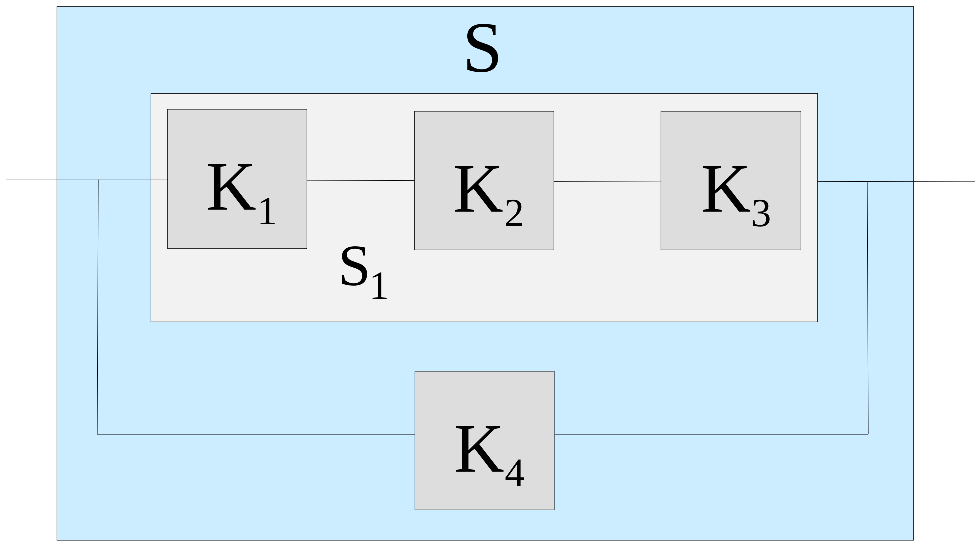
- Example:

  Two components with $R_1 = R_2 = 0{,}8$: $\underline{R_S = 0{,}96}$



Reliability Block Diagrams
Combined Serial/Parallel Connection

- Combinations of serial and parallel connections can be solved hierarchically



- Example:

  System S is a parallel connection of the subsystem S_1 with component K_4

  The reliability of the subsystem S_1 is:

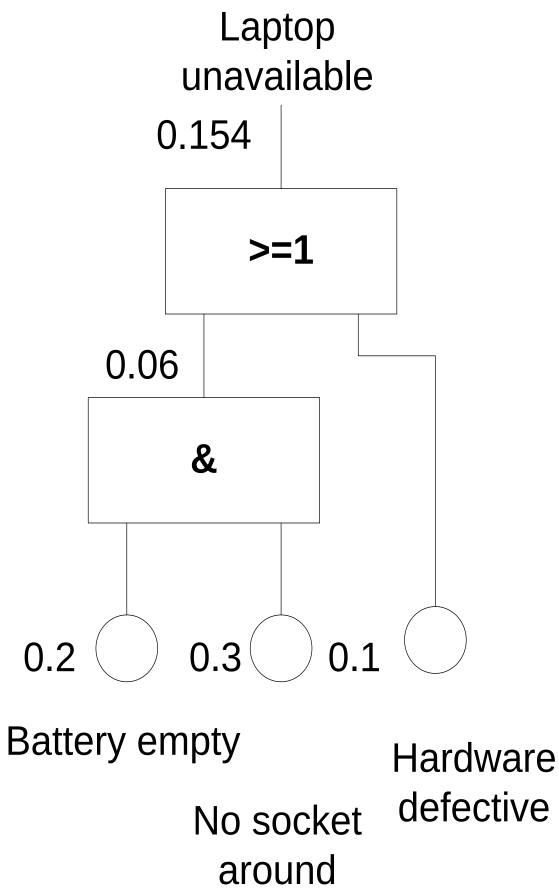
  $R_{S_1}(t) = R_{K_1}(t)\ R_{K_2}(t)\ R_{K_3}(t)$

  The reliability of the system S is:

  $R_S(t) = 1 - [(1 - R_{K_4}(t))\ (1 - R_{S_1}(t))]$

  $\quad\quad = 1 - [(1 - R_{K_4}(t))\ (1 - R_{K_1}(t)\ R_{K_2}(t)\ R_{K_3}(t))]$

  All components have the reliability R = 0,8: $\underline{R_S = 0,9024}$



- 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

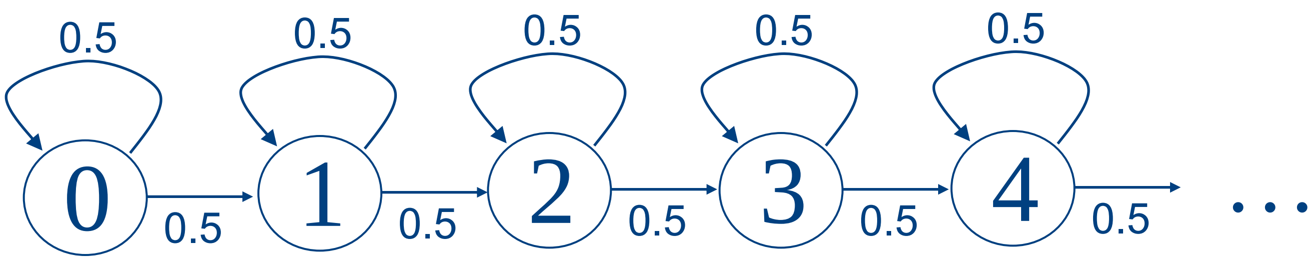
- Markov Analysis
  - Markov Chain
  - Markov Processes
- Petri Nets
  - Condition/Event Petri nets
  - State/Transition Petri nets
  - Predicate/Transistion Petri Nets / Coloured Petri Nets
  - Timed Petri Net Types
    - SPN
    - GSPN
    - DSPN

Markov Modeling

- 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 Chain

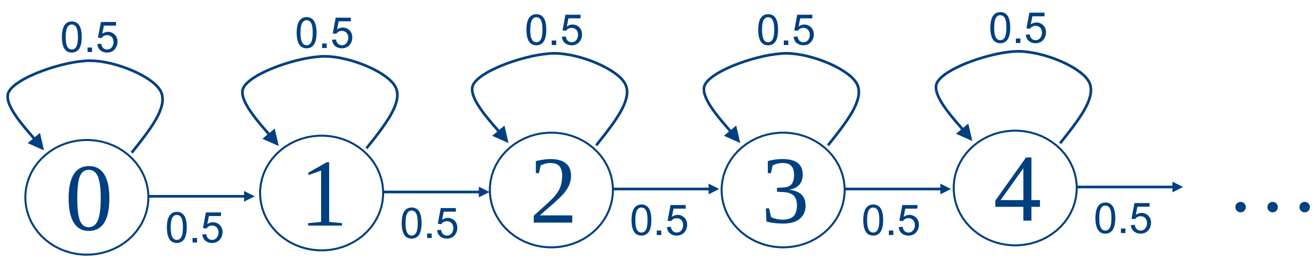
- Markov chains assume that state changes occur at discrete points in time
- Example: Throwing a coin *n* times and counting the results where we got the "front side" as the result of this experiment
- Obviously this can be modeled with the following Markov chain:



- If the current state is *i* then the probabilities to stay at state *i* or to enter state *(i+1)* are both *0.5*. A state change may only occur at the discrete point in time, when the coin is thrown.

