



Elements of System Dynamics

Antoine B. Rauzy

PART 4. SYSTEMIC DIGITAL TWINS

LECTURE 10. RELIABILITY ENGINEERING

LECTURE 10. RELIABILITY ENGINEERING

Key notions:

- Risk and Hazard, Risk Matrices, Risk Analysis
- Probabilistic Risk Indicators
- Probabilistic Reliability Models
- Fault Trees, Reliability Block Diagrams, Event Trees
- Markov Chains, Probabilistic State Automata

Learning objectives

The operation of any industrial system induces a **risk** for the system itself, its operators and its environment. This risk is impossible to avoid completely. The only question is therefore to know whether it is socially **acceptable**, i.e. if **accidents** have a sufficiently low **likelihood** to occur and if it is possible to **mitigate**, at least to some extent, their **consequences**.

This is the role of **reliability engineering**.

These analyses are supported by **models**.

Learning objectives of this chapter are:

- To know the key vocabulary and concepts of **reliability engineering**.
- To understand the **mathematical framework** of reliability models.
- To understand the **limits** of reliability engineering.

Agenda

Section 1. Risk Analysis

Section 2. System Reliability Theory

Section 3. Failure Modes and Consequences Analyses

Section 4. Combinatorial Models

Section 5. State Automata

Section 6. Σ^{TM} Patterns

Section 7. Discussion

LECTURE 10. RELIABILITY ENGINEERING

SECTION 1. RISK ANALYSIS

Two Important Remarks

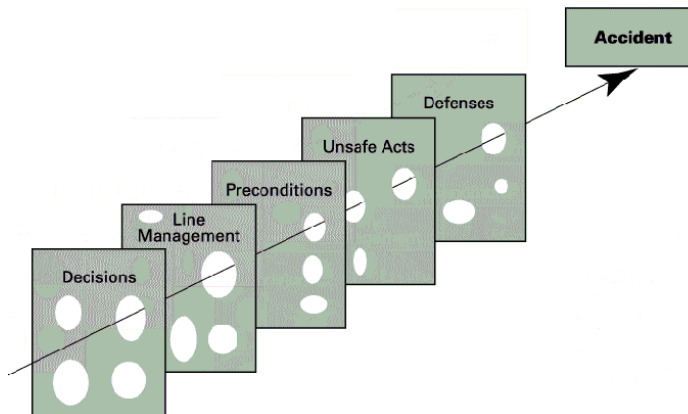
- 1) **Systems architecture** and **risk analyses** are two faces of the same medal.

How to make it work?
How to make it profitable?



What can go wrong?
How likely is something going wrong?
What are the consequences of something going wrong?

- 2) No risk/safety analyses without **accident analyses**.



Incident/accident scenarios are another type of **use cases**.

Hazard versus Risk

Risks must be distinguished from **hazards**:

A **hazard** is an event, or series of events, that occur **outside** the system under study and that impacts the system. Although a hazard may have a strong impact on the system, there is not much the system can do to prevent its occurrence.

Earthquakes are a typical example of hazards.

A **risk** is an event, or series of events, that occurs **inside** the system under study, often because of the occurrence of a hazard.

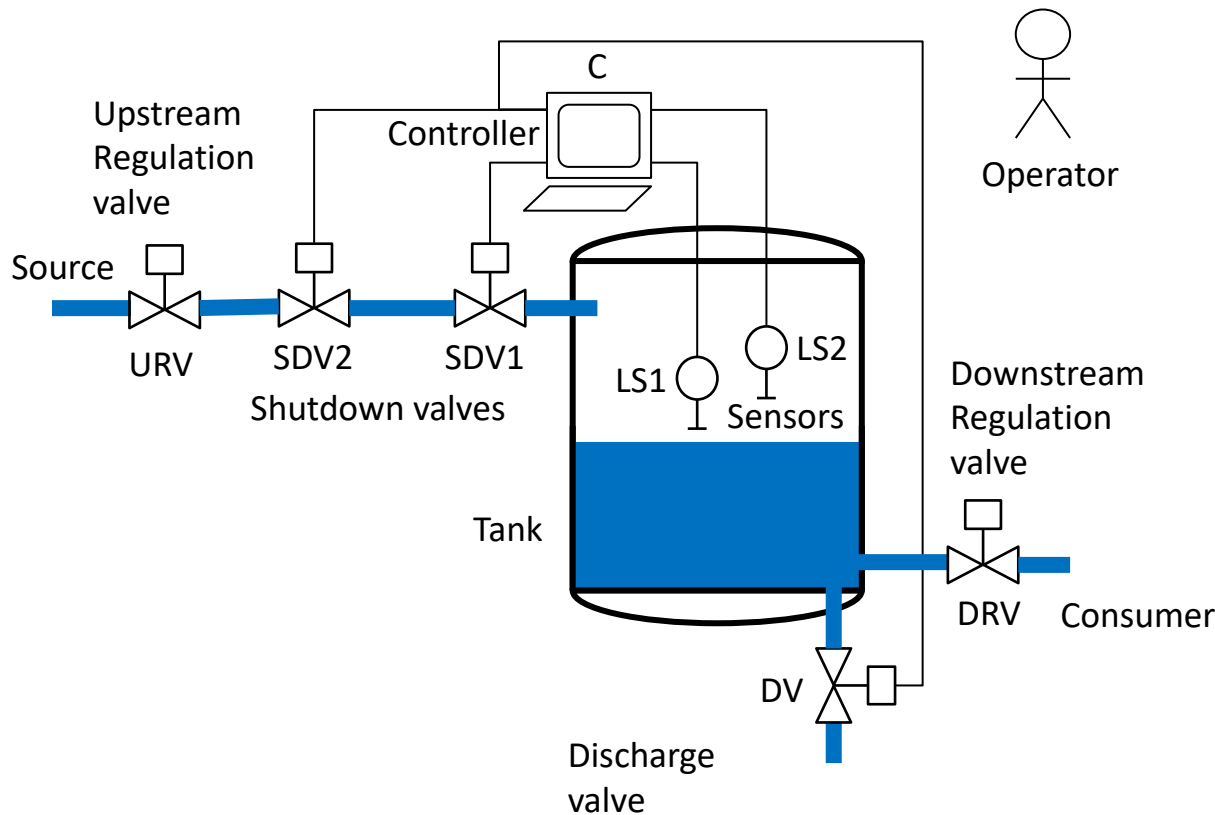
The **crash of a building** after an earthquake is a typical example of risk.

The first task of risk analyses consists in determining hazards, also called initial / external / failure conditions and assessing their potential impact on the system.



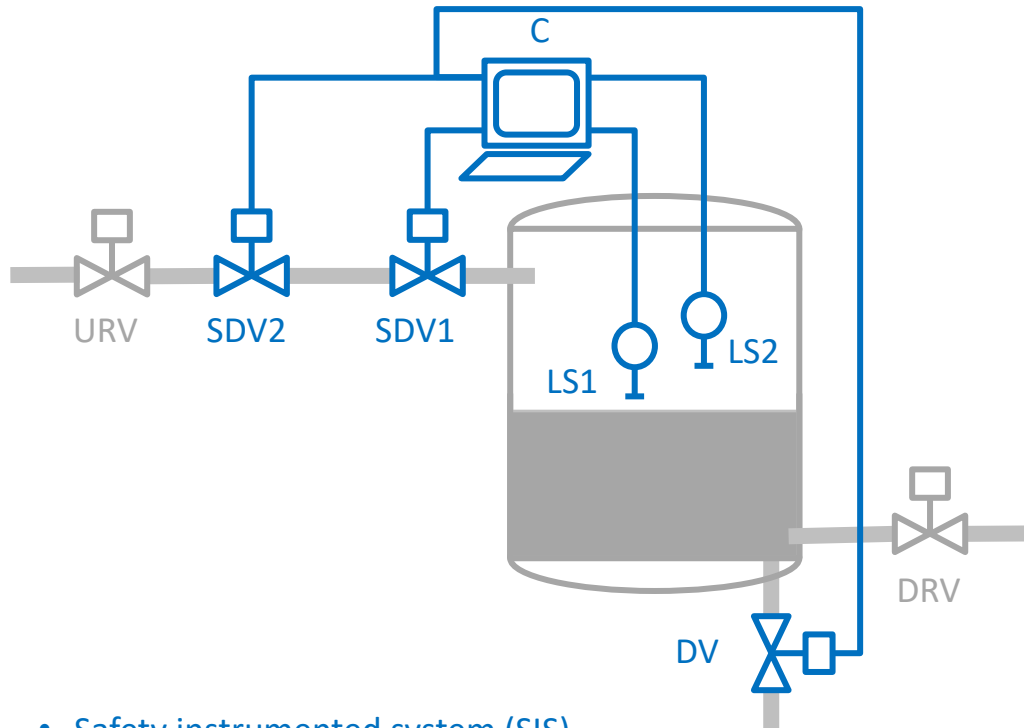
Overflow Protection System

The role of the system is to prevent an overflow of the tank, which may have catastrophic consequences.



We want to know whether this protection system is good enough for the plant to be operated in sustainable economic and safety conditions, and possibly to suggest modifications to its design.

Safety Instrumented Systems



- Safety instrumented system (SIS)
- Equipment under control (EUC)

Safety requirements are often met by means of **safety (instrumented) systems**.

The **safety (instrumented) system** is here a control system made of some sensors, a controller and some actuators (the valves).

Its role is to prevent the occurrence of the **risk** “overflow”, in case of occurrence of the **hazard** “too-high in flow”.

It is a system working **on-demand** (low demand rate) as opposed to safety systems having a **continuous** action on the equipment under control (systems with high demand rate), e.g. the brakes of your car.

Bi-Dimensional Nature of Risks

A risk has a certain **likelihood/probability/frequency of occurrence**, and its occurrence may have more or less **severe consequences**.

This is the reason why **safety standards** such as IEC 61508 provides risk matrices telling whether a risk is acceptable given its frequency and its severity.

IEC 61508 Risk Matrix			Severity			
			<i>Negligible</i>	<i>Marginal</i>	<i>Critical</i>	<i>Catastrophic</i>
			Minor injuries at worst	Major injuries to one or more persons	Loss of a single life	Multiple loss of life
Frequency	<i>Frequent</i>	$> 10^{-3}$	Undesirable	Unacceptable	Unacceptable	Unacceptable
	<i>Probable</i>	10^{-3} to 10^{-4}	Tolerable	Undesirable	Unacceptable	Unacceptable
	<i>Occasional</i>	10^{-4} to 10^{-5}	Tolerable	Tolerable	Undesirable	Unacceptable
	<i>Remote</i>	10^{-5} to 10^{-6}	Acceptable	Tolerable	Tolerable	Undesirable
	<i>Improbable</i>	10^{-6} to 10^{-7}	Acceptable	Acceptable	Tolerable	Tolerable
	<i>Incredible</i>	$\leq 10^{-7}$	Acceptable	Acceptable	Acceptable	Acceptable

Risk Reduction

To **reduce** a risk, we can either reduce its **frequency/probability/likelihood**, or its **severity**, or both.

IEC 61508 Risk Matrix			Severity			
			Negligible	Marginal	Critical	Catastrophic
			Minor injuries at worst	Major injuries to one or more persons	Loss of a single life	Multiple loss of life
Frequency	Frequent	$> 10^{-3}$	Undesirable	Unacceptable	Unacceptable	Unacceptable
	Probable	10^{-3} to 10^{-4}	Tolerable	Undesirable	Unacceptable	Unacceptable
	Occasional	10^{-4} to 10^{-5}	Tolerable	Tolerable	Undesirable	Unacceptable
	Remote	10^{-5} to 10^{-6}	Acceptable	Tolerable	Tolerable	Undesirable
	Improbable	10^{-6} to 10^{-7}	Acceptable	Acceptable	Tolerable	Tolerable
	Incredible	$\leq 10^{-7}$	Acceptable	Acceptable	Acceptable	Acceptable

Risk reduction is in general obtained by means of **safety mechanisms** and requires **analysis and development efforts**.

Reduction of frequency

Electronic Stability Program (ESP®)



Risk



Reduction of severity

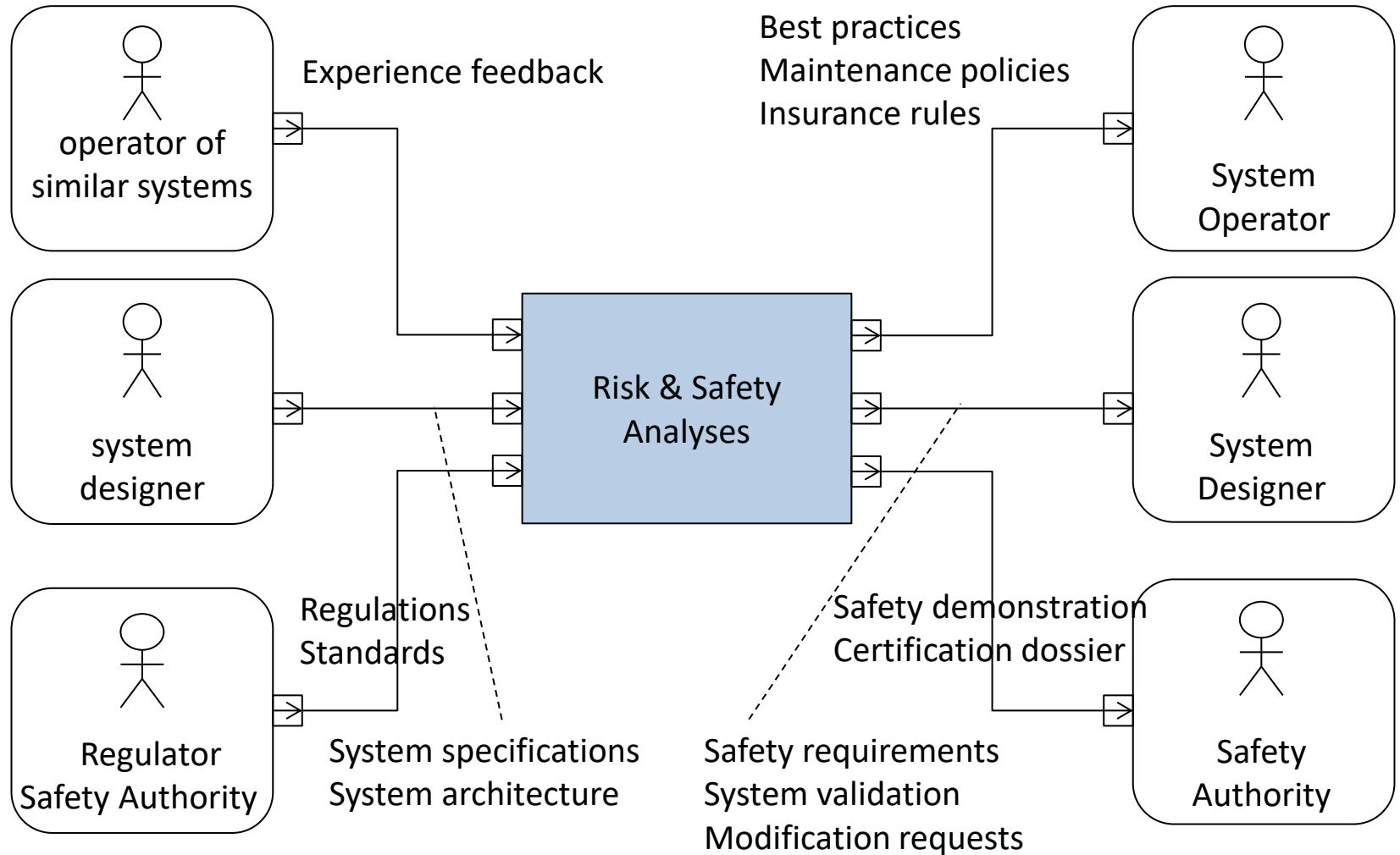


Risk Reduction Effort in Airborne Systems

In avionic, the **risk reduction effort** is measured with **Design Assurance Level** (DAL, ARP4751 & ARP4754)