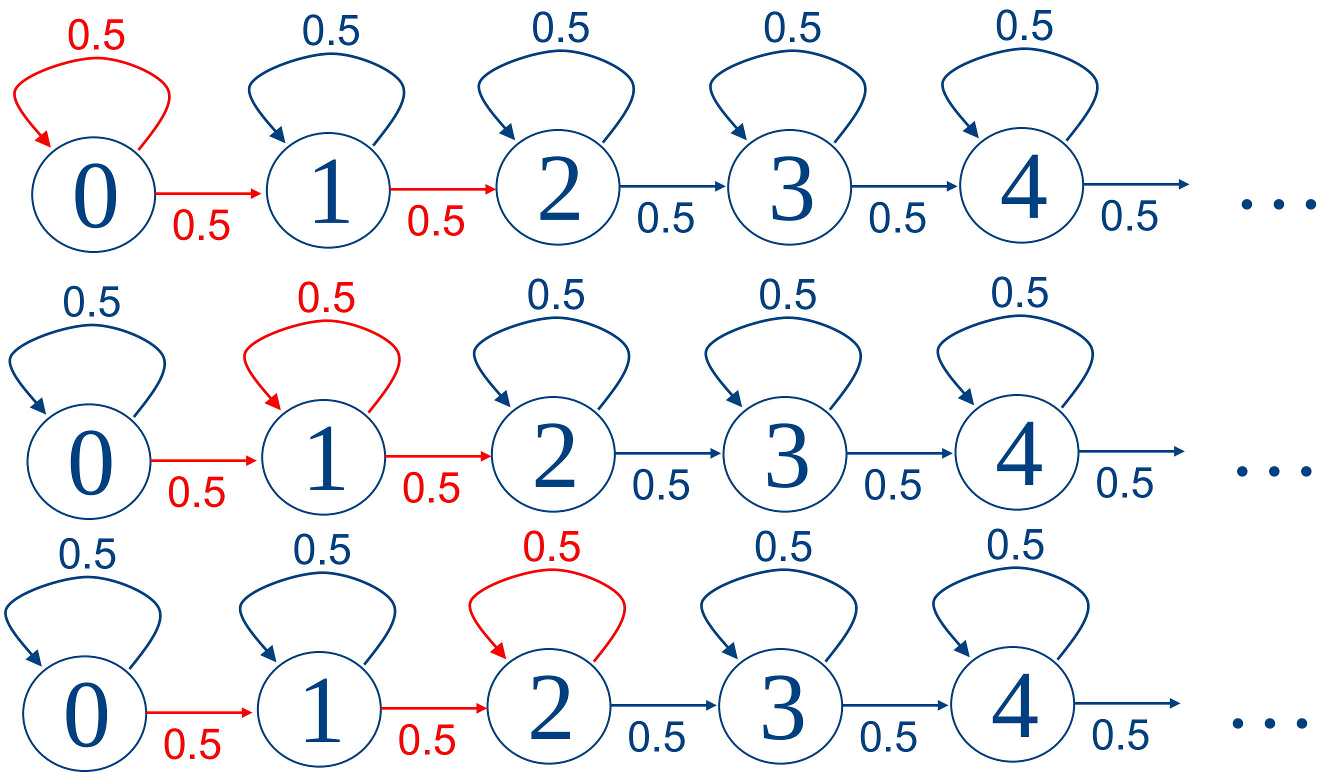
Markov Chain

- What is the probably for 2 “front sides” after throwing the coin twice?
- Answer: There is one path of length 2, that leads into state 2, associated with probability (0.5 * 0.5) = 0.25



Markov Chain

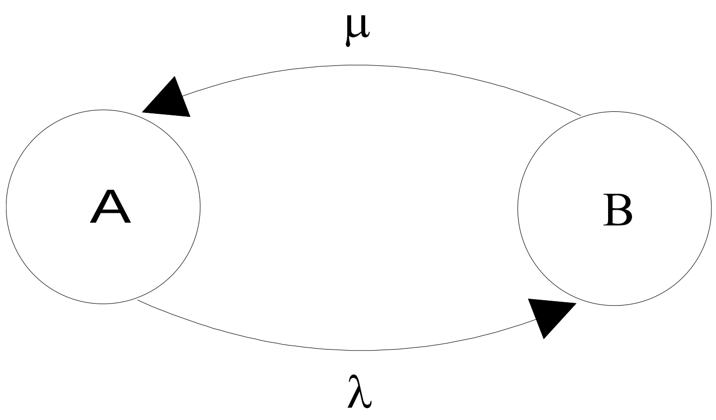
- What is the probably for 2 “front sides” after throwing the coin three times?
- Answer: There are the following paths of length 3, that lead into state 2



Markov Chain

- Each of the three paths has probability (0.5 * 0.5 * 0.5) = 0.125
- The probability for 2 “front sides” after throwing the coin three times is 3 * 0.125 = 0.375

- Markov processes are continuous time models
- Example
  - A system with failure rate λ and repair rate μ is to be analyzed with the aid of a Markov model. The Markov model has the states A and B
  - A is the state where the system is intact. B is the state where the system failed
  - The system changes with the failure rate λ from the intact state into the failed state. With the repair rate μ it changes from the failed state into the intact operation



Markov Processes

$$\frac{dP_A(t)}{dt} = -\lambda P_A(t) + \mu P_B(t)$$

$$\frac{dP_B(t)}{dt} = \lambda P_A(t) - \mu P_B(t) = -\frac{dP_A(t)}{dt}$$

$$P_A(t) + P_B(t) = 1$$

$$P_A(t) = \frac{\mu}{\mu + \lambda} + \left(c - \frac{\mu}{\mu + \lambda}\right) e^{-(\mu + \lambda)t}$$

$$P_B(t) = 1 - P_A(t) = 1 - \left[\frac{\mu}{\mu + \lambda} + \left(c - \frac{\mu}{\mu + \lambda}\right) e^{-(\mu + \lambda)t}\right]$$

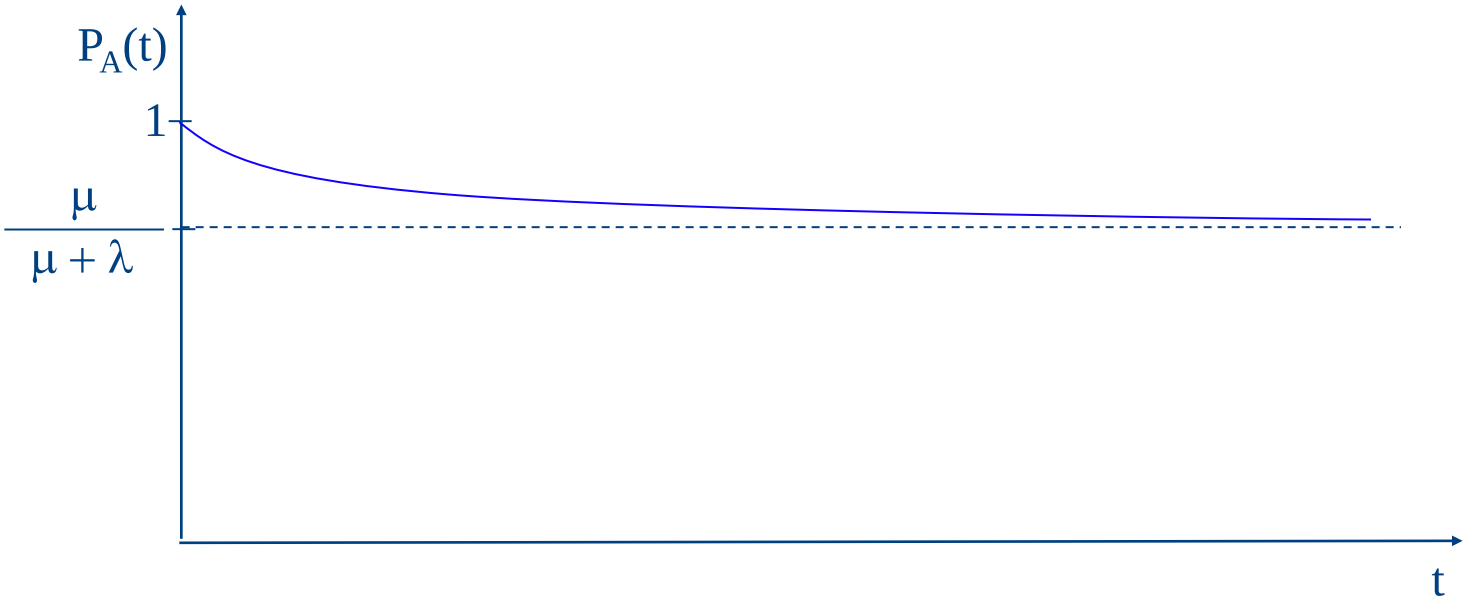
Markov Processes

- For t towards infinite one gets the steady state of the system

$$\lim_{t \to \infty} P_A(t) = \frac{\mu}{\mu + \lambda}$$

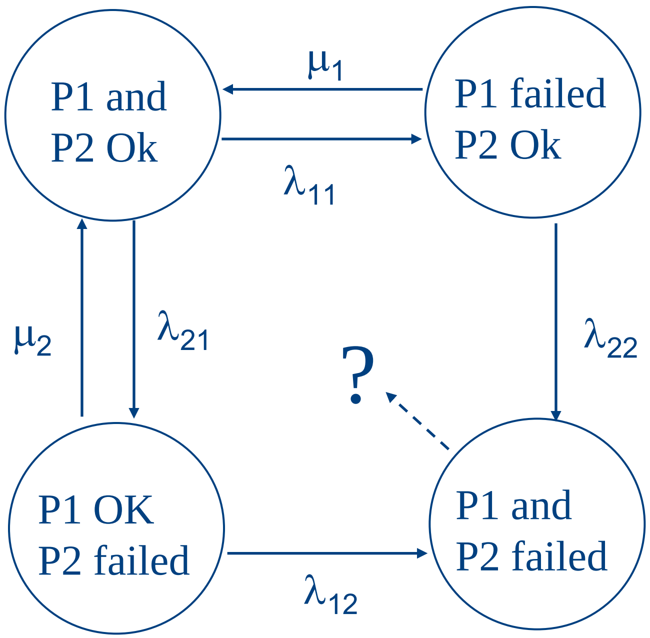
$$\lim_{t \to \infty} P_B(t) = 1 - \frac{\mu}{\mu + \lambda}$$

- If the repair rate is high compared to the failure rate the probability that the system is intact approaches 1
- If the repair rate is low compared to the failure rate the probability that the system is intact approaches zero



$P_A(t)$
1
$\frac{\mu}{\mu+\lambda}$
t

- Let's assume we have to model a system that uses two pumps to pump water to a higher level. In normal operating mode each pump runs at 50% of its maximum power. The remaining pumps takes over the complete load, if one pump fails and thus gets additional stress, which increases its failure probability. How could that be modeled?

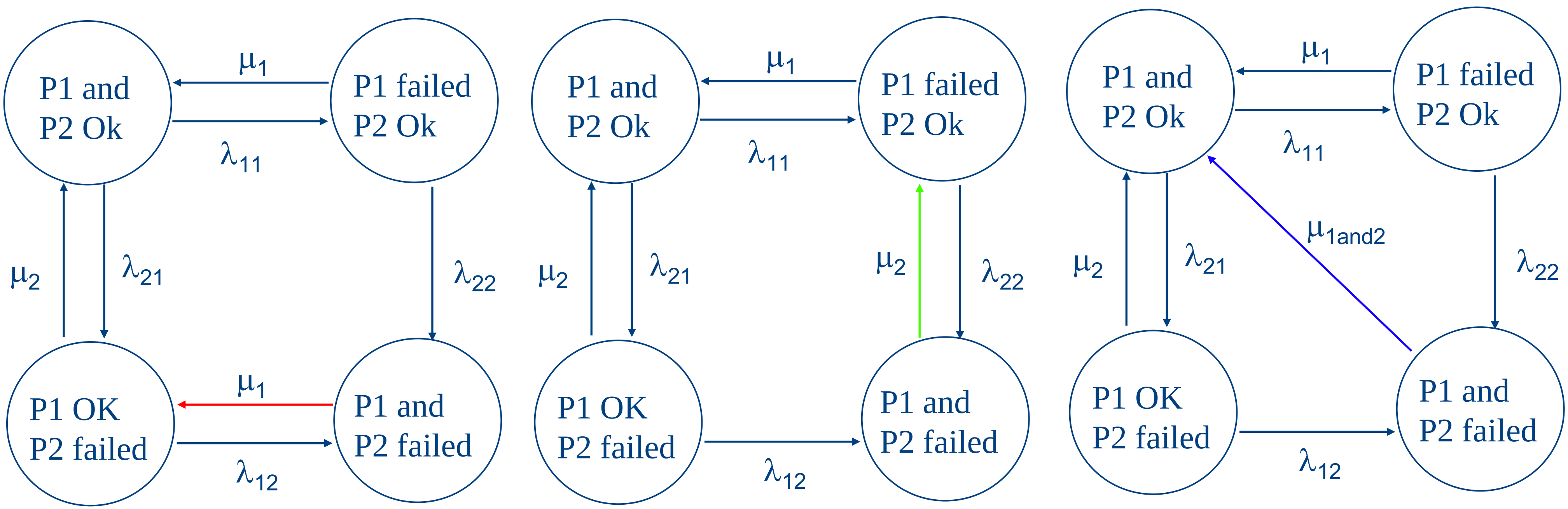


$\lambda_{12} > \lambda_{11}$

$\lambda_{22} > \lambda_{21}$

Markov Processes

- Alternative repair strategies when both pumps have failed:
  - Repair P1 then switch on again,
  - Repair P2 then switch on again,
  - Repair P1 and P2 and then switch on again.



- Petri Nets
  - Condition/Event Petri nets
  - State/Transition Petri nets
  - Predicate/Transistion Petri Nets / Coloured Petri Nets
  - Timed Petri Net Types
    - SPN
    - GSPN
    - DSPN

Petri Nets

- The concept of Petri nets has its origin in Carl Adam Petri's dissertation *Kommunikation mit Automaten*, submitted in 1962 to the faculty of Mathematics and Physics at the Technische Universität Darmstadt, Germany
- Various Petri net types, e.g.:
  - Condition/Event Petri nets
  - State/Transition Petri nets
  - Predicate/Transistion Petri Nets / Coloured Petri Nets
  - Timed Petri Net Types
    - Stochastic delay
    - No delay
    - Deterministic delay

Petri Nets

A Petri Net N contains at least *places* (P), *transitions* (T) and a *flow relation* (F) as well as an initial marking (M_0): $N = (P, T, F, M_0)$:

- $P \cap T = \emptyset$
- $F \subseteq (P \times T) \cup (T \times P)$
- $M_0: P \to \mathbb{N}_0$

Condition / Event Petri Nets

- State elements hold either one or no token
  - state elements represent conditions, which can be true or false
  - transition elements are represent local events
- Event is enabled if and only if
  - all its pre-conditions (connected by incoming arcs) are true
  - all its post-conditions (connected by outgoing arcs) are false
- An event occurrence negates its pre- and post-conditions
- Events with overlapping pre-conditions are in conflict
- Events with overlapping post-conditions are in contact

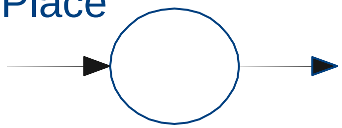
Petri Nets - Condition / Event Petri Nets Fundamentals

- **Petri nets**
  - Directed graph, which consists of two different **kinds** of nodes:
    - **Places** and **Transitions**

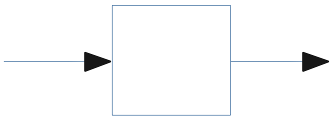
Places
represent a clipboard of information

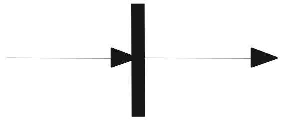
Transitions
describe the processing of information