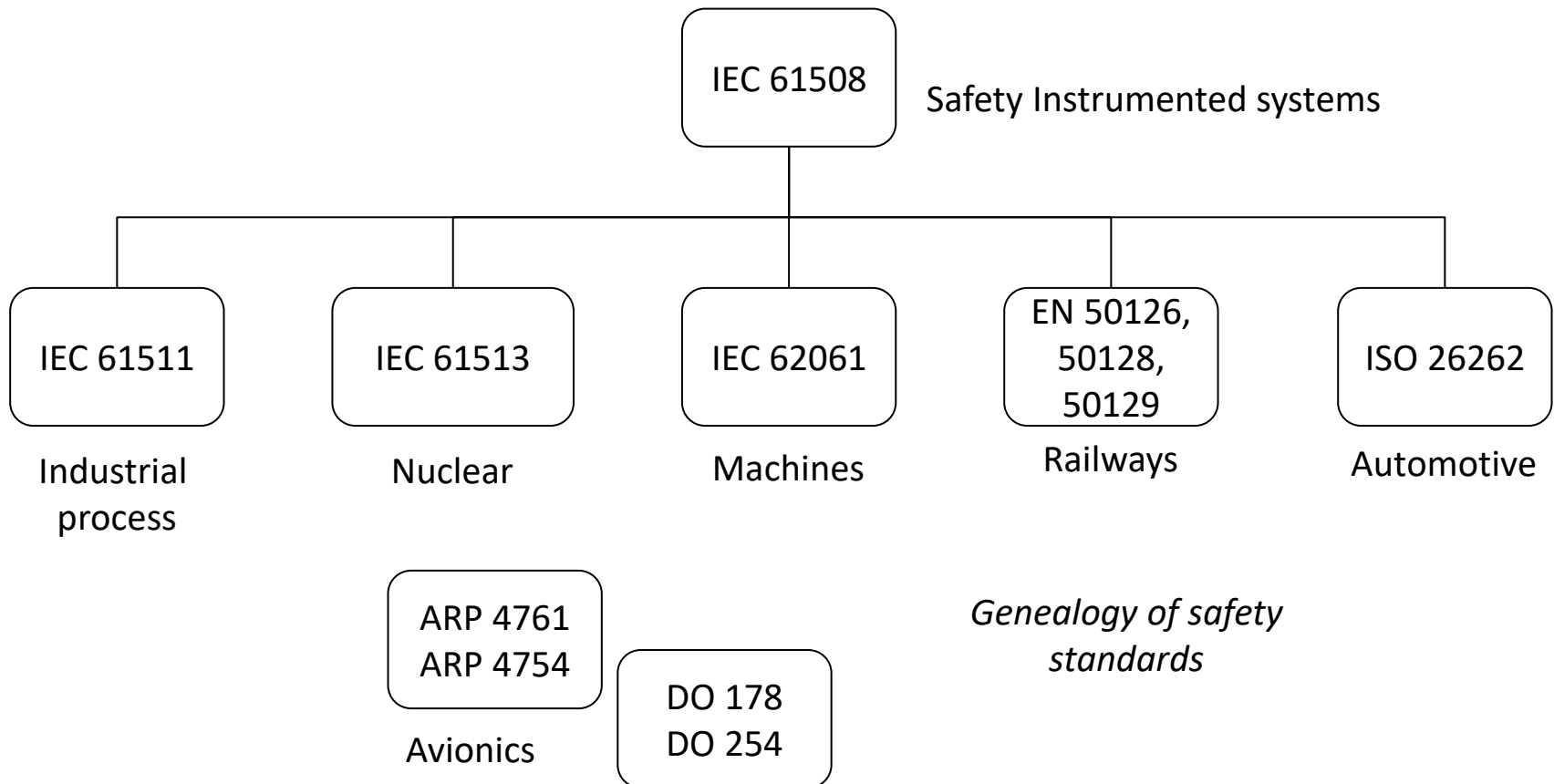
A risk is acceptable when its frequency per flight hour is less than that defined by the DAL level associated with the severity of the risk.			Severity			
			Minor	Major	Hazardous	Catastrophic
			- Slight reduction of safety margins - Slight increase in crew workload - Discomfort for passengers	- Significant reduction in safety margins and functionality - Significant increase in crew workload - Uncomfortable situation for passengers	- Strong reduction in safety margins and functionality - Strong increase in workload and crew stress - Negative impacts on passengers	Any situation preventing the continuation of the flight or landing in safe conditions
Frequency per flight hour	Probable	$10^{-5} < F \leq 10^{-1}$	DAL Level D	Inacceptable	Inacceptable	Inacceptable
	Unusual	$10^{-7} < F \leq 10^{-5}$	Acceptable	DAL Level C	Inacceptable	Inacceptable
	Improbable	$10^{-9} < F \leq 10^{-7}$	Negligible	Acceptable	DAL Level B	Inacceptable
	Extremely improbable	$F \leq 10^{-9}$	Negligible	Negligible	Acceptable	DAL Level A

Risk & Safety Analyses



Safety Standards

Safety analyses are strongly guided by **safety standards** and **best practices guides**. These define more requirements on the **process** to be followed (e.g. ensuring the traceability of design choices) than on the actual content of safety analyses.



Safety Integrity Levels

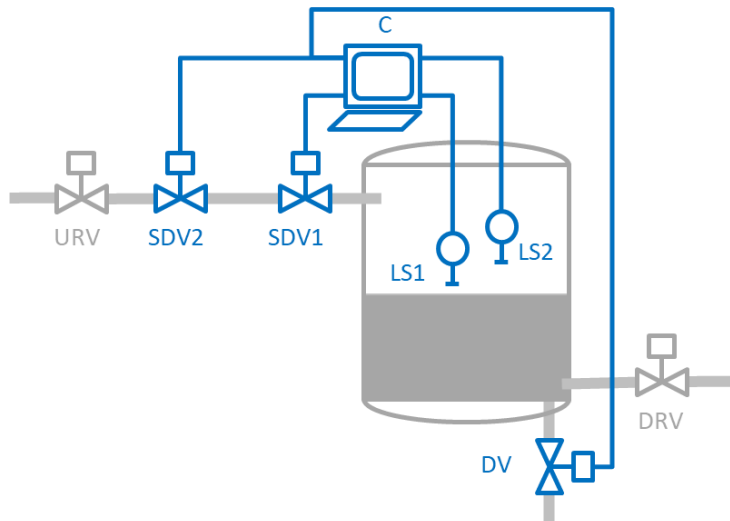
The **Safety Integrity Level (SIL)** of the **Safety Instrumented System (SIS)** is determined by the **Risk Reduction Factor (RRF)** provided by the SIS to the **Equipment Under Control (EUC)**. Assuming that the SIS prevents the whole risk, this is also a measure of the likelihood of a **Failure of the SIS**.



	Low Demand Mode: Probability of Failure on Demand ($PFD_{average} \approx F(t)$)	High Demand Mode: Probability of Failure per Hour ($PFH \approx Q(t)$)
SIL 4	10^{-5} to 10^{-4} (RFF > 10000)	10^{-9} to 10^{-8}
SIL 3	10^{-4} to 10^{-3} ($1000 \leq RFF < 10000$)	10^{-8} to 10^{-7}
SIL 2	10^{-3} to 10^{-2} ($100 \leq RFF < 1000$)	10^{-7} to 10^{-6}
SIL 1	10^{-2} to 10^{-1} ($10 \leq RFF < 100$)	10^{-6} to 10^{-5}

Risk Analysis

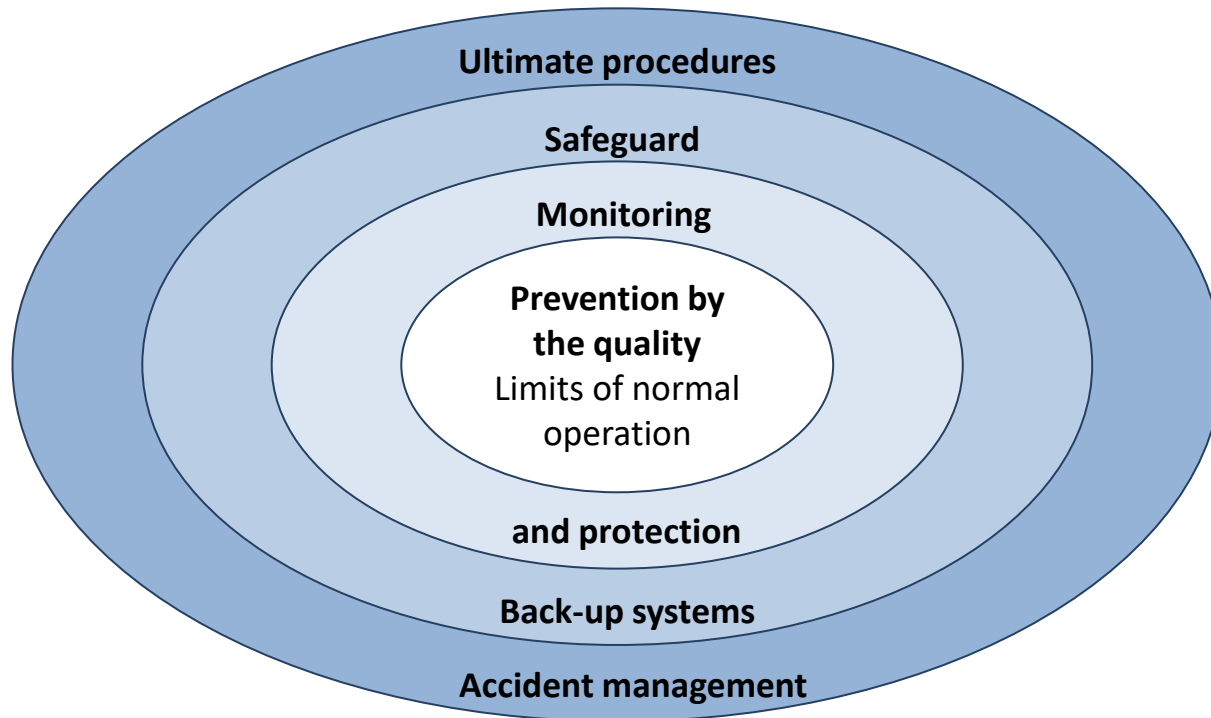
At plant level, the main risk stands in an **overflow of the tank** while the plant is in operation. This overflow may be the consequence of the hazard consisting in a too strong in-flow. This risk is **critical** and must be mitigated. Moreover, the plant operator must be able to **prove** that he makes **sufficient efforts** to get this risk low enough.



The plant operator must therefore demonstrate that the countermeasures against an overflow are effective enough.

The role of the **probabilistic risk/safety analysis** is therefore to assess the **on-demand availability of the safety instrumented system!**

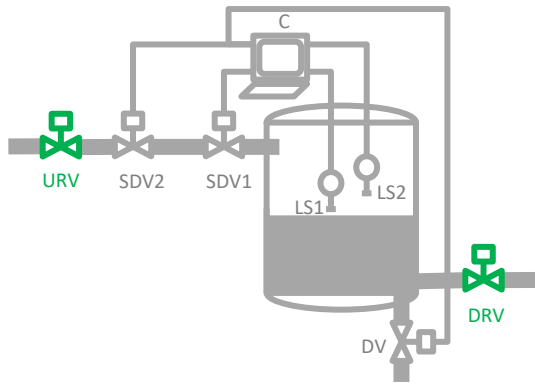
Defense in Depth



Defense in depth is a safety doctrine which consists in structuring the architecture of a system by designing **several successive and independent safety barriers** in the system so to prevent a risk to occur or to mitigate its consequences.

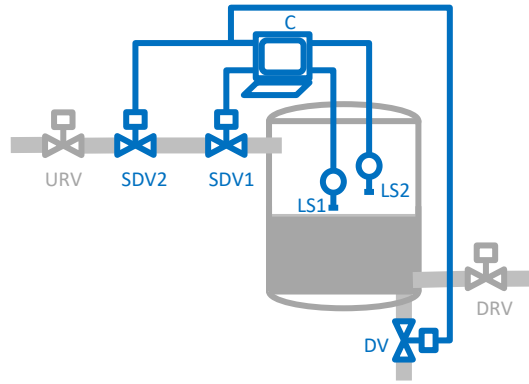
Defense in Depth

Regulation System



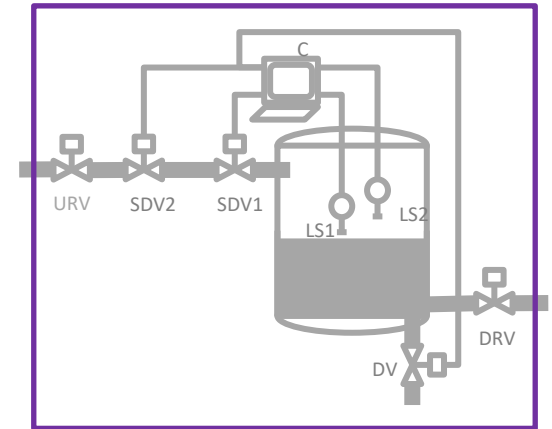
Prevention by the quality
Monitoring and Management

Safety Instrumented System



Safeguard
Back-up systems

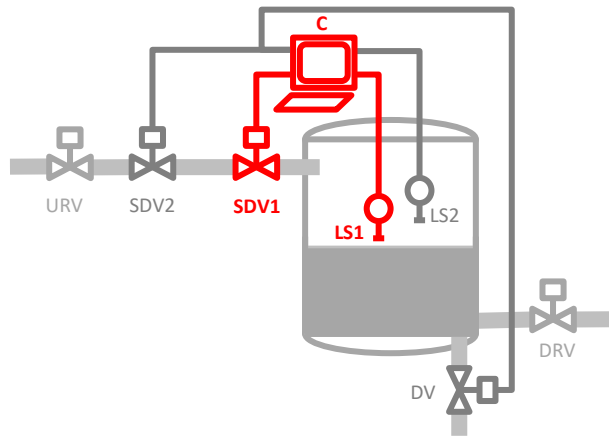
Containment



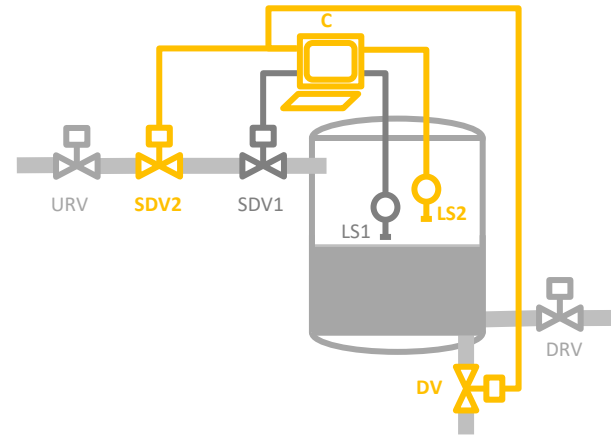
Accident Management
Mitigation

Safety Barriers

The system acts as successive **safety barriers** for the equipment under control.



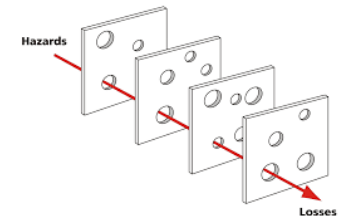
Primary (active) barrier prevention



Back-up (active) barrier prevention

The failure of the controller causes the **loss** of the first and the second barriers. They are not **independent**.

The failure of the controller is a **common cause failure** for the two barriers (there may have some other, e.g. a shock on the two shutdown valves).