Place



Transition

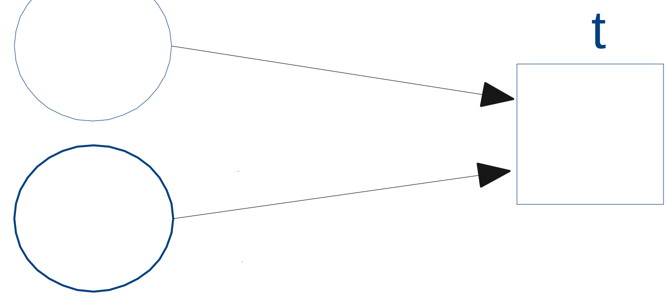




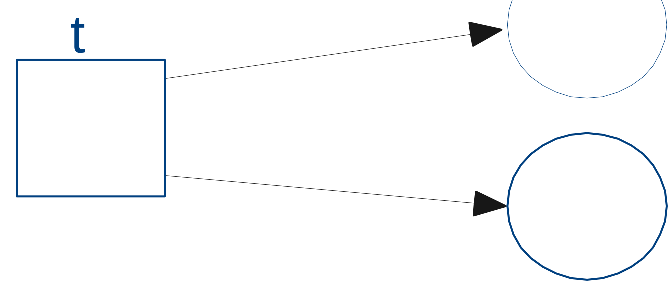
Petri Nets - Condition / Event Petri Nets Fundamentals

- Semantic
  - **Arcs** are only allowed between a node and the other kind of node.
  - **Places** from which arcs run to a transition t are called **Input Places** of t
  - **Places** to which arcs run from a transition t are called **Output Places** of t.

Input places of t



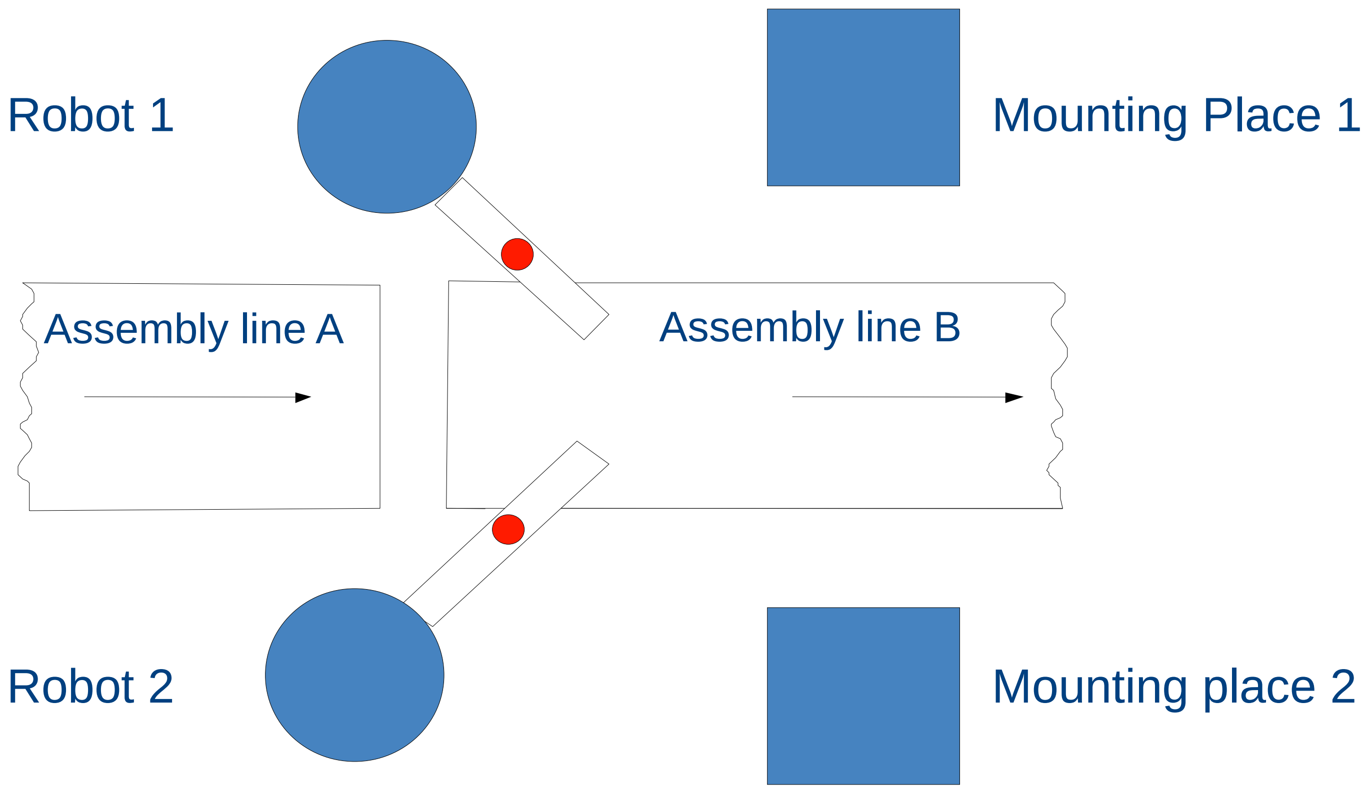
Output places of t



- **C/E Net**
  - Objects, respectively tokens, are of **Boolean** data type
  - Transitions are interpreted as **Events**
  - Places are denoted as **Conditions**
  - Each place is allowed to receive exactly one or no token.
  - Additional firing condition:
  - A transition t can fire if each input space of t contains one token and if each output space of t is empty. When it fires, the token in each input space will be consumed respectively. One token will be assigned to each output space.

Petri Nets - Condition / Event Petri Nets Example

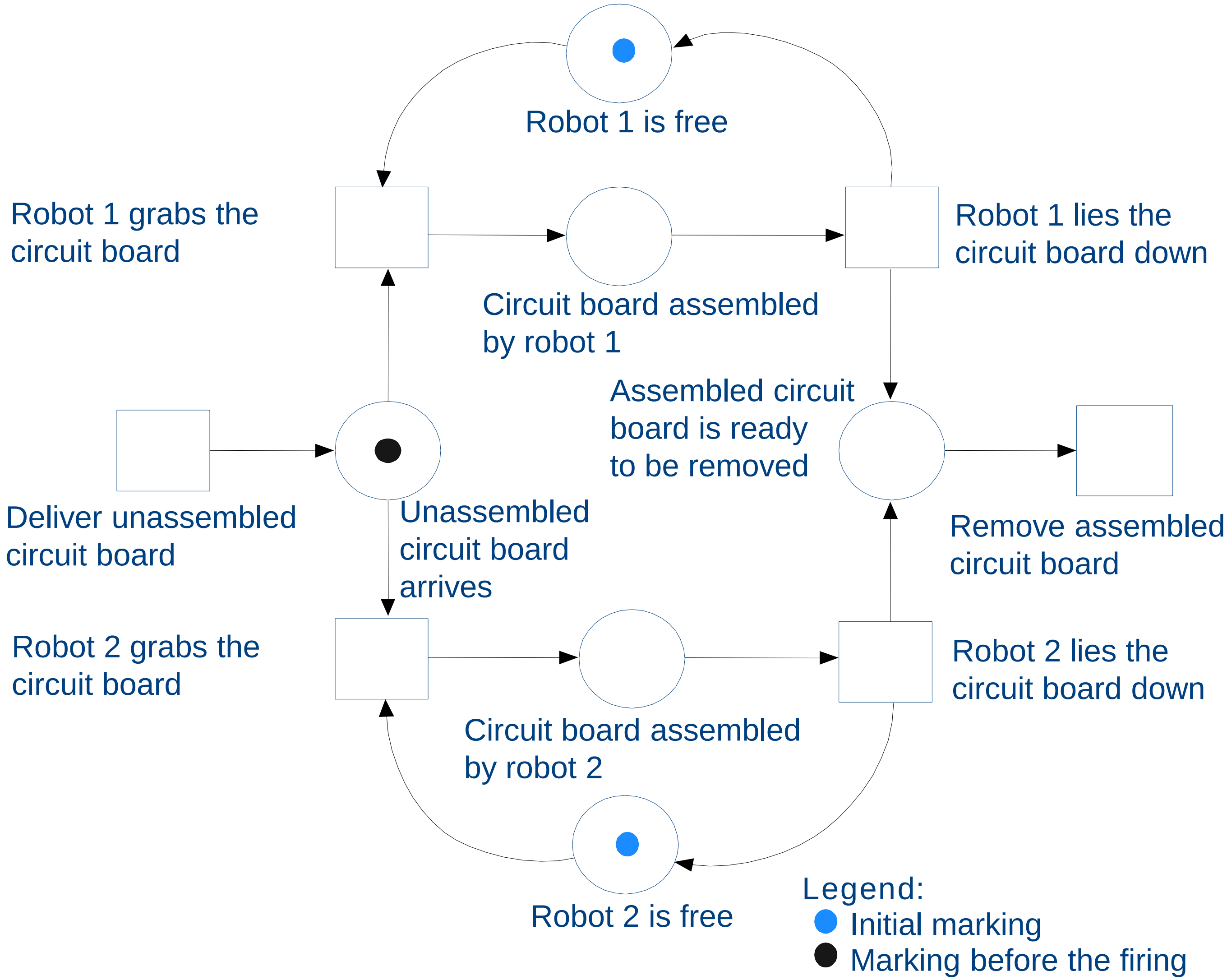
- Example
  - 2 Robots assemble circuit boards with electronic devices, which are delivered on an assembly line A.



Petri Nets - Condition / Event Petri Nets Example

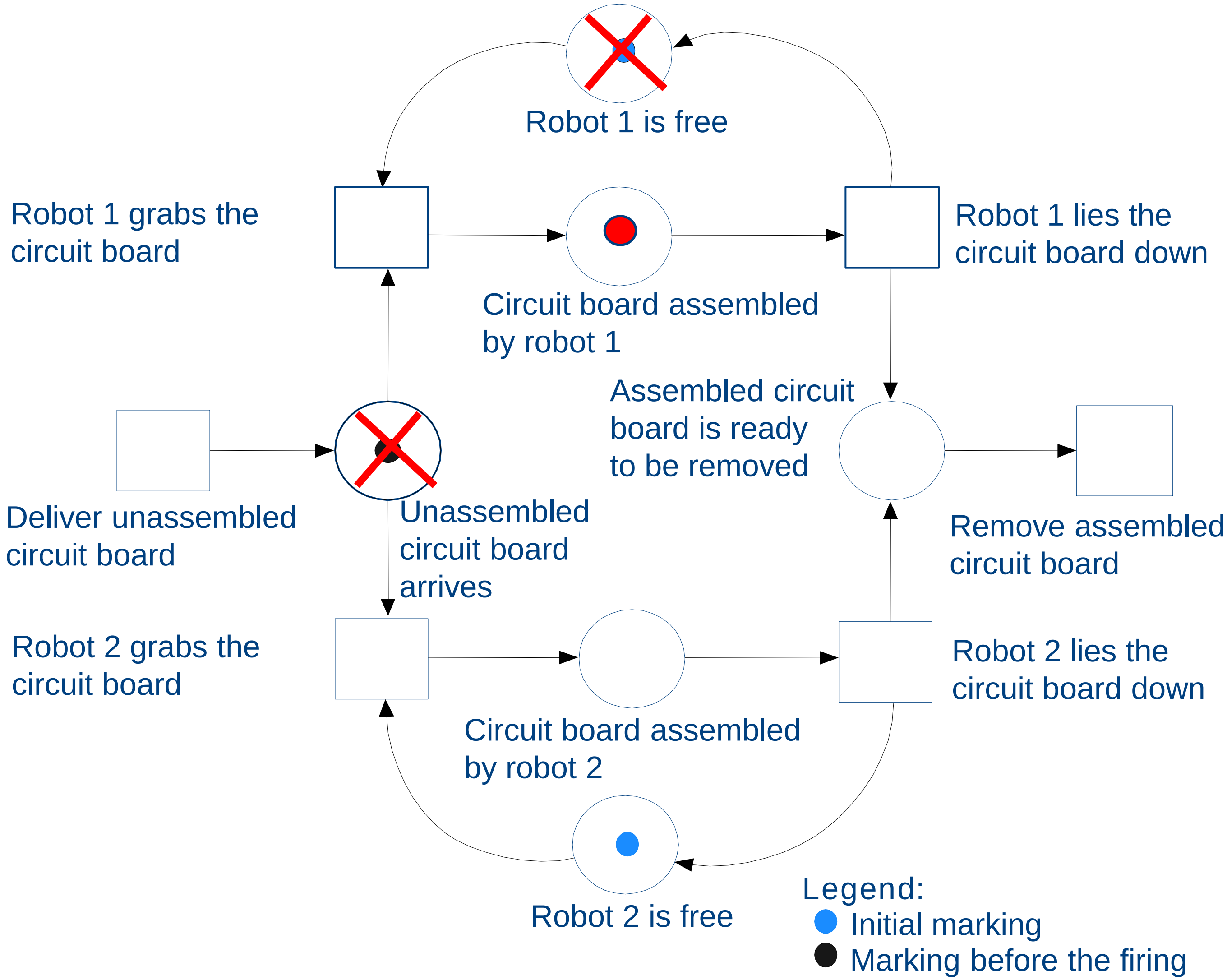
- C/E Net of the assembling robot

Before the firing



Condition/Event Net: Example

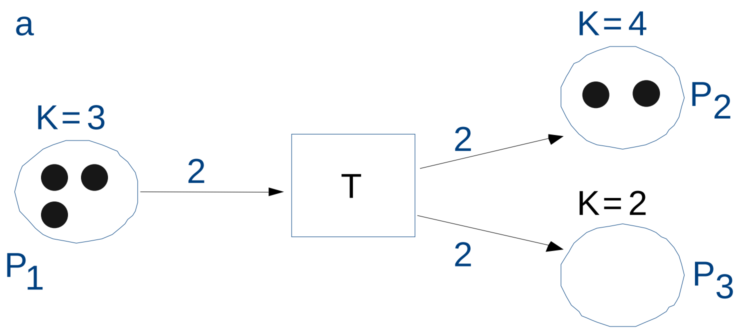
- C/E Net of the assembling robot



After the firing

- **P/T Nets** *(P/T Net, Place/Transition Net)*
  - Places can obtain **more than one** token (in C/E nets only one token)
  - Transitions must release or add as many tokens when firing as the **weights** that are given on the arrows. (in C/E nets only one token)
  - If the capacity of a place is to be bigger than 1, this will be denoted as **»K = ...«** at the place.
  - The capacity defines the maximum number of tokens that may lie in one place.

- Firing with P/T Nets

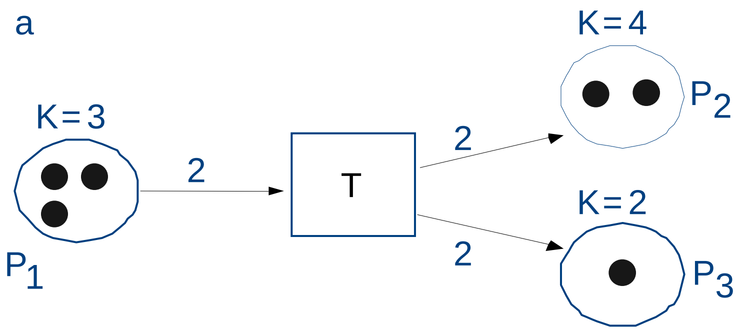


- Before the firing
  - P_1: 3 Tokens
  - P_2: 2 Tokens
  - in P_3: no Token.