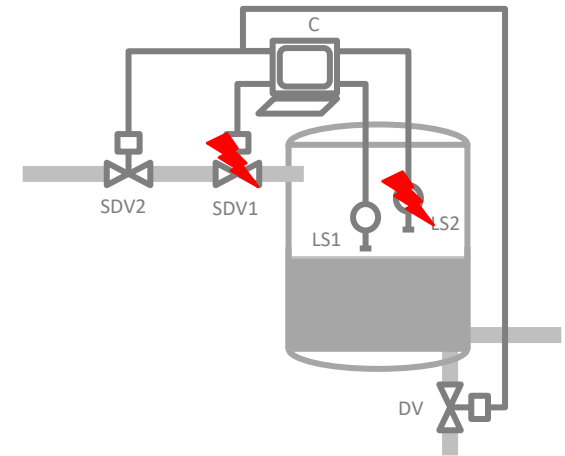
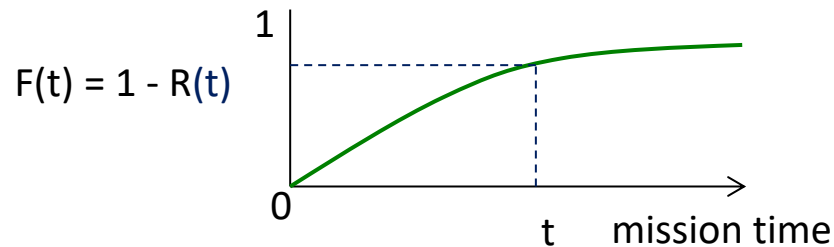
CHAPTER 10. RELIABILITY ENGINEERING

SECTION 2. SYSTEM RELIABILITY THEORY

Reliability & Availability

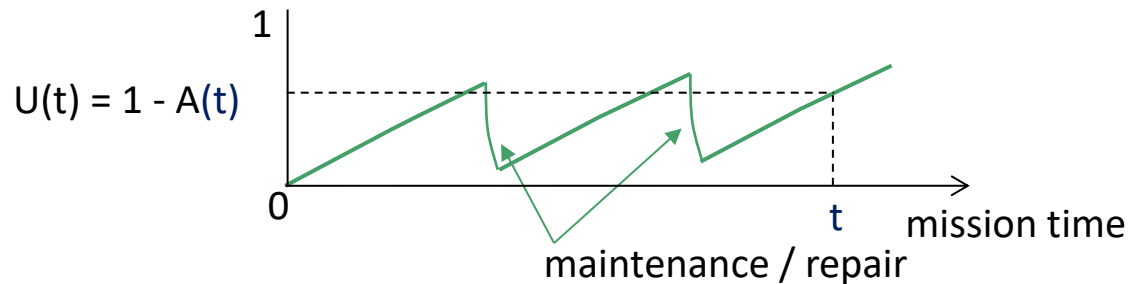
Reliability

- $R(t)$ = Probability that the system is operating without interruption from time 0 to time t .



Availability

- $A(t)$ = Probability that the system is operating at time t .



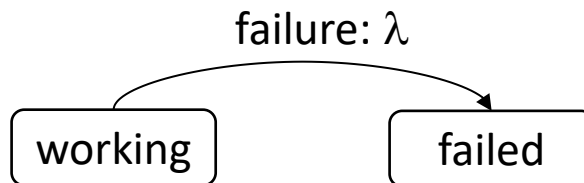
We may lose the safety system but repair it before the occurrence of the hazard

Failure Rate & Mean-Time to Failure

- The **failure rate** is the frequency with which an engineered system or component fails, expressed in failures per unit of time. It is often denoted by the Greek letter λ .

$$\lambda(t) = \lim_{dt \rightarrow 0} \frac{R(t) - R(t+dt)}{dt \cdot R(t)}$$

- The failure rate is often assumed constant over the time, obeying in this way Markovian hypothesis (memoryless process)



$$F(t) = 1 - R(t) = 1 - e^{-\lambda t}$$

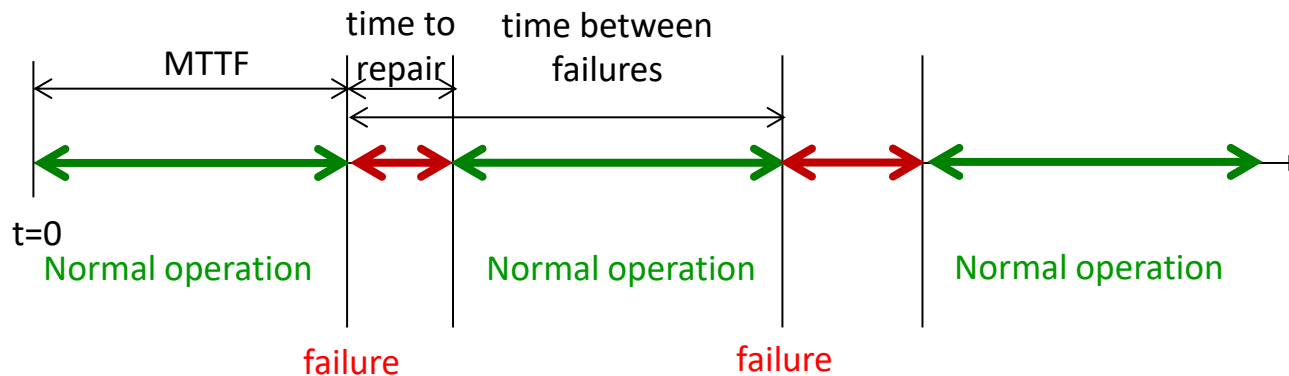
- In this case, the **mean time to failure (MTTF)** is: $MTTF = \frac{1}{\lambda}$

Considering Repairs

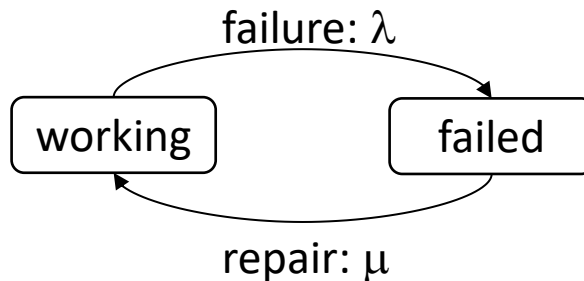
Mean Time To Failure (MTTF)

Mean Time Between Failure (MTBF)

Mean Time To Repair (MTTR)



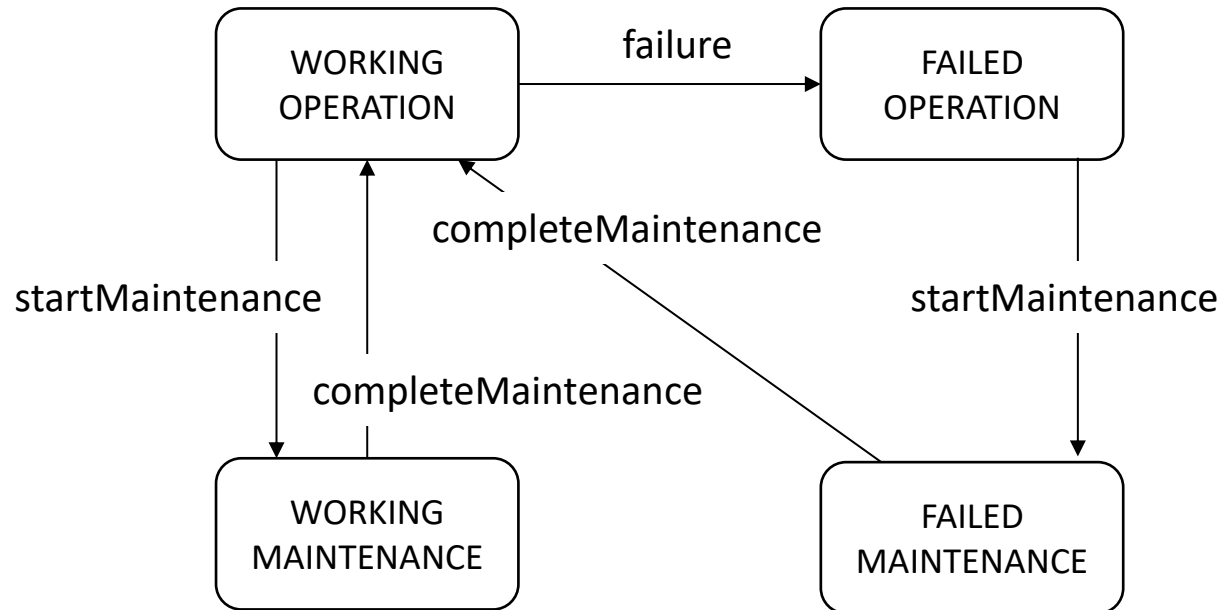
Repair rate (μ)



Under Markovian hypotheses:

- $MTBF = \frac{1}{\lambda}$
- $MTTR = \frac{1}{\mu}$

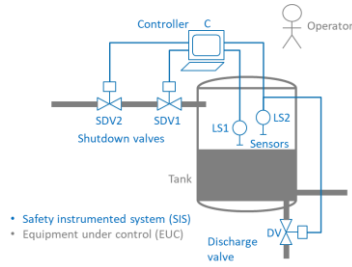
Considering Maintenance Operations



- The events startMaintenance and failure are in competition.
- When the system is failed, the event startMaintenance may represent either the beginning of a corrective maintenance operation (immediate) or a preventive maintenance operation (delayed).

Probabilistic Risk Assessment

System specifications



Reliability data for components



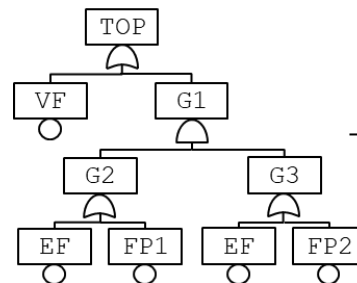
Safety requirements



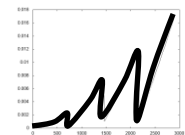
- The bad news: assessing the severity of a risk is industry dependent.
- The good news: models to assess the probability of a risk are very similar across the different industries.

Design

Model



Assessment



- Probabilistic risk indicators
- Scenarios of failure
- Ranking of components w.r.t. to their contribution to the risk
- ...

Classes of Modeling Languages

Combinatorial Formalisms

- Fault Trees
- Event Trees
- Reliability Block Diagrams
- Finite Degradation Structures

States Automata

- Markov chains
- Dynamic Fault Trees
- Stochastic Petri Nets
- AltaRica
- Σ^{TM}
- ...

Process Algebras

- Agent-based models
- Process algebras
- Python/Java/C++
- ...

Expressive power

States

States + transitions

Deformable systems

Complexity of assessments

#P-hard but reasonable
polynomial approximation

PSPACE-hard

Undecidable

Difficulty to design, to validate and to maintain models

CHAPTER 10. RELIABILITY ENGINEERING

SECTION 3. FAILURE MODES AND EFFECTS ANALYSIS

Failure Modes, Effects and Criticality Analysis

The **Failure Modes, Effects and Criticality Analysis (FMECA)** is an inductive safety analysis (i.e. it is based on the existing system), aiming at identifying a system or sub-systems level, the potential failure modes of its elements, their causes and their effects.

Function or component	Failure mode	Potential effect of the failure	C (severity level)	Possible cause of the failure	F (frequency)	D (detection level)	Implemented control	RPN (Risk Priority Number)	Recommended action	Responsible	Actions to be done & implementation date
Container	Water overflow	Water flowing on the ground	8	Out of order or faulty maximum water level sensor	2	5	Filling test	80	Analyze the cost of adding a medium water level sensor (between min & max sensors)	Jane Doe	Replacement of the maximum water level sensor (12 June 2012)

Creating an FMECA aims at understanding how to cover the risks (see above) associated with each functional or physical component of a system and each of its potential failure modes

Failure Modes

Component	Icon	Failure modes	Probability distribution
Controller		•Shutdown •Spurious trip	•Exponential •Failure rate: $1.0 \cdot 10^{-7} \text{ h}^{-1}$
Shutdown valve		•Fail to open •Fail to close •Stuck halfway	•Exponential •Failure rate: $1.0 \cdot 10^{-4} \text{ h}^{-1}$
Discharge valve		•Fail to open •Fail to close •Stuck halfway	•Exponential •Failure rate: $1.0 \cdot 10^{-5} \text{ h}^{-1}$
Level sensor		•No signal •Erroneous signal	•Weibull •Scale parameter: $1.0 \cdot 10^3 \text{ h}$ •Shape parameter: 2

CHAPTER 10. RELIABILITY ENGINEERING

SECTION 4. COMBINATORIAL MODELS

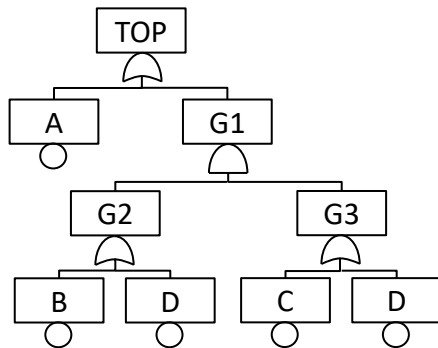
What to Do?

We shall design a probabilistic risk assessment model to:

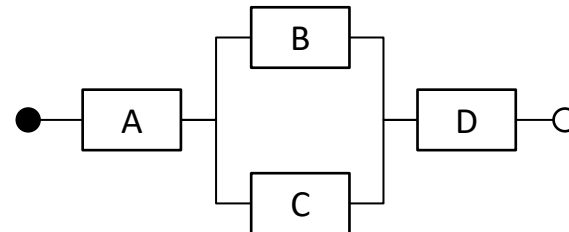
- Assess the **on-demand availability** of the safety instrumented system and possibly require some modifications in the system design and/or operation rules.
- Prove that the safety instrumented system is available enough to warranty the safe operation of the plant.

In this chapter we shall consider two modeling approaches, relying on two different combinatorial (Boolean) modeling formalisms.

Approach 1: **fault trees**

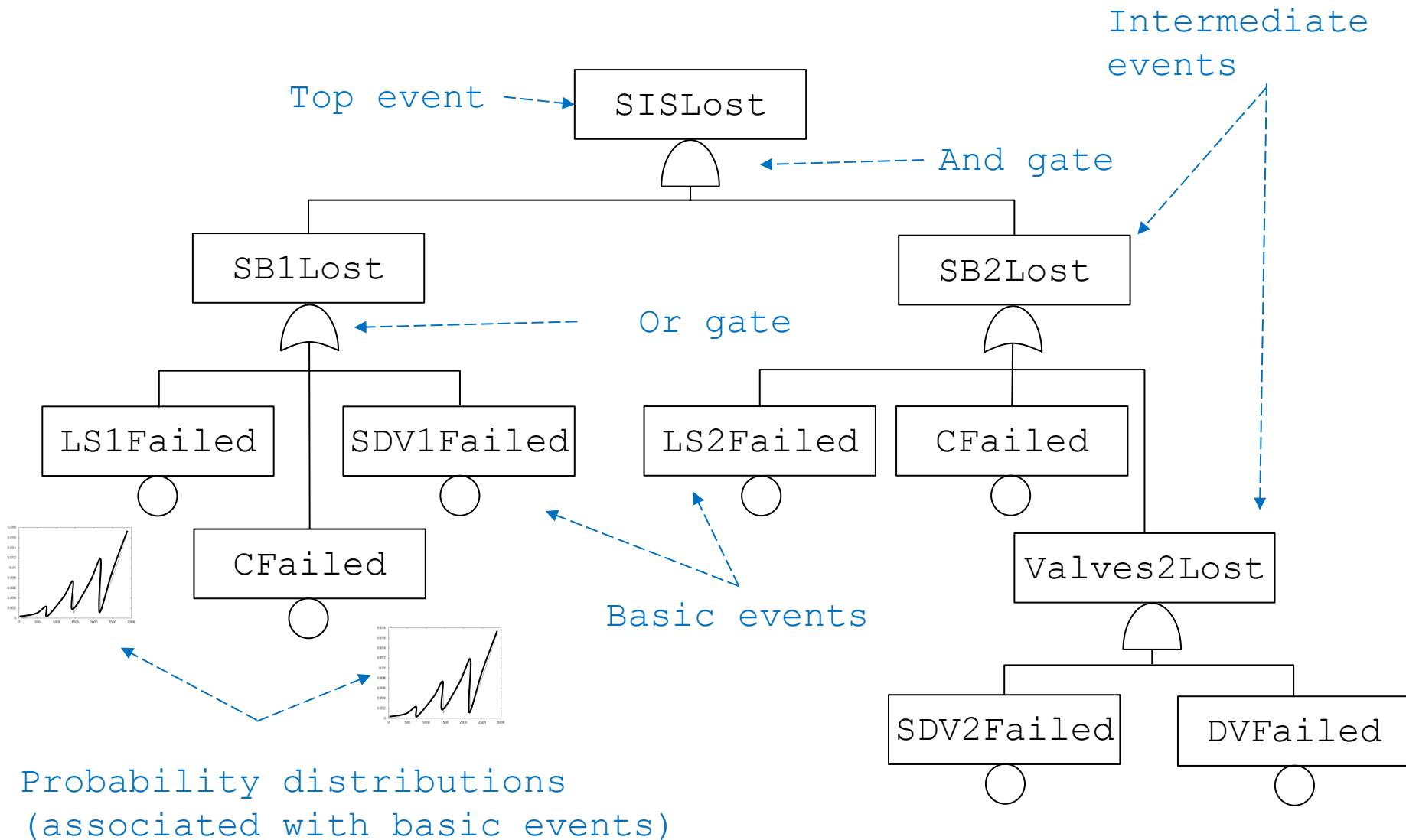


Approach 2: **reliability block diagrams**



SECTION 4.1. FAULT TREES

Fault Trees

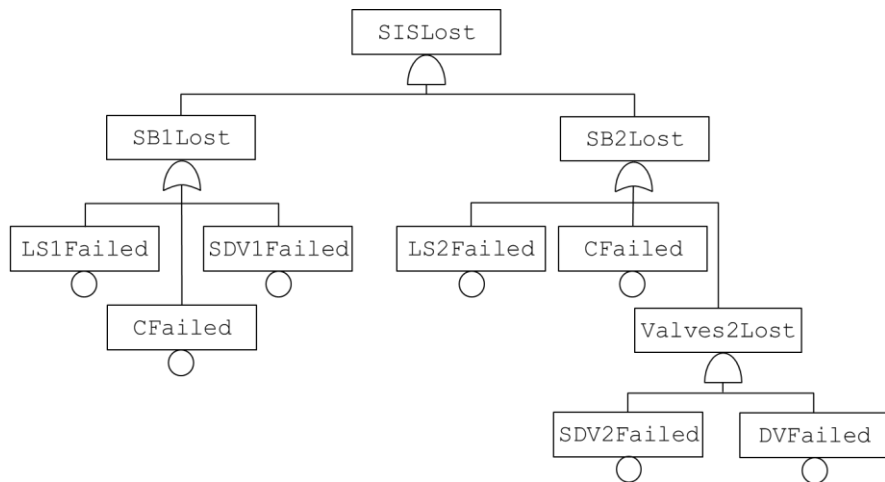


Important Remark

Top-, intermediate- and basic events are **events** in the **sense** of **probability theory**, i.e. **out-comes of an experience**.

They are not events in the common English sense, i.e. something that happens.

They describe **states of the system**, not transitions between these states.



Interpretation: the SIS is lost if the event **SISLost** is realized.

SISLost is realized if both **SB1Lost** and **SB2Lost** are realized.

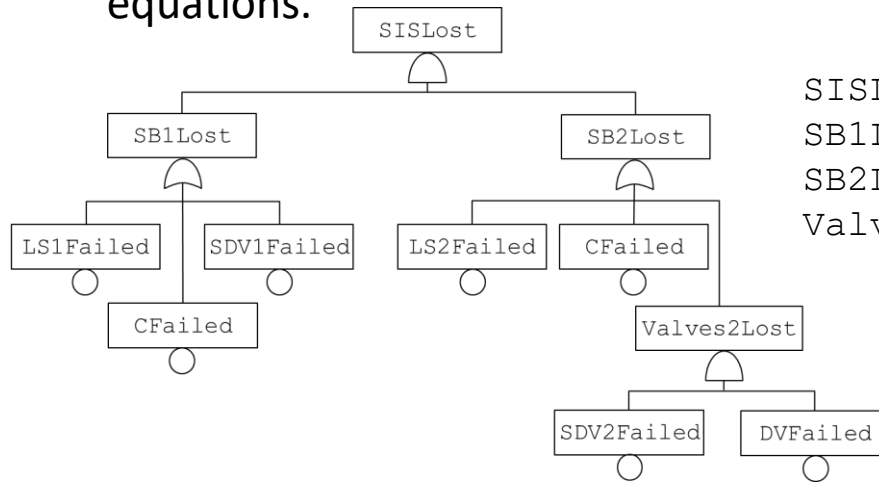
SB1Lost, i.e. the first safety barrier is lost is realized if either **LS1failed** is realized, or **Cfailed** is realized or **SDV1** is realized.

...

Do not confuse fault (the state of a failed component) and failure (the event by which the component fails).

Systems of Boolean Equations

Any fault tree is the **graphical representation** of a uniquely rooted, data flow Boolean circuit. In other words, the real model is the system of Boolean equations.



$SISLost = SB1Lost \text{ and } SB2Lost$

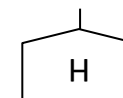
$SB1Lost = LS1Failed \text{ or } Cfailed \text{ or } SDV1Failed$

$SB2Lost = LS2Failed \text{ or } Cfailed \text{ or } Valves2Lost$

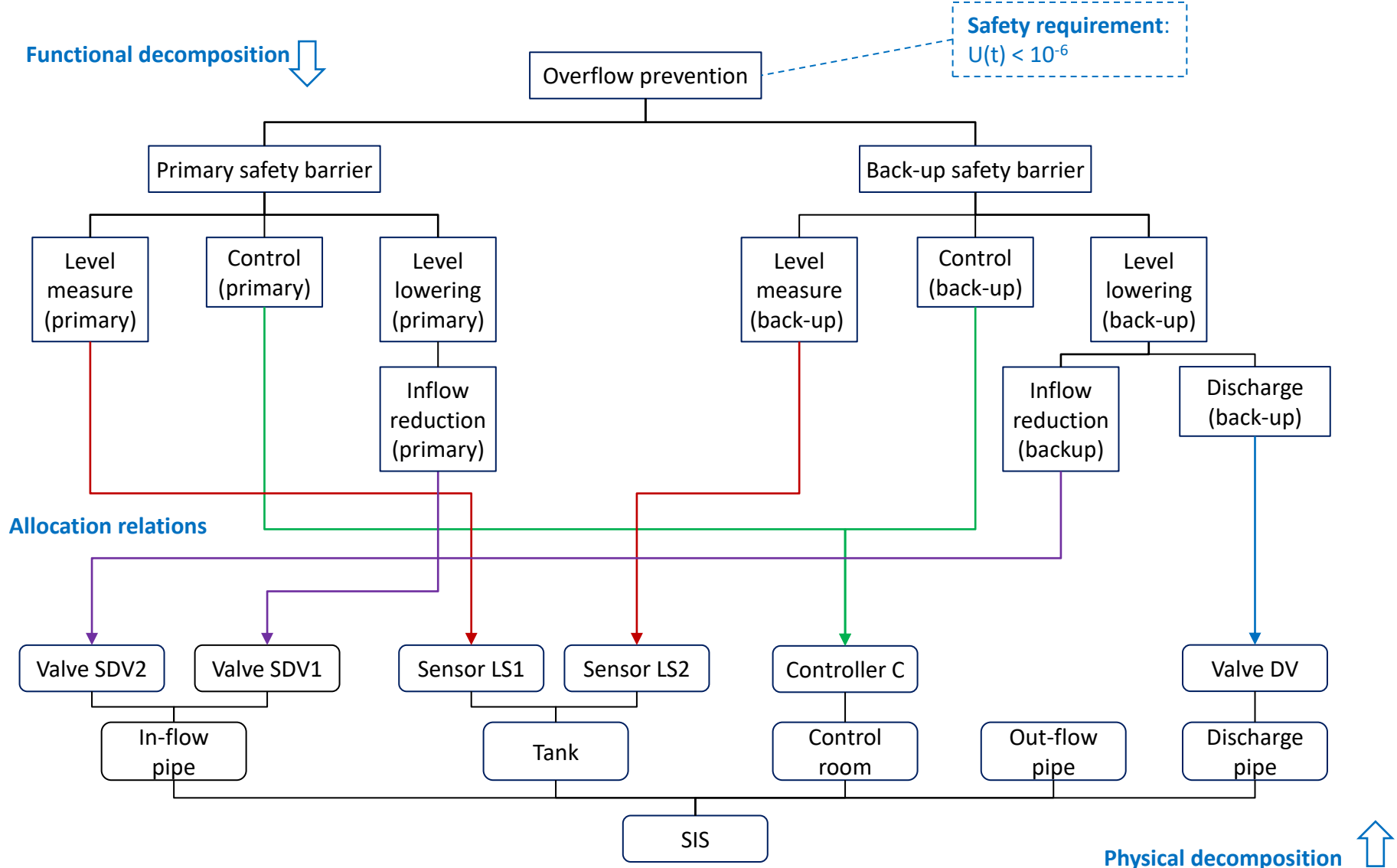
$Valves2Lost = SDV2Failed \text{ and } DVFailed$

Fault trees	Boolean equations
Basic events	State variables
Intermediate events	Flow variables
Top event	Root variable
House events	Source variables

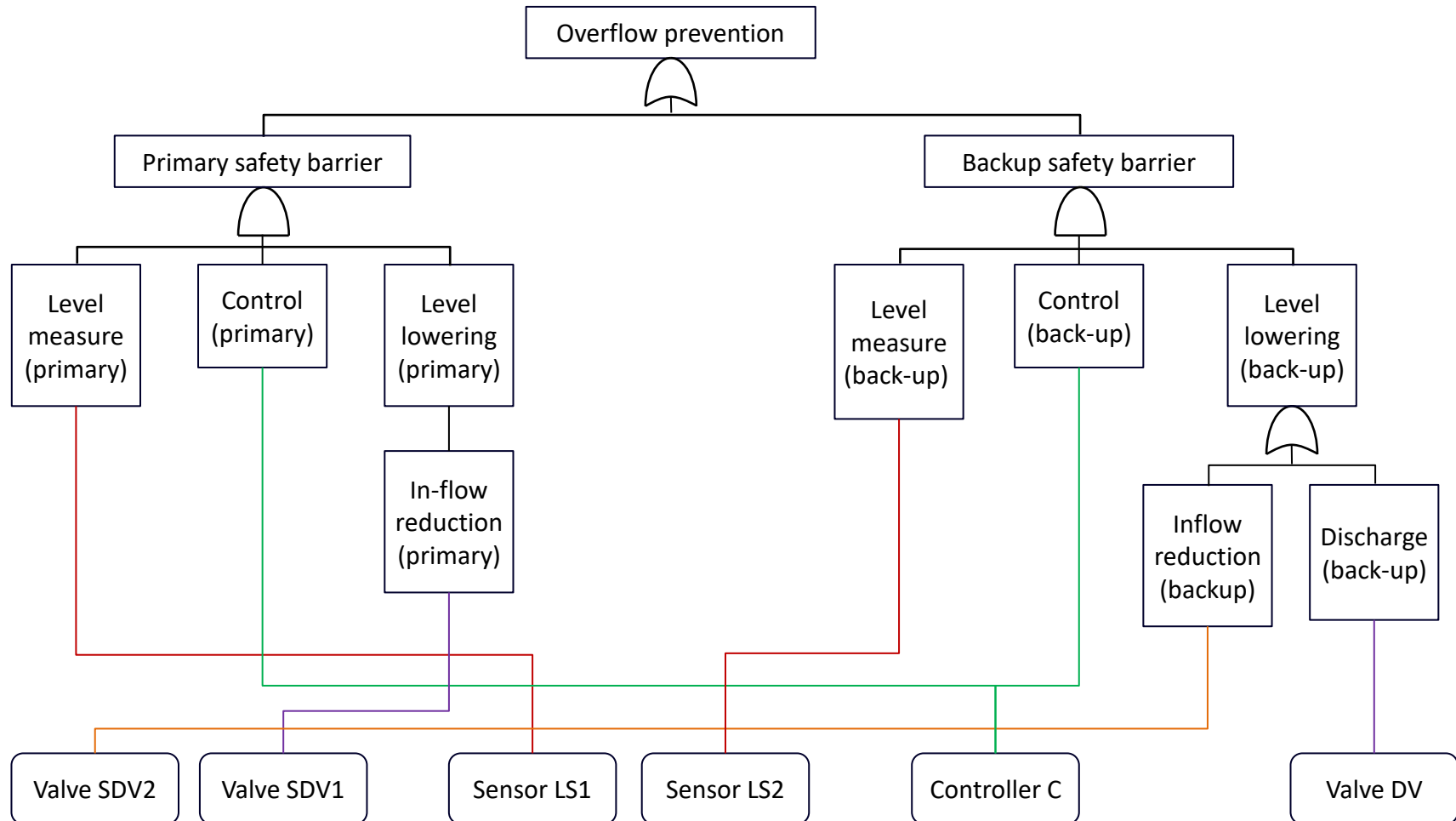
House events are configuration means. They can be seen as flow variables that can take only the value false and true.