- Firing with P/T Nets

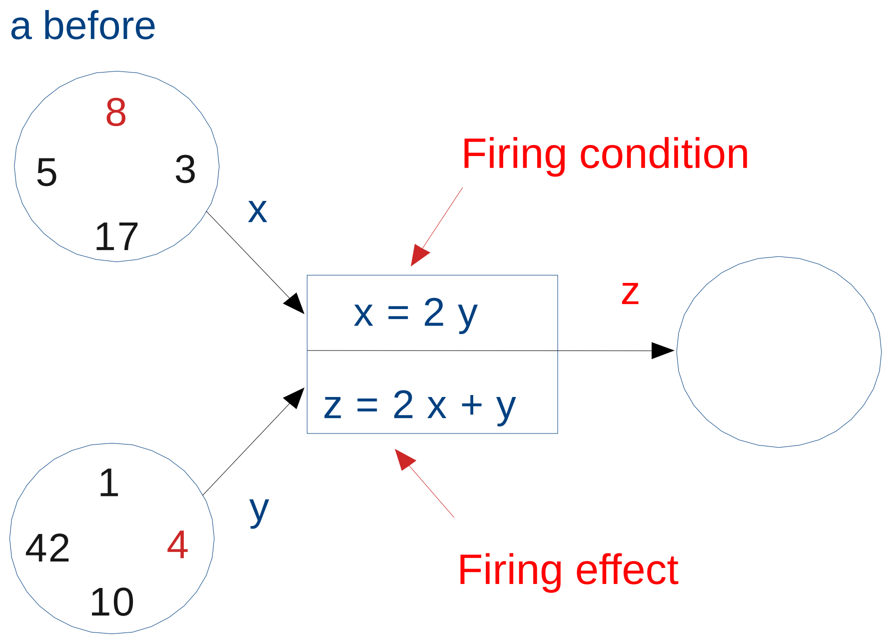
- Before the firing
  - P_1: 3 Tokens
  - P_2: 2 Tokens
  - in P_3: no Token.
- After the firing
  - P_1: 1 Token
  - P_2: 4 Tokens
  - in P_3: 2 Tokens.

- Firing conditions in P/T Nets

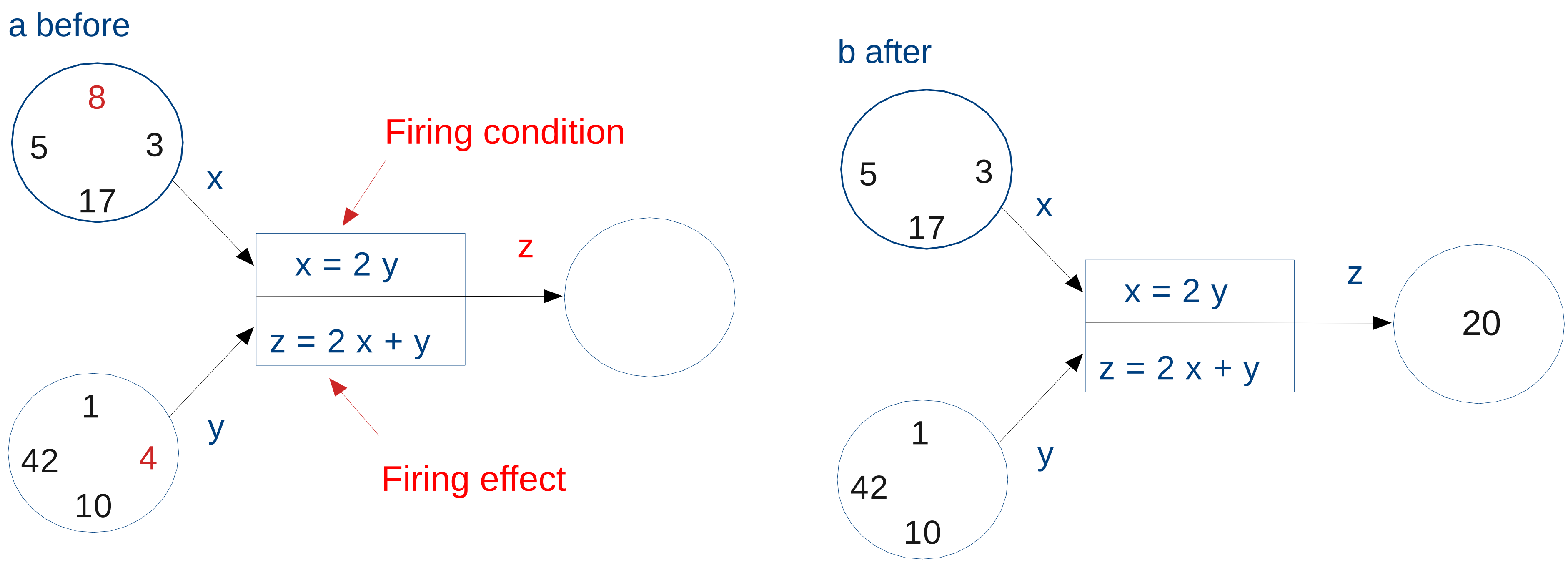


- T cannot fire because 3 Tokens would then lay in P_3
  - This is not allowed due to K = 2 of P_3.

- **Pr/T Nets**
  - Apply **individual, »colored« tokens**
  - C/E and P/T Nets apply only »**black**« tokens, which are all the same

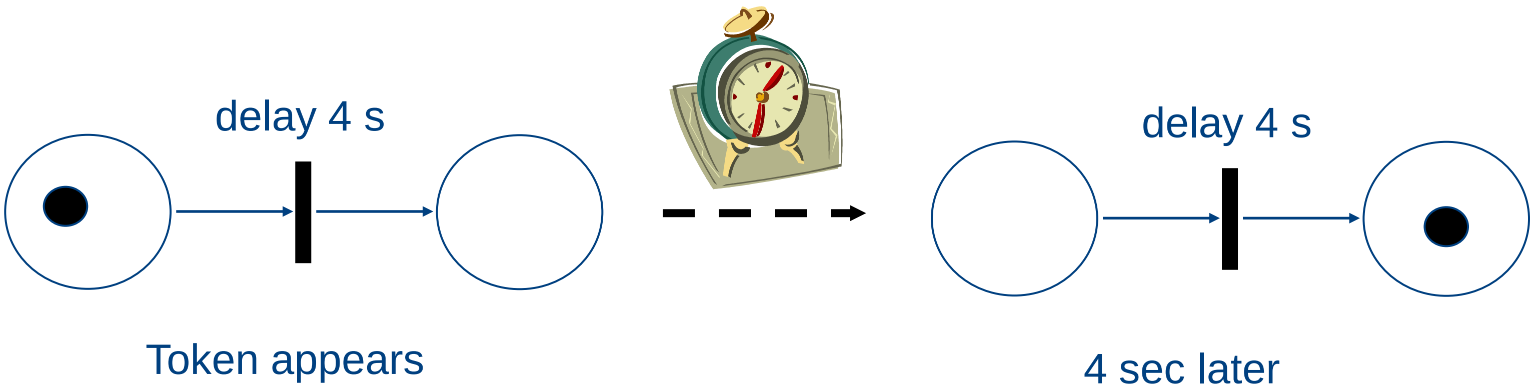


- **Pr/T Nets**
  - Use **individual, »colored« tokens**
  - C/E and P/T Nets use only »**black**« tokens, which are all the same



Petri Nets
Timed Petri Net Types: Simple approach

- To study performance and dependability issues of systems it is necessary to include a timing concept into the model.
- There are several possibilities to do this for a Petri net; however, the most common way is to associate a *firing delay* with each transition. This delay specifies the time that the transition has to be *enabled*, before it can actually fire:



Timed Petri Net Types

SPN (Stochastic Petri Net)

- If the delay is a random distribution function (exponential distribution), the resulting net class is called *stochastic Petri net*.