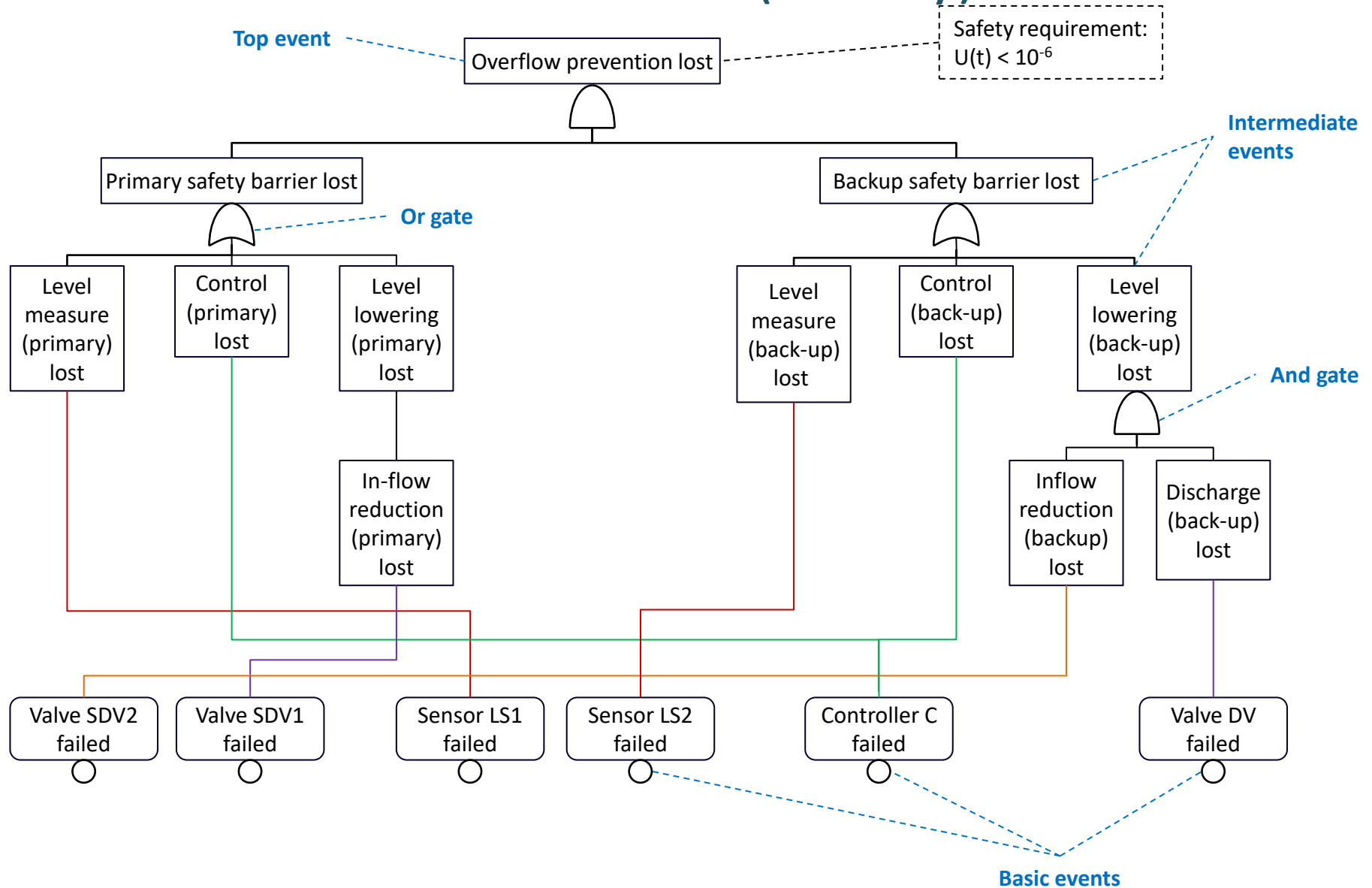
Architecture of the System



Logical Architecture of the System

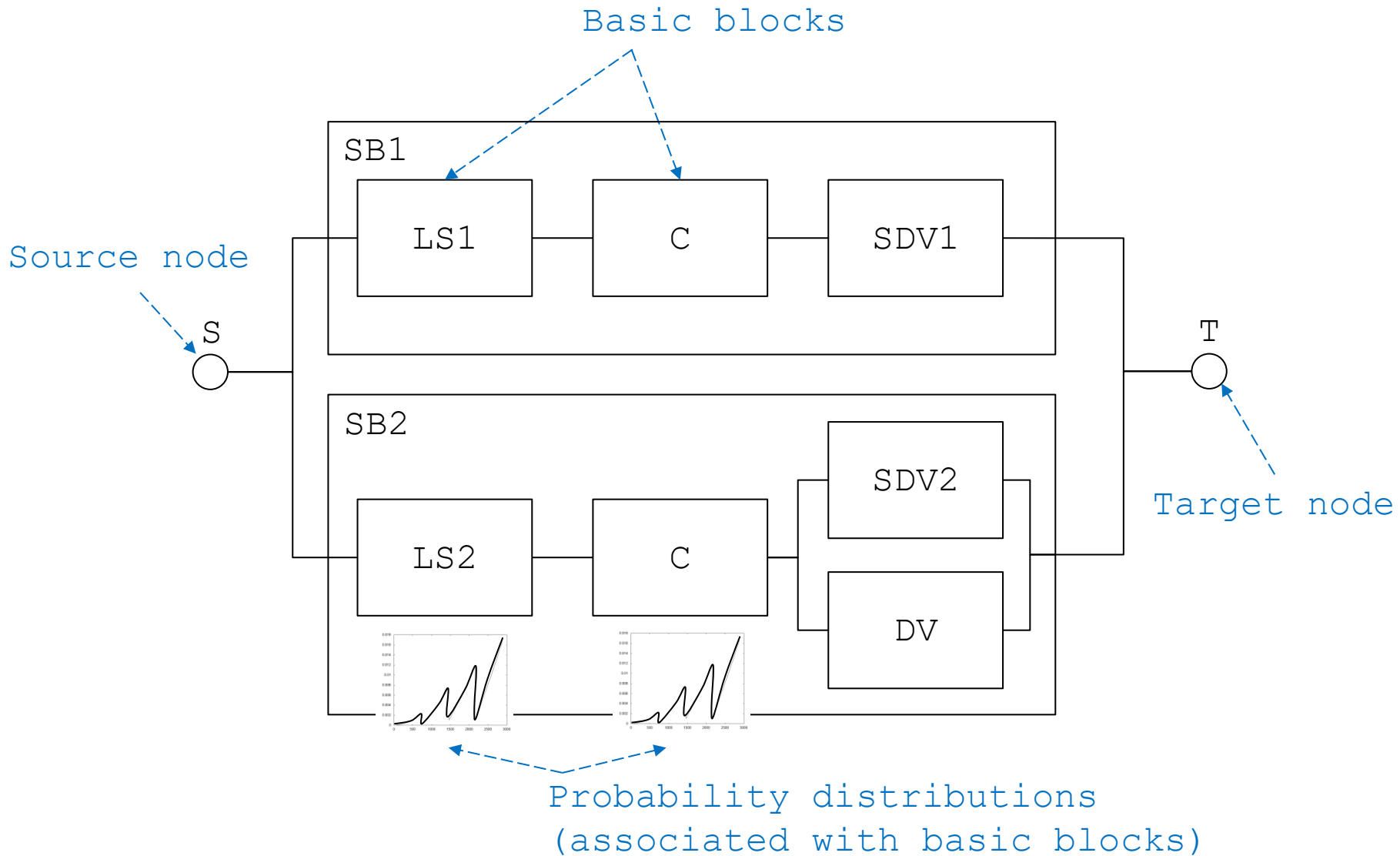


Fault tree (Duality)



SECTION 4.2. RELIABILITY BLOCK DIAGRAMS

Reliability Block Diagrams



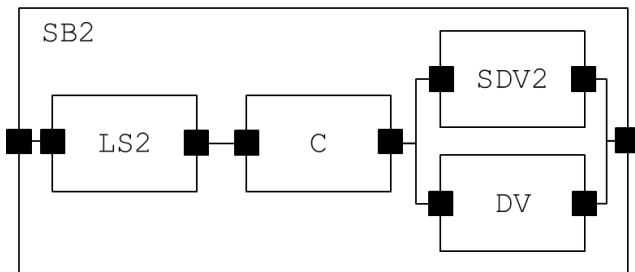
Important Remarks

As fault trees, reliability block diagrams must be interpreted in terms of states (of components and systems):



There is a flow going out the output port of LS2 if there is a flow coming in the input port of LS2 and LS2 is working.

Dually: there is no flow going out the output port of LS2, if either there is no flow coming in the input port of LS2 or LS2 is failed.

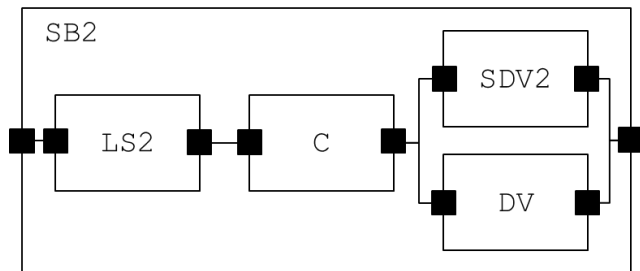


There is a flow coming in the output of SB2 if either there is a flow going out the output port of SDV2 or an output flow going out the output port of DV (or both).

Dually: there is no flow coming in the output port of SB2 if there is no output flow going out of the output port of SDV2 and there is no output flow going out the output port de DV.

Systems of Boolean Equations

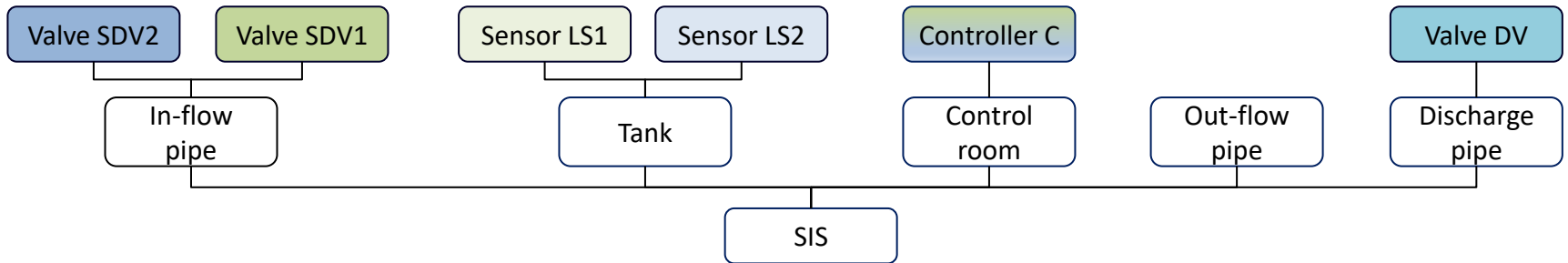
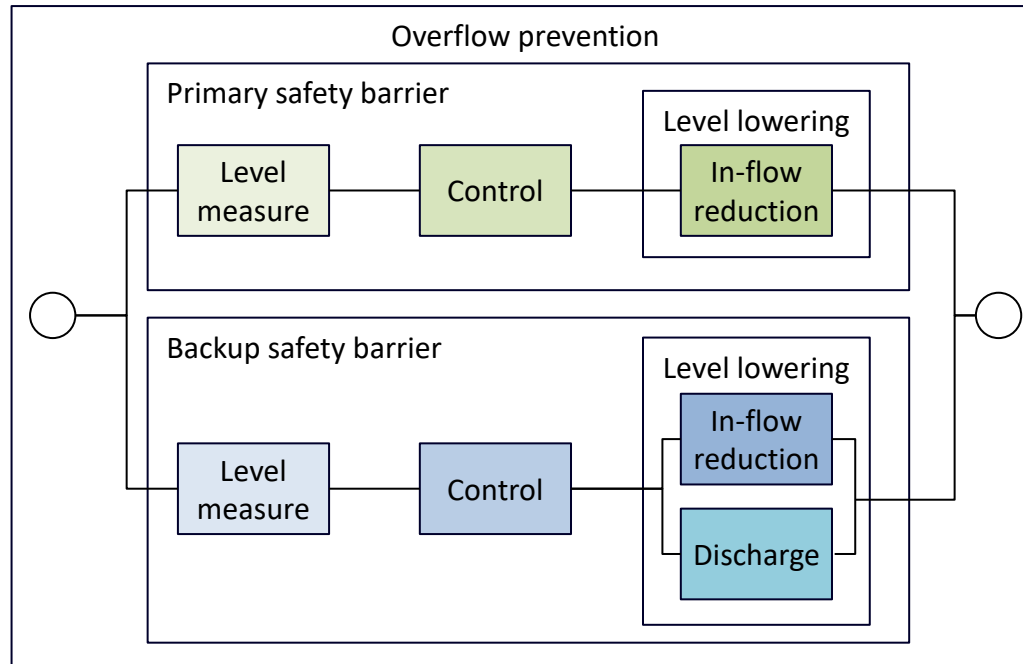
Any reliability block diagram is the **graphical representation** of a uniquely rooted, data flow Boolean circuit. Again, the real model is the system of Boolean equations.



```
SB2.out = SDV2.out or DV.out
SDV2.out = SDV2.in and not SDV2.failed
SDV2.in = C.out
DV.Out = DV.in and not DV.failed
DV.In = C.out
...
```

Reliability block diagrams	Boolean equations
Basic block states	State variables
Ports	Flow variables
Target node	Root variable
Source node(s)	Source variables

Logical Architecture of the System (2)



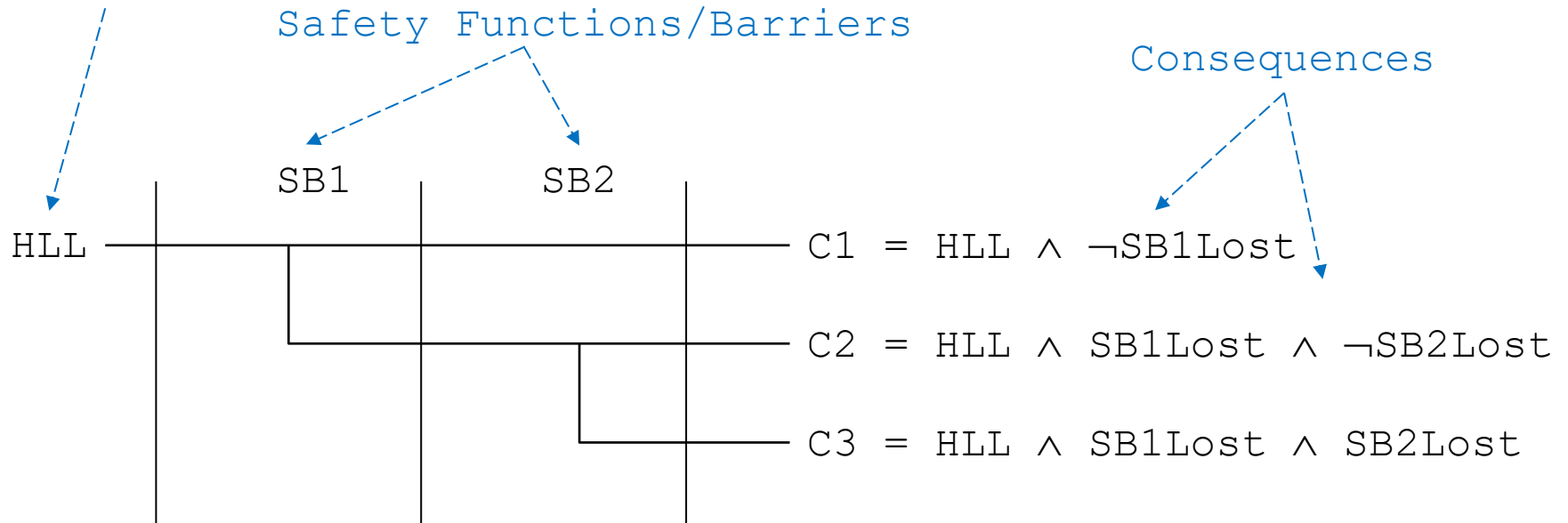
SECTION 4.3. EVENT TREES

Event Trees

Initiating
Event

Safety Functions/Barriers

Consequences



SECTION 4.4. DISCUSSION

Models and Algorithms

The principle of combinatorial models is as follows.

- The failure of components of the system are described by **basic events** (Boolean variables). Basic events are events in the sense of **probability theory**.
- Basic events are assumed to be **statistically independent** (introducing dependencies makes the model leave the realm of combinatorial models).
- A probability is associated with each basic event of the model (possibly via a time-dependent distribution considered at the mission time).
- A **system of Boolean equations** describe how combinations of basic events produce **loss of capabilities** in the system, including its main capability.

Thanks to **specific assessment algorithms**, it is possible:

- To extract **minimal cutsets**, i.e. sets of basic events that induce the loss of the main capability of system.
- To calculate various **reliability indicator**, including the **probability of loss of the main capability** of the system.

Discussion

Summary:

- Combinatorial models are a **good compromise** between expressivity of the models and computational complexity of assessment algorithms.
- They are by far the **most widely used** in industry.

A full exposition of the mathematical and algorithmic framework of combinatorial models goes beyond this course.

Recommended readings:

- Antoine Rauzy. Probabilistic Safety Analysis with XFTA. AltaRica Association. 2020.isbn: 978-82-692273-0-7
- Antoine Rauzy and Liu Yang. Finite Degradation Structures. In *Journal of Applied Logics – IfCoLog Journal of Logics and their Applications*. College Publications. Vol. 6, Num. 7, pp 1471–1495, 2019.