GSPN (Generalised stochastic Petri Net)

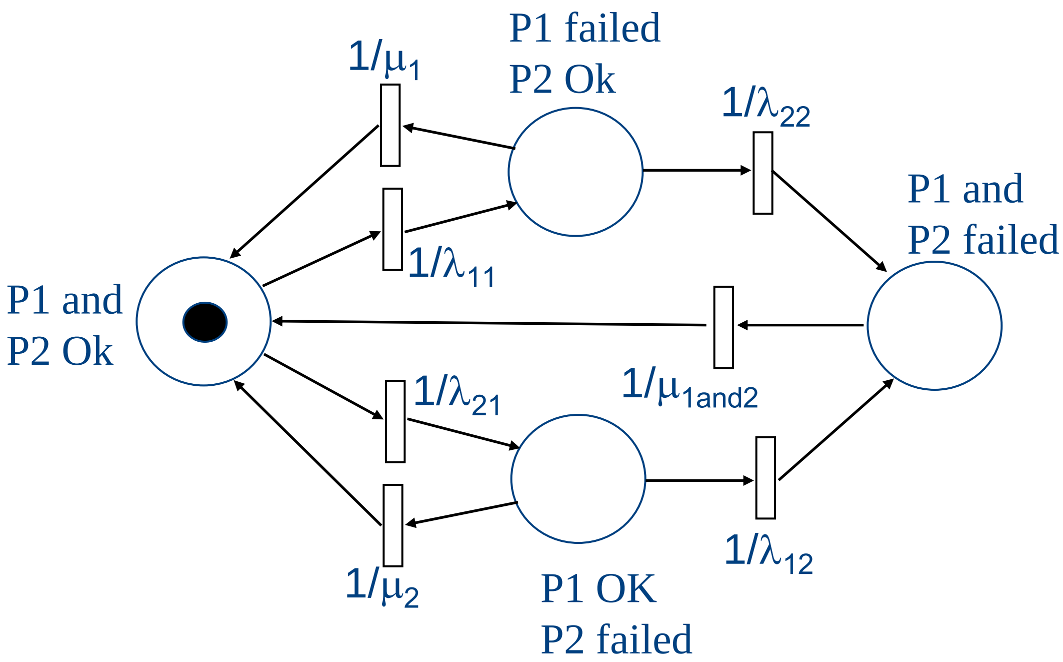
- SPN plus *immediate transitions* (no delay) and inhibit edges.

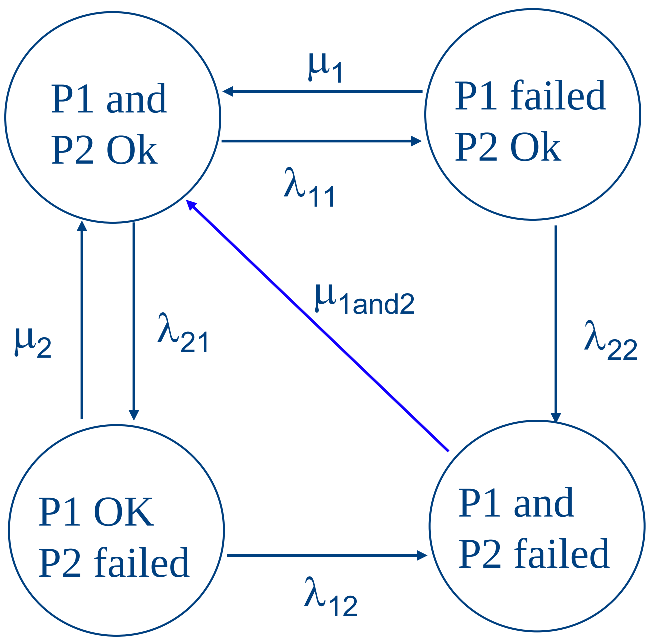
DSPN

- GSPN plus *deterministic transitions* (delay is fixed).

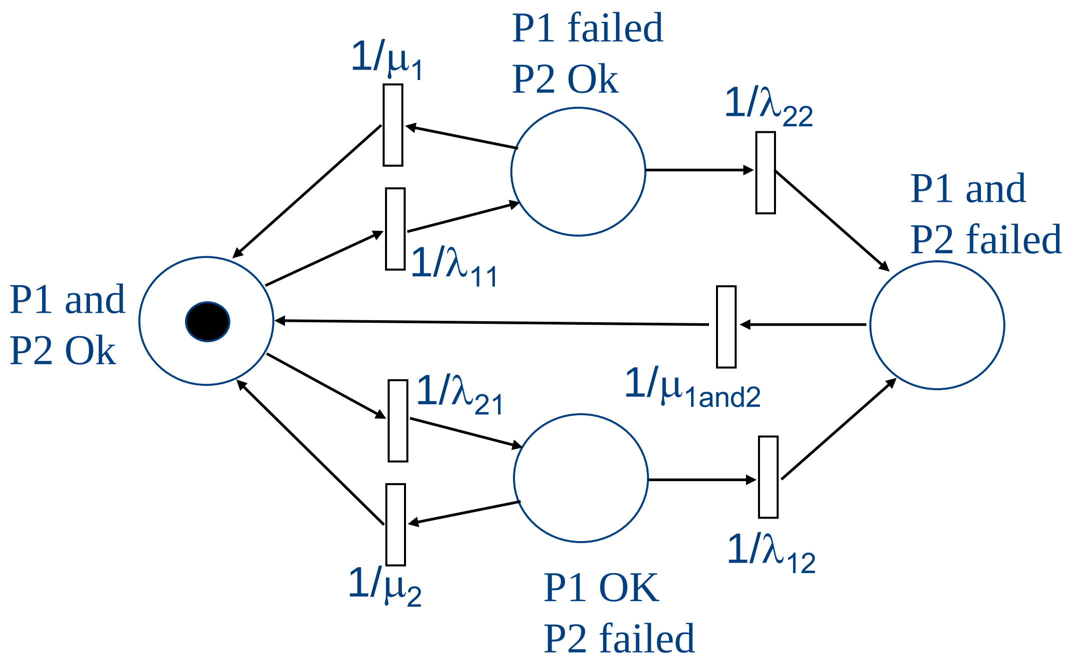
SPN (Stochastic Petri Net)

- Delay is exponentially distributed
- Can be transformed into an equivalent Markov Process