CHAPTER 10. RELIABILITY ENGINEERING

SECTION 5. STATE AUTOMATA

SECTION 5.1. DEPENDENCIES

What Makes Events Dependent?

Common Cause Failures:

- Design flaws or manufacturing defects.
- Environmental events, e.g impacts, flooding, fire...
- Common external resource, e.g. power supply.
- Maintenance or human error.

Cascading Failures:

- E.g. pump badly restarted → gas leak → explosion disables firewater system → ... (Piper Alpha, 1988)

Temporal dependencies:

- Cold and warm redundancies
- Fatigue and cumulated damage
- Backup systems with limited time effectiveness (e.g. batteries)
- Maintenance operations

Shared resources:

- Spare parts
- Maintenance crew

...S

SECTION 5.2. FINITE STATE AUTOMATA

Finite State Automata

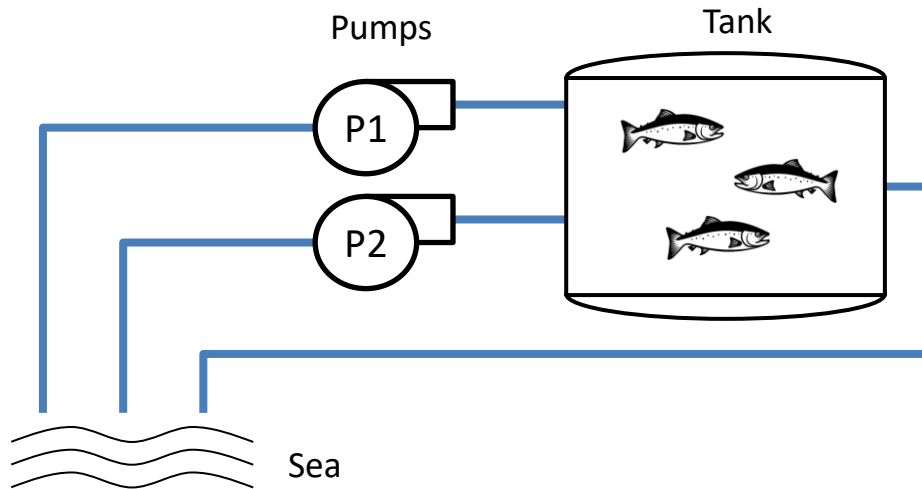
A **finite state automaton** is a quadruple $\langle S, E, T, i \rangle$ where:

- S is a finite set of **states**.
- E is a finite set of **events**.
- T is a finite set of **transitions**, i.e. of triples $\langle e, s, t \rangle$, where:
 - e is an event of E .
 - s and t are states of S , s is called the **source state** and t the **target state** of the transition.
- i is a state of S , called the **initial state**.

It is assumed moreover, that each state s has a finite number of out going transitions (transitions whose source state is s).

For the sake of the clarity, transitions $\langle e, s, t \rangle$ are denoted $e: s \rightarrow t$ or $s \xrightarrow{e} t$.

On-Shore Fish Farm



Maintenance technician

- Pumps P1 and P2 may fail and be repaired by the maintenance technician T
- The maintenance technician can repair only one pump at a time

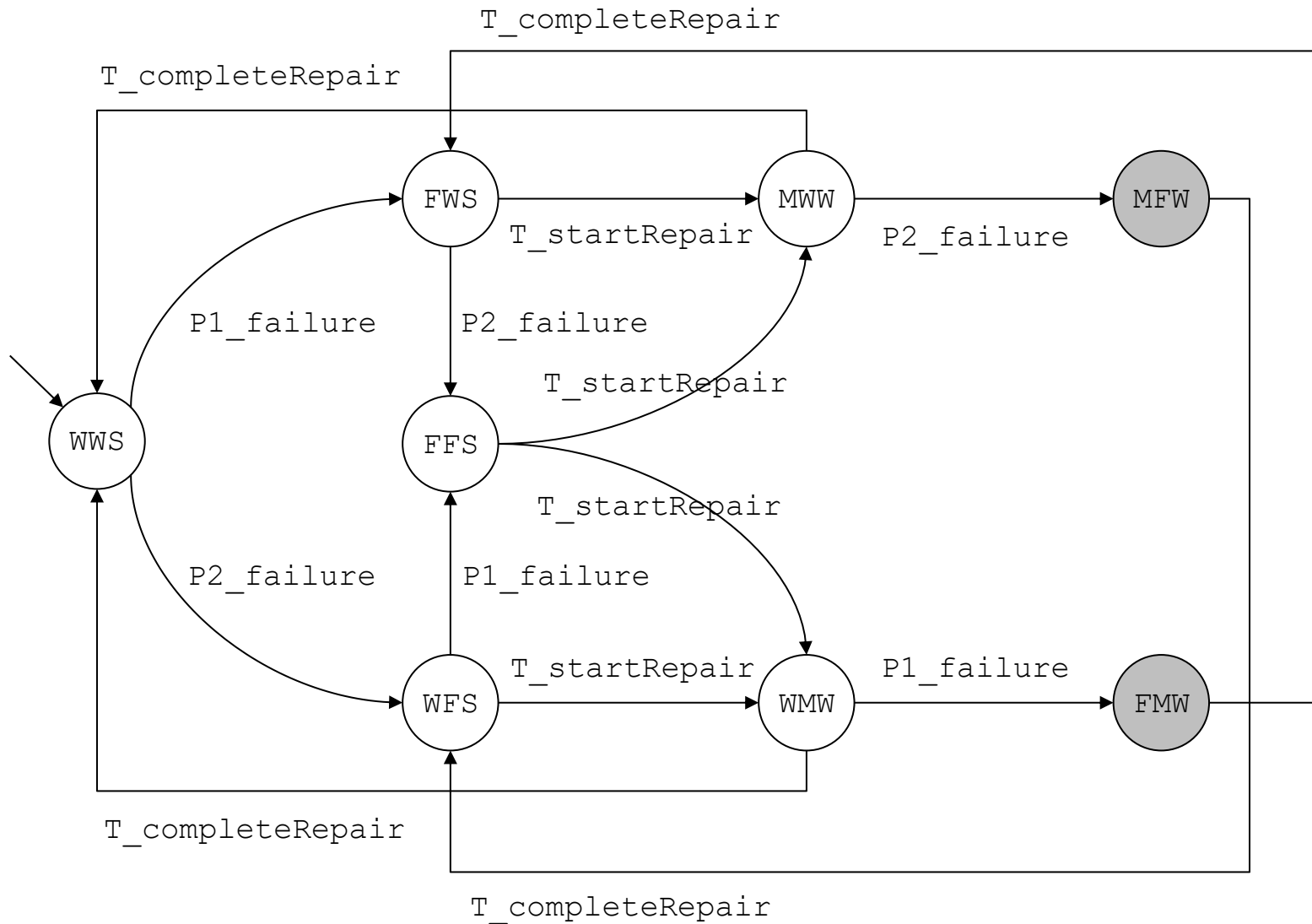
Pumps:

- W: working
- F: failed
- M: Maintenance

Maintenance technician:

- S: standby
- W: working

On-Shore Fish Farm: Finite State Automaton



Executions

The **semantics** of a finite state automaton is defined as its set of **executions**. This set of executions is called the **language** of the finite state automaton.

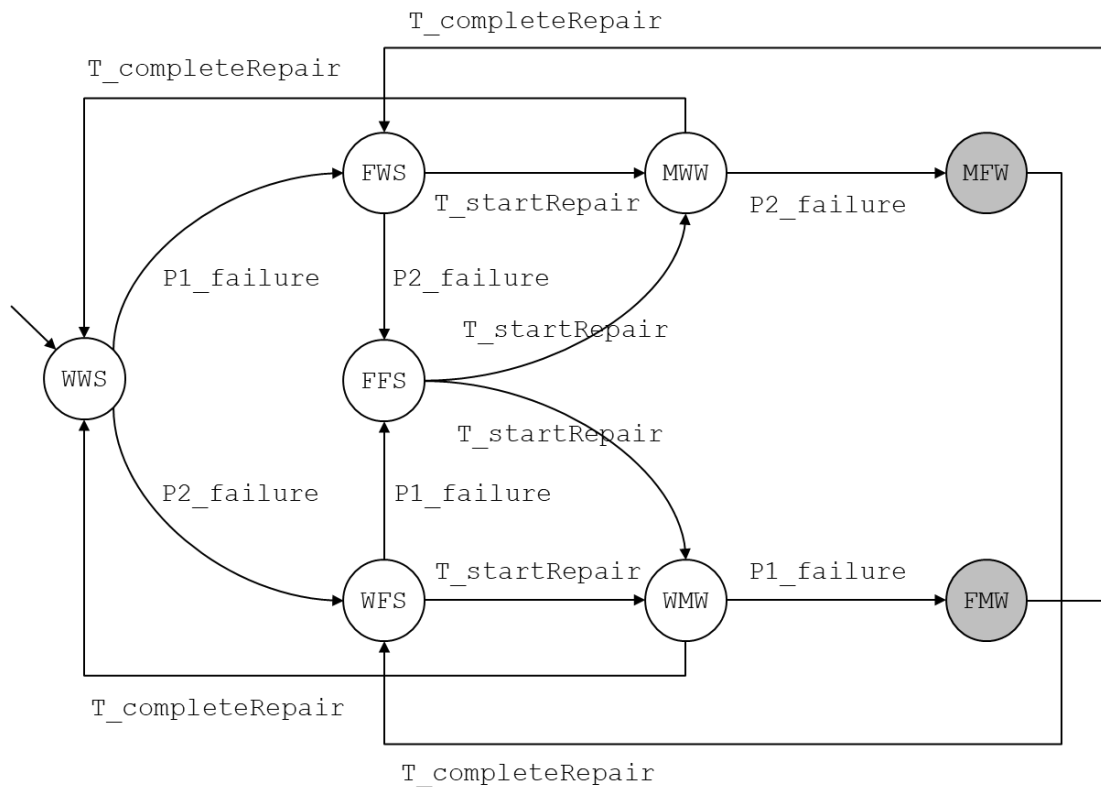
Let $M: \langle S, E, T, i \rangle$ be a finite state automaton. The set of executions of M , denoted by $[|M|]$, is the smallest set such that:

- The initial state i alone is an execution, called the **empty execution** of M .
- If $\sigma = i \xrightarrow{e_1} s_1 \xrightarrow{e_2} s_2 \cdots \xrightarrow{e_n} s_n$, $n \geq 0$, is an execution of M and $s_n \xrightarrow{e} t$ is a transition of T , then $\sigma \rightarrow t$ is an execution of M .

According to our definition, all executions of a finite state automaton are **finite**, although they may be **arbitrary long**, i.e. contain an arbitrary large number of transitions. We denote by $|\sigma|$ the length, i.e. the number of transitions, of the execution σ .

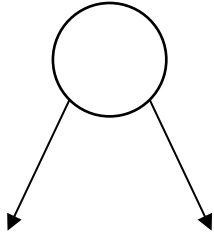
A finite state automaton is **deterministic** if at each step only one transition is possible. It is **non-deterministic** otherwise. Most, if not all, models we shall consider are non-deterministic.

On-Shore Fish Farm: Executions

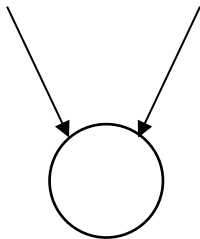


P1_failure, T_startRepair, P2_failure, T_completeRepair...