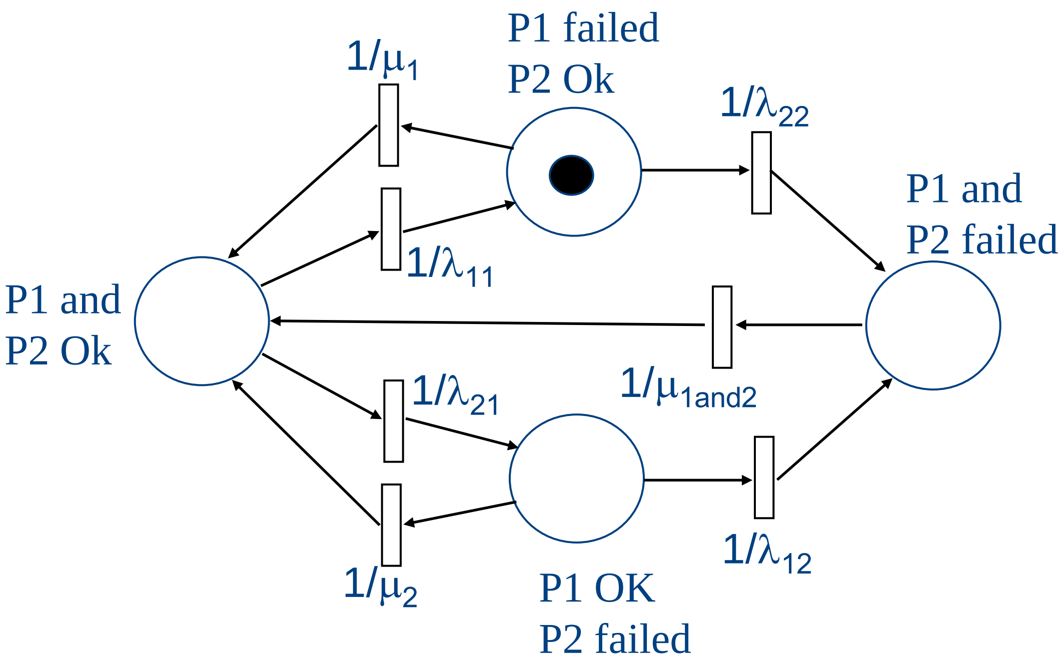




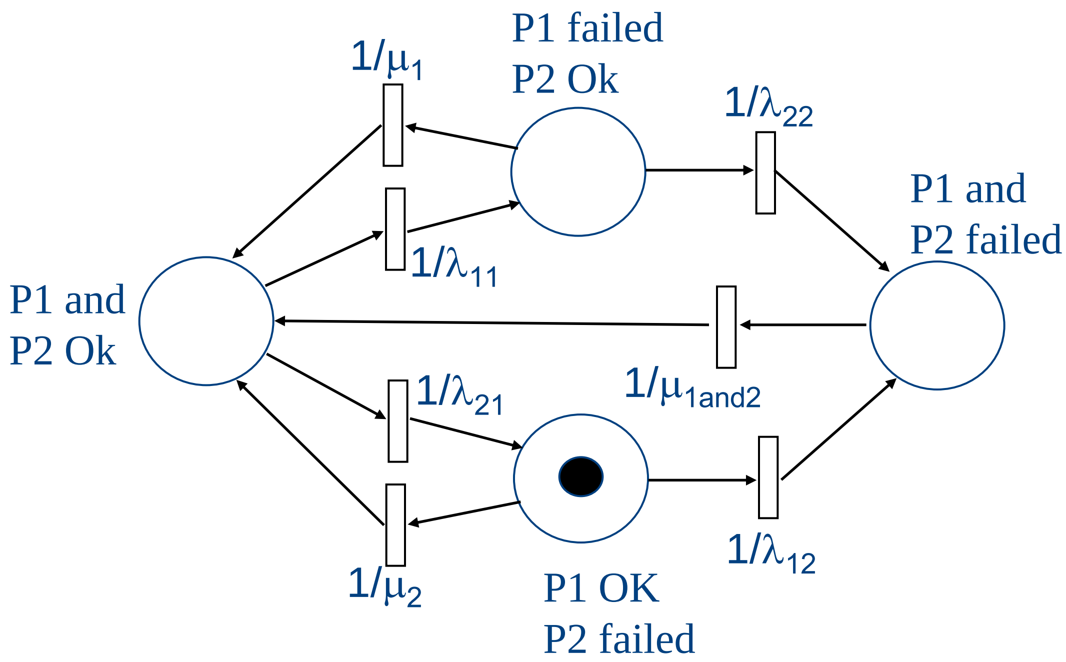
Possible markings: Initial



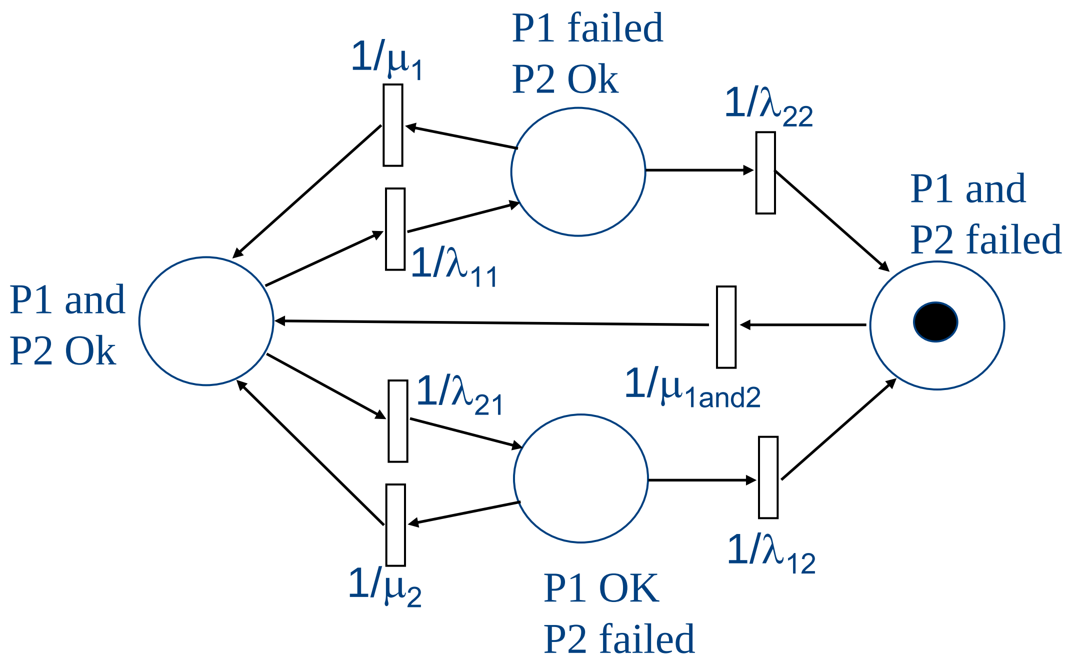
Possible markings: P1 failed



Possible markings: P2 failed

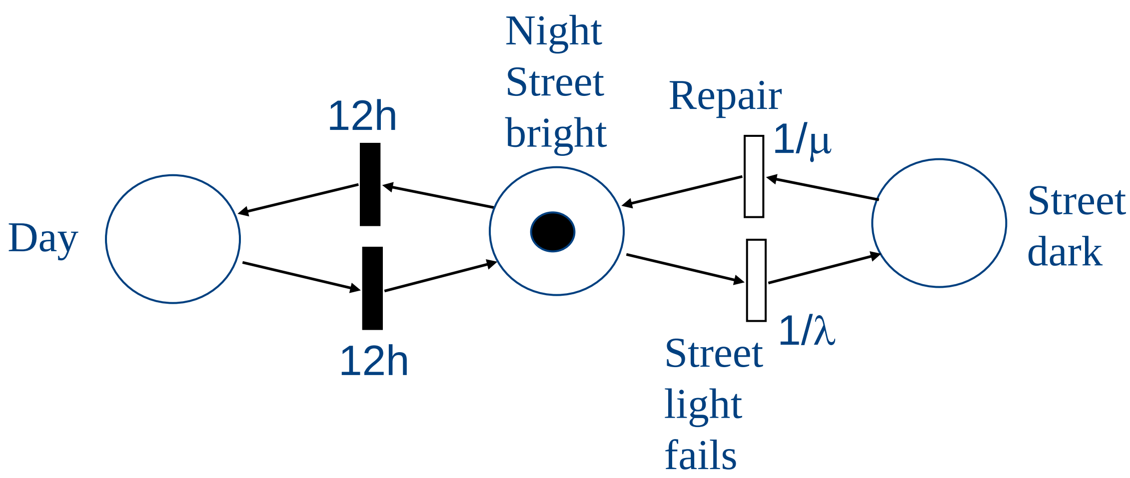


Possible markings: Both failed



DSPN (Deterministic Stochastic Petri Net)

- Exponentially distributed delay + *immediate transitions* (no delay) + *deterministic transitions* (delay is fixed) and inhibit arcs



Literature

- Liggesmeyer 2000, Qualitätssicherung softwareintensiver technischer Systeme, Heidelberg: Spektrum-Verlag 2000
- DIN 25424; DIN 25424-1, Fehlerbaumanalyse Methoden und Bildzeichen, September 1981; DIN 25424-2: Fehlerbaumanalyse Handrechenverfahren zur Auswertung eines Fehlerbaumes, April 1990; Berlin: Beuth Verlag
- DIN 25448, Ausfalleffektanalyse (Fehler-Möglichkeits- und -Einfluß-Analyse), Berlin: Beuth Verlag, Mai 1990
- IEC 812, Analysis Techniques for System Reliability - Procedure for Failure Mode and Effect Analysis (FMEA), International Electrotechnical Commission 1985
- IEC 61025, Fault tree analysis (FTA), International Electrotechnical Commission 1990
- IEC 61078, Analysis techniques for dependability - Reliability block diagram method, International Electrotechnical Commission 1991
- IEC 61165, Application of Markov techniques, International Electrotechnical Commission 1995