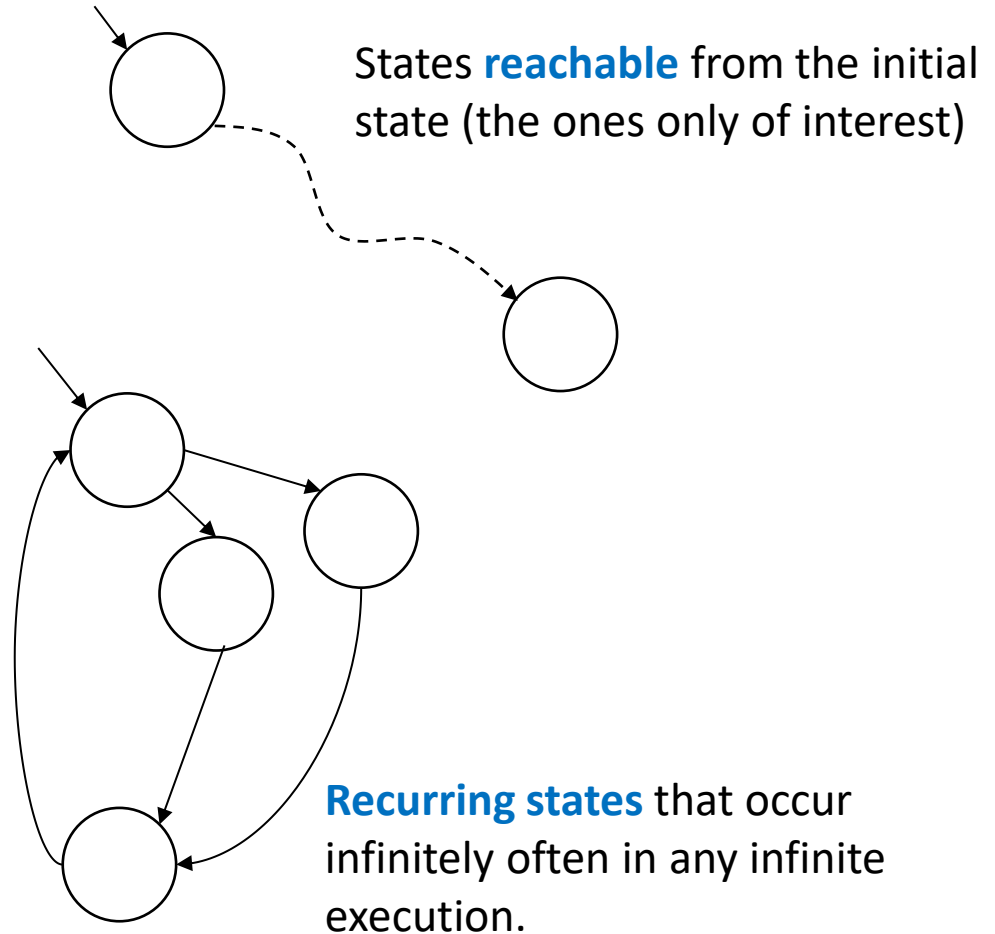
Classes of States



Source states: states with no in coming transitions



Sink states: states with no out-going transitions



Extensions

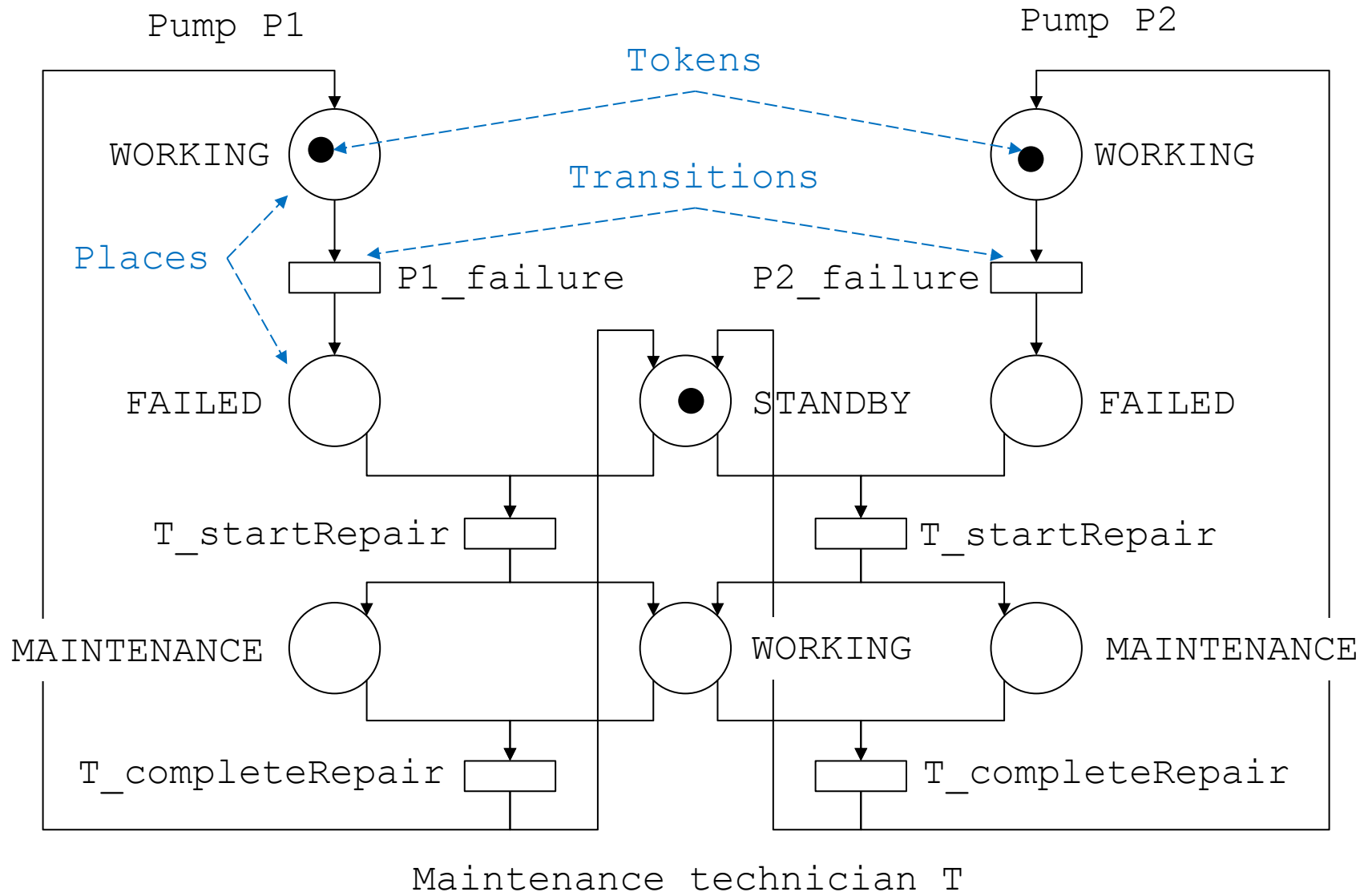
The concept of finite state automata can be extended in two independent directions.

In the definition of finite state automata we gave, states and transitions are **explicitly** given. For large automata, this is both tedious and error prone. This is the reason why, for practical applications, **implicit** descriptions are in general preferred.

Reliability engineering is all about **probabilities**. Any industrial system may fail; the only questions are with which probability and which consequences. We shall see several extension to make (explicitly or implicitly described) finite state automata probabilistic.

SECTION 5.3. IMPLICIT DESCRIPTIONS

Petri Nets



Guarded Transition Systems

A **guarded transitions system** is a quintuple $\langle V, E, T, s_0 \rangle$, where:

- V is a set of **variables**. Each variable of V takes its value into a finite or infinite set $\text{dom}(V)$ of constants called the **domain** of V .
- E is a set of **events**.
- T is a set of **transitions**, i.e. of triples $e: g \rightarrow a$, where:
 - e is an event of E ,
 - g is a Boolean expression built on variables of V , and
 - a is an instruction modifying the values of (some of) the variables.
- s_0 is a valuation of variables of V , called the **initial state**.

States of the system are thus represented by **valuations** of variables of V .

A transition $e: g \rightarrow a$ is **enabled** in the state s , if $s(g) = \text{true}$.

Firing the transition $e: g \rightarrow a$ in the state s transforms s into $s' = a(s)$

On-Shore Fish Farm: Guarded Transition System

```
domain PumpState {WORKING, FAILED, MAINTENANCE}
domain TechnicianState {STANDBY, WORKING}

block FishFarm
  PumpState P1, P2 (init = WORKING);
  TechnicianState T (init = STANDBY);
  event P1_failure, P2_failure, T_startRepair, T_completeRepair;
  transition
    P1_failure: P1==WORKING -> P1 = FAILED;
    P2_failure: P2==WORKING -> P2 = FAILED;
    T_startRepair: P1==FAILED and T==STANDBY -> {
      P1 = MAINTENANCE; T = WORKING; }
    T_startRepair: P2==FAILED and T==STANDBY -> {
      P2 = MAINTENANCE; T = WORKING; }
    T_completeRepair: P1==MAINTENANCE and T==WORKING -> {
      P1 = WORKING; T = STANDBY; }
    T_completeRepair: P2==MAINTENANCE and T==WORKING -> {
      P2 = WORKING; T = STANDBY; }
end
```

Executions and Reachability Graph

The **semantics** of guarded transition systems is defined in terms of executions.

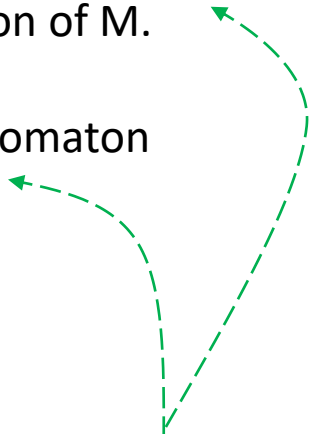
Let $M: \langle V, E, T, s_0 \rangle$ be a guarded transition system. Then, the set of **executions** of M is the smallest set such that:

- s_0 is an execution (the empty execution)
- If $s_0 \xrightarrow{e_1} s_1 \xrightarrow{e_2} \dots \xrightarrow{e_n} s_n$, $n \geq 0$, is an execution of M and there is a transition $e: g \rightarrow a$ of T enabled in s_n , then $s_0 \xrightarrow{e_1} s_1 \xrightarrow{e_2} \dots \xrightarrow{e_n} s_n \xrightarrow{e} s_{n+1} = a(s_n)$ is an execution of M .

The **reachability graph** $\langle \Sigma, E, \Theta, s_0 \rangle$ of M is the smallest finite state automaton such that:

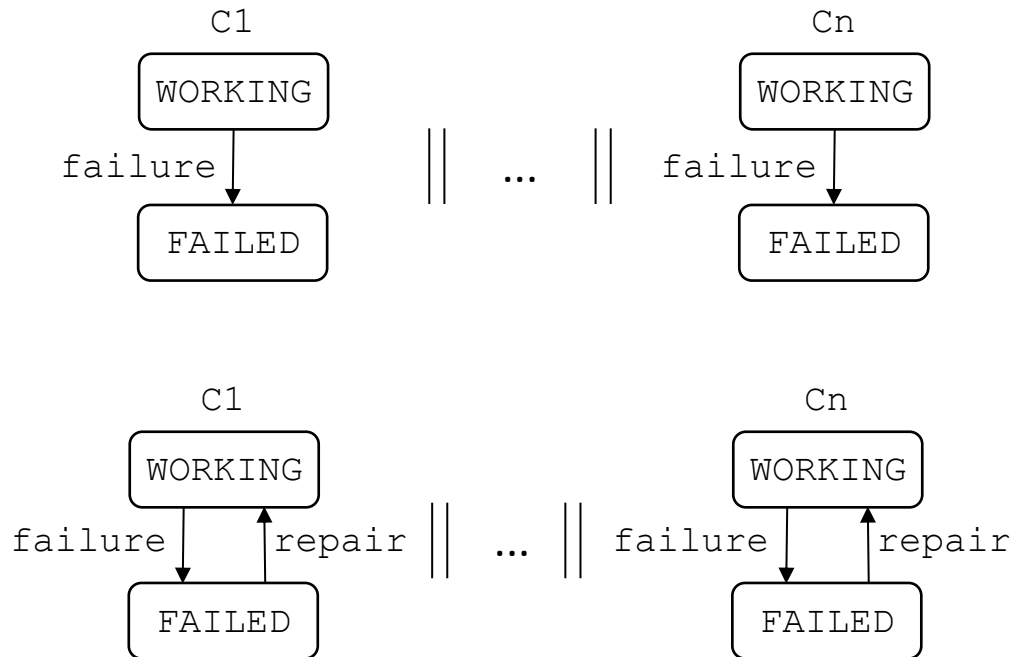
- $s_0 \in \Sigma$
- If $s \in \Sigma$ and there is a transition $e: g \rightarrow a$ of T enabled in s , then:
 - $t = a(s) \in \Sigma$
 - $s \xrightarrow{e} t \in \Theta$

Fixpoint definitions



Combinatorial Explosion

Guarded Transition System



n independent components

Reachability Graph

	n
States	2^n
Transitions	$\frac{n}{2} \times 2^n$
Executions	$n!$

	n
States	2^n
Transitions	$n \times 2^n$
Executions	∞

SECTION 5.4. PROBABILISTIC STATE AUTOMATA

SECTION 5.4.1. DISCRETE-TIME MARKOV CHAINS

Discrete-Time Markov Chains (1)

Markov chains are pervasive in science and engineering. The classical definition of discrete-time Markov chains goes as follows.

Let Ω be a set of outcomes and $\mathcal{F}$ be a σ -algebra on Ω . Let T be a set of indices. Then, a **stochastic process** on $\langle \Omega, \mathcal{F} \rangle$ is a T -indexed family $\{X_t; t \in T\}$ of random variables built on $\langle \Omega, \mathcal{F} \rangle$.

A stochastic process is **discrete** if $T = \mathbb{N}$.

A discrete-stochastic process is a **discrete-time Markov chain** if it satisfies the Markov condition:

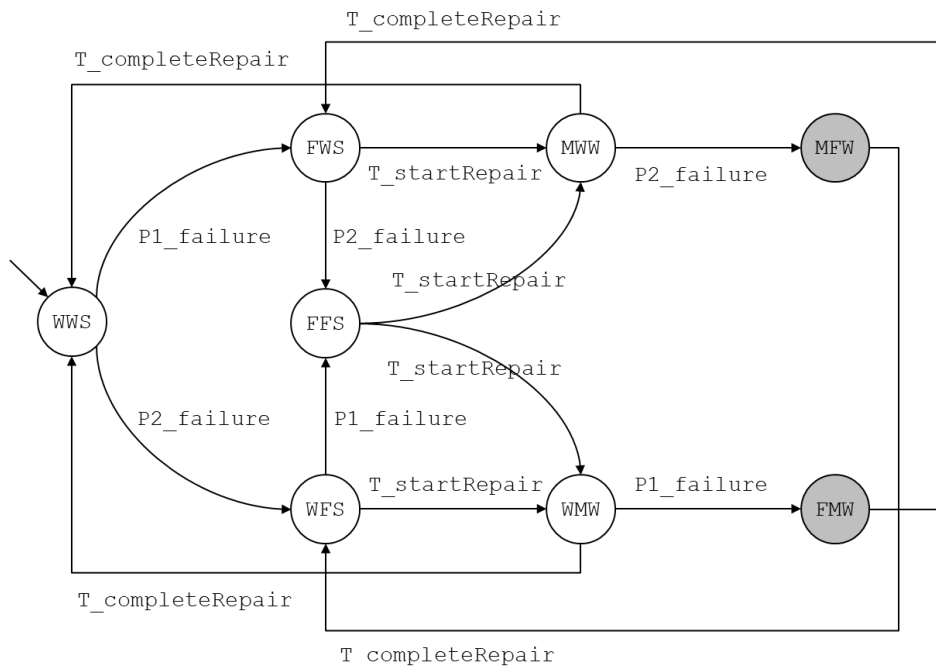
$$p(X_n = x_n | X_0 = x_0, X_1 = x_1, \dots, X_{n-1} = x_{n-1}) = p(X_n = x_n | X_{n-1} = x_{n-1})$$

for all $n \geq 1$ and all $x_0, x_1, \dots, x_n$ in Ω .

Discrete-Time Markov Chains (2)

In practice, a **discrete-time Markov chain** (with a finite set of outcomes) can be seen as **probabilistic finite state automaton** $\langle S, E, T, P, I \rangle$ where:

- $\langle S, E, T \rangle$ is a finite state automaton.
- P is a function from T into $[0, 1]$ such that $\sum_t P\left(s \xrightarrow{e} t\right) \leq 1$, for all state $s \in S$.
- I is a function from S into $[0, 1]$ such that $\sum_s P(s) = 1$



	P
P1_Failure	$1.66 \cdot 10^{-5}$
P2_Failure	$1.66 \cdot 10^{-5}$
T_startRepair	0.5
T_completeRepair	0.125

	I
WWS	1
others	0

Probability Matrices

The equation defining $p_s(n)$ in terms of $p_s(n - 1)$ gives rise to the **probability matrix** presentation of discrete-time Markov chains found in most of the textbooks.:

$$P_{s,s'} \stackrel{\text{def}}{=} \sum_{\substack{e \\ s \xrightarrow{e} s'}} p(s \xrightarrow{e} s')$$

If $\pi(n - 1)$ and $\pi(n)$ as vectors of probability of states. we can write:

$$\pi(n) = \pi(n - 1) \times P$$

Therefore,

$$\pi(n) = \pi(0) \times P \times \cdots \times P = \pi(0) \times P^n$$

Such a presentation, although convenient for pedagogical purposes, is however misleading:

- The matrix P is in general **sparse** (all but a few cells are 0's).
- The power of a sparse matrix is in general not sparse (as it describes paths).
- In practice, calculations are made by multiplying vectors by sparse matrices.

Assessment Algorithm

```
 $\pi = \pi_0$  // the vector of probability  $\pi$  is filled with  
           // initial probabilities  
  
for i=1 to n do: // n is the number of steps  
     $\rho = \pi$   
    for all  $s \xrightarrow{e} t$  in the chain  
         $\pi[s] = \pi[s] - \rho[s] \times p(s \xrightarrow{e} t)$   
         $\pi[t] = \pi[t] + \rho[s] \times p(s \xrightarrow{e} t)$   
    if  $\rho \approx \pi$  then // the situation does not evolve  
        break
```

The $p_s(n)$'s are called **transient probabilities**.

If the situation stabilized, i.e. when $p_s(n) = p_s(n - 1)$ for all states s , the $p_s(n)$'s are called **steady state probabilities**.

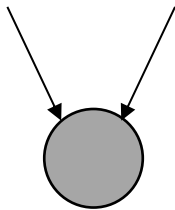
This algorithm is very efficient... but works on the reachability graph of the model.

Steady State Probabilities

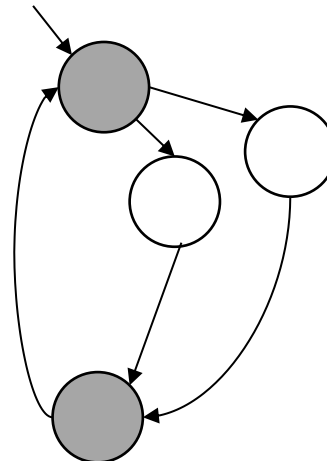
Steady state probabilities exist if the equation $\pi \times P = \pi$ has a solution (in which case this solution is unique).

The above equation can be solved by linear algebra techniques, e.g. Gauss-Jordan elimination. However, this transforms usually sparse matrices into non-sparse one. This is the reason why the previous algorithm is in general more efficient.

Reasons for which there is no steady state probabilities:



Sink states: states with no out-going transitions



Recurring states that occur infinitely often in any infinite execution.

SECTION 5.4.2. CONTINUOUS-TIME MARKOV CHAINS

Markov Property (again)

We want to extend the **Markov property** to the continuous time case, i.e. intuitively that the probability to jump from state s' to state s does not depend on the execution prior arriving in state s' , i.e.

$$p(X(t) = s \mid X(t') = s', X(t'') = s'') = p(X(t) = s \mid X(t') = s')$$

for all states s, s' and s'' , and all time $t > t' > t'' \geq 0$.

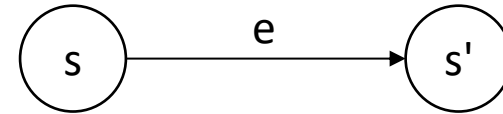
Moreover, we want this property to be **time-homogeneous**, i.e.

$$p(X(t + t') = s \mid X(t') = s') = p(X(t) = s \mid X(0) = s')$$

for all states s and s' , and all time $t, t' \geq 0$.

Continuous "Memorylessness"

Consider the simple finite state automaton:



Let us denote by T the time spent in state s . By time homogeneity, we should have:

$$p(T > t + t' \mid T > t') = p(T > t)$$

Applying the definition of conditional probabilities, we have:

$$p(T > t + t' \mid T > t') = \frac{p(T > t + t')}{p(T > t')}$$

Therefore:

$$p(T > t + t') = p(T > t) \times p(T > t')$$

The only continuous function that satisfies this equation for any positive t is the exponential:

$$p(T > t) = e^{-\lambda \cdot t} \quad \text{where} \quad \lambda = -\log(p(T > 1))$$

Continuous-Time Markov Chains

Let $M: \langle S, E, T, \delta, \omega, i \rangle$ be a stochastic finite state automaton, then M is a **continuous-time Markov chain** if:

- $\delta(e, \sigma)$ is the inverse of a negative exponential distribution of parameter λ , where λ depends only e , i.e.

$$\delta(e, \sigma) = -\frac{\log z}{\lambda} \quad \text{for some number } z \text{ chosen uniformly at random in } [0, 1]$$

- $\omega(e, \sigma)$ plays no role (as the probability that two transitions have the same delay is null).

Remarks:

- Again, the above definition is simpler than the definition one can find in most of the textbooks. It is constructive and involves only finite objects.

Assessment Algorithm

Calculating transient probabilities of a continuous-time Markov chain requires solving (numerically) a system of differential equations, called Chapman-Kolmogorov equations.

It is possible to discretize the problem and to use the algorithm for discrete-time Markov chains to solve the Chapman-Kolmogorov equations.

- William J. Stewart. Introduction to the Numerical Solution of Markov Chains. Princeton University Press. Princeton, New Jersey, USA. ISBN 978-0691036991. 1994.
- Antoine Rauzy. An Experimental Study on Six Algorithms to Compute Transient Solutions of Large Markov Systems. Antoine Rauzy
In Reliability Engineering and System Safety. Elsevier. Vol. 86, Num. 1, pp 105–115, October 2004. doi: 10.1016/j.ress.2004.01.007

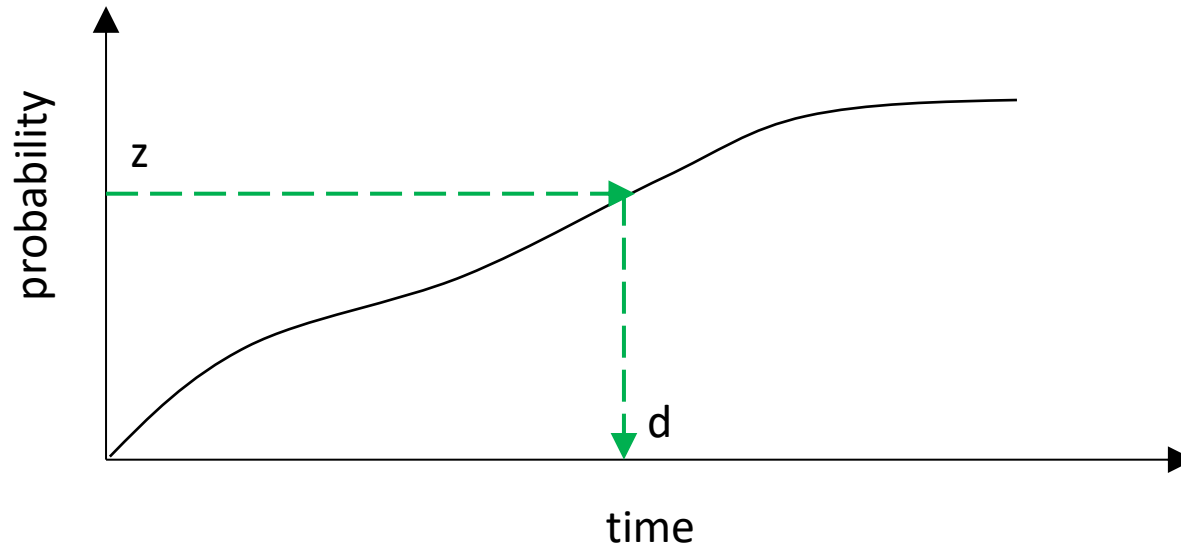
The resulting algorithms are efficient... but work on the reachability graph of the model.

SECTION 5.4.3. STOCHASTIC GUARDED TRANSITION SYSTEMS

Delays

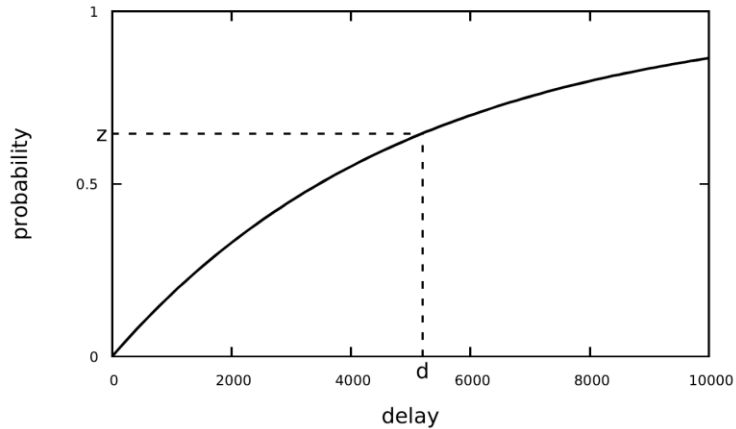
Delays are associated with events.

They are inverse functions of cumulative probability distributions:

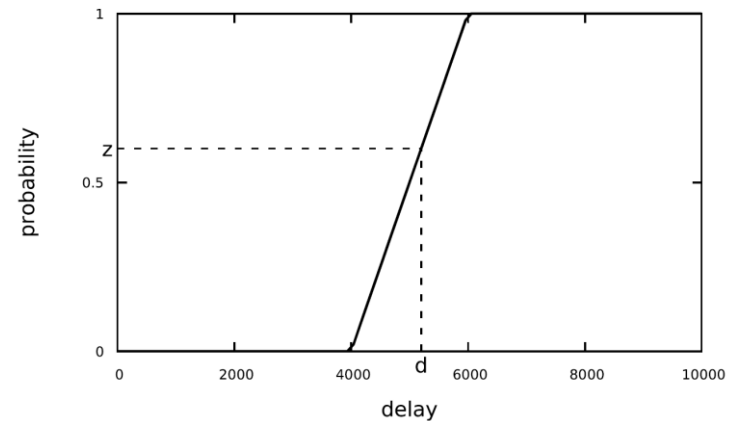


Cumulative Probability Distributions

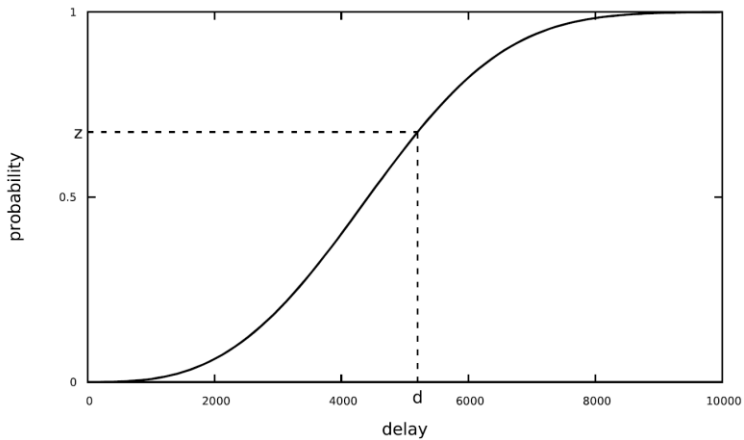
Exponential



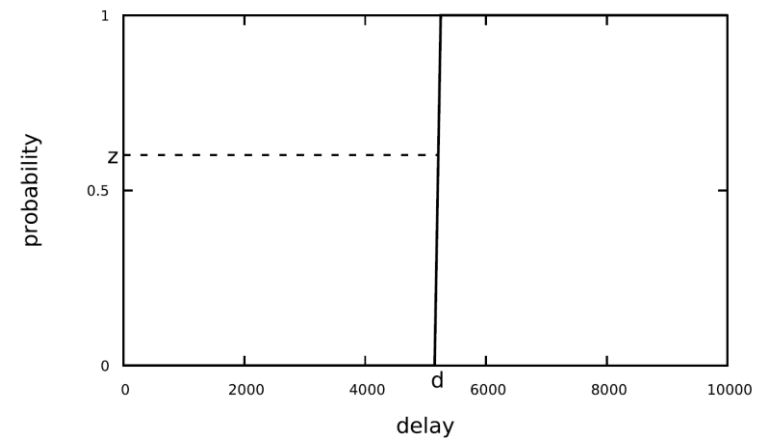
Uniform



Weibull



Dirac



Non-Timed Semantics

Let $M: \langle V, E, T, A, s_0 \rangle$ be a guarded transitions system.

The semantics of M is defined as the smallest set of executions such that:

- s_0 is the empty execution.
- If $\sigma: s_0 \xrightarrow{e_1} s_1 \cdots \xrightarrow{e_{n-1}} s_{n-1}$ is an execution, $n \geq 1$, $e: g \rightarrow a$ is a transition of T then $\sigma \xrightarrow{e} s_n$ is an execution if:
 - Condition 1: $e: g \rightarrow a$ is enabled in state s_{n-1} , i.e. if $g(s_{n-1}) = \text{true}$
 - Condition 2: $s_n = A(a(s_{n-1}))$

Timed Semantics

A schedule Γ is a function from $dom(V) \times T$ to $\mathbb{R}^+ \cup \{\infty\}$ such that:

- $\Gamma(s, t) \in \mathbb{R}^+$ if the transition t is enabled in the state s .
- $\Gamma(s, t) = \infty$ if the transition t is not enabled in the state s .

Let $M: \langle V, E, T, s_0 \rangle$ be a guarded transitions system.

The semantics of M is defined as the smallest set of executions such that:

- $\langle s_0, 0, \Gamma_0 \rangle$ is the empty execution.
- If $\sigma: \langle s_0, 0, \Gamma_0 \rangle \xrightarrow{e_1} \langle s_1, d_1, \Gamma_1 \rangle \cdots \xrightarrow{e_{n-1}} \langle s_{n-1}, d_{n-1}, \Gamma_{n-1} \rangle$ is a timed execution, $n \geq 1$, $e: g \rightarrow a$ is a transition of T then $\sigma \xrightarrow{e} \langle s_n, d_n, \Gamma_n \rangle$ is a timed execution if:
 - Condition 1: $e: g \rightarrow a$ is enabled in state s_{n-1} , i.e. if $g(s_{n-1}) = true$
 - Condition 2: $s_n = a(s_{n-1})$
 - Condition 3: t is one of the transitions with the earliest schedule dates.
 - Condition 4: Γ_n is obtained from Γ_{n-1} (see book).

On-Shore Fish Farm: Guarded Transition System

```
domain PumpState {WORKING, FAILED, MAINTENANCE}
domain TechnicianState {STANDBY, WORKING}

block FishFarm
  PumpState P1, P2 (init = WORKING);
  TechnicianState T (init = STANDBY);
  event P1_failure, P2_failure (delay = Weibull(2000, 3));
  event T_startRepair (delay = Dirac(0));
  event T_completeRepair (delay = Uniform(8, 12));
  transition
    P1_failure: P1==WORKING -> P1 = FAILED;
    P2_failure: P2==WORKING -> P2 = FAILED;
    T_startRepair: P1==FAILED and T==STANDBY -> {
      P1 = MAINTENANCE; T = WORKING; }
    T_startRepair: P2==FAILED and T==STANDBY -> {
      P2 = MAINTENANCE; T = WORKING; }
    T_completeRepair: P1==MAINTENANCE and T==WORKING -> {
      P1 = WORKING; T = STANDBY; }
    T_completeRepair: P2==MAINTENANCE and T==WORKING -> {
      P2 = WORKING; T = STANDBY; }
end
```

Discussion

- Stochastic guarded transition systems are at the core of the **AltaRica 3.0** modeling language, a full-fledged object-oriented language.
- AltaRica models are assessed by means of Monte-Carlo simulations.
- Any resemblance to Σ^{TM} is absolutely not coincidental: the AltaRica technology served as the basis for the development of Σ^{TM} and WordlabTM.

More on Model-Based Reliability Engineering:

- Antoine Rauzy. Model-Based Reliability Engineering – An Introduction from First Principles. AltaRica Association. 2022. isbn: 978-82-692273-2-1

CHAPTER 10. RELIABILITY ENGINEERING

SECTION 6. Σ^{TM} PATTERNS

Shared Maintenance Crew

```
class MaintenanceCrew
  MaintenanceCrewState state(init = STANDBY);
  virtual ComponentState stateComponent1(init = WORKING);
  virtual ComponentState state2(init = WORKING);
end

activity MaintenanceCrew.RepairComponent1
  trigger:
    return state==STANDBY and stateComponent1==FAILED;
  start: {
    state = WORKING;
    stateComponent1 = MAINTENANCE;
  }
  completion: {
    state = STANDBY;
    stateComponent1 = STANDBY;
  }
  duration:
    return 8;
end
```

Preventive Maintenance

```
class Unit
  UnitState state(init = STANDBY);
  ...
end

activity Unit.Operate
  float operationDuration(init = 0);
  trigger:
    return state==STANDBY;
  start: {
    timeToFailure = exponentialDeviate(failureRate);
    operationDuration = min(timeToPreventiveMaintenance, timeToFailure);
    state = WORKING;
  }
  completion: {
    if operationDuration==timeToPreventiveMaintenance
    then state = STANDBY;
    else state = FAILED;
  }
  duration:
    return operationDuration;
end
```

Failure On Demand

```
domain ValveState {MOVING, STUCK}
domain ValveMode {OPEN, CLOSED}

class Valve
  ValveState state(init = MOVING);
  ValveMode mode(init = OPEN);
  port ValveMode requestedMode = OPEN;
  bool operating(init = false);
  ...
end

activity Valve.Closing
  trigger:
    return not operating and mode==OPEN and requestedMode==CLOSED and
      state==MOVING;
  start:
    operating = true;
  completion: {
    if uniformDeviate(0.0, 1.0)<failureOnDemandProbability
    then state = STUCK;
    else mode = CLOSED;
    }
  duration:
    return 0;
end
```

Reliability Indicators

```
system OverflowProtectionSystem
```

```
...
```

```
  port bool failed = ...;
```

```
end
```

```
system OverflowProtectionSystem
```

```
  observer unavailability = if failed then 1 else 0;
```

```
  observer unreliability = if occasionally(failed) then 1 else 0;
```

```
  observer meanDowntime = sojournTime(failed);
```

```
  observer meanTimeToFailure = firstRealizationTime(failed);
```

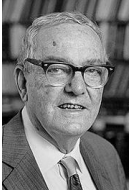
```
  observer expectedNumberOfFailures = numberOfRealizations(failed);
```

```
end
```

CHAPTER 5. RELIABILITY ENGINEERING

SECTION 7. DISCUSSION

Uncertainties



Herbert Simon
(1916-2001)
Bounded Rationality

