Reliability Assessment of Complex Systems

Prof. Antoine B. Rauzy

Department of Production and Quality Engineering
Norwegian University of Science and Technology
Trondheim, Norway

&

Chair Blériot-Fabre
CentraleSupélec / SAFRAN
Paris, France

Reliability Assessment

We shall use "**Reliability Assessment**" as a generic term for the (probabilistic) assessment of:

- Risk
- Reliability
- Availability
- Maintainability
- Safety
- Safety Integrity Levels
- Production assurance / production availability
- …

i.e.

- All types of **(probabilistic) performance indicators** of systems subject to failures, defects, human errors, degraded environmental conditions…

Complex Systems

We shall use "**Complex Systems**"* as a generic term to describe technical systems that involve:

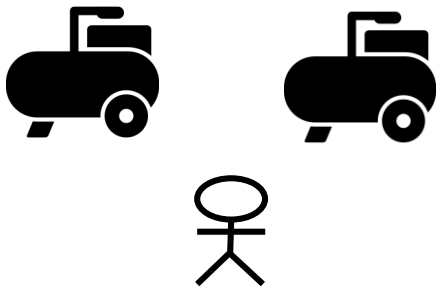
- Dependent and/or cascading failures,
- Cold redundancies and/or spare parts,
- Limited access to some resources,
- Reconfigurations and/or maintenance strategies,
- Production levels,
- Feedback loops,
- …

i.e. systems for which reliability assessment by means of Fault Trees (or any equivalent formalism) is hardly possible or give **too coarse** (**over pessimistic**) **results** typically because of dependencies amongst events.

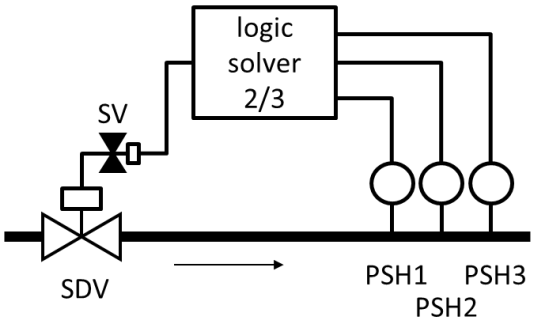
(*) Some authors would define the above systems "just" as complicated. But as we shall see, the reliability assessment of these systems is provably hard (complex) in the sense of computational complexity theory.

Typical Modeling Issues

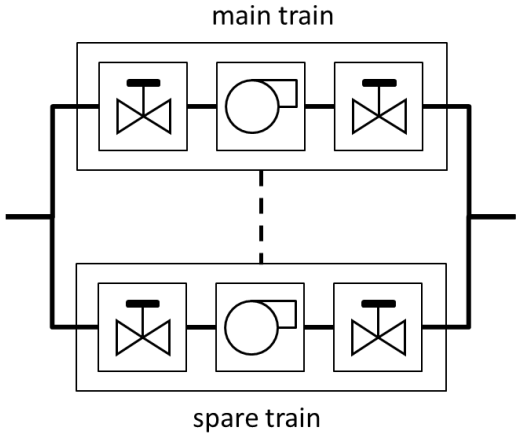
How to model shared/limited resources?



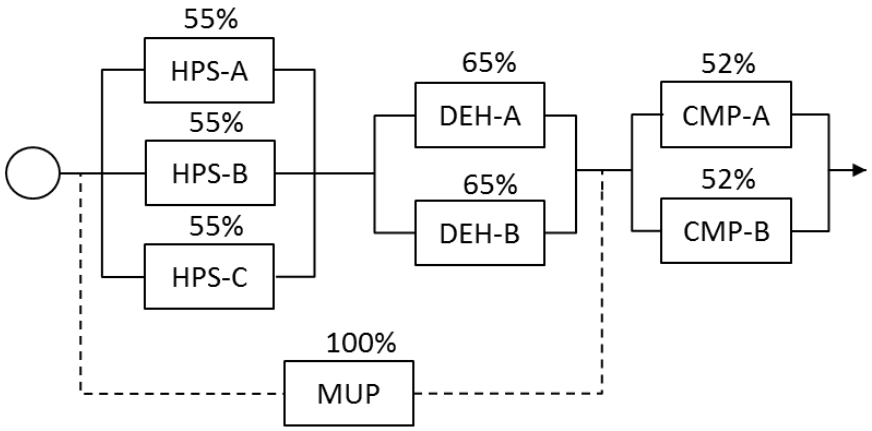
How to model maintenance policies?



How to model system reconfigurations?



How to model production availability?



Questions:

- Is there a generic/unified **modeling framework** to assess reliability of complex systems?
- What are the available **algorithmic tools** to assess reliability of complex systems?

Agenda

Part 1. The reliability assessment process

Part 2. Fundamental complexity results (the bad news)

Part 3. Model-Based Systems Engineering (the good news)

Wrap-Up & Challenges

Agenda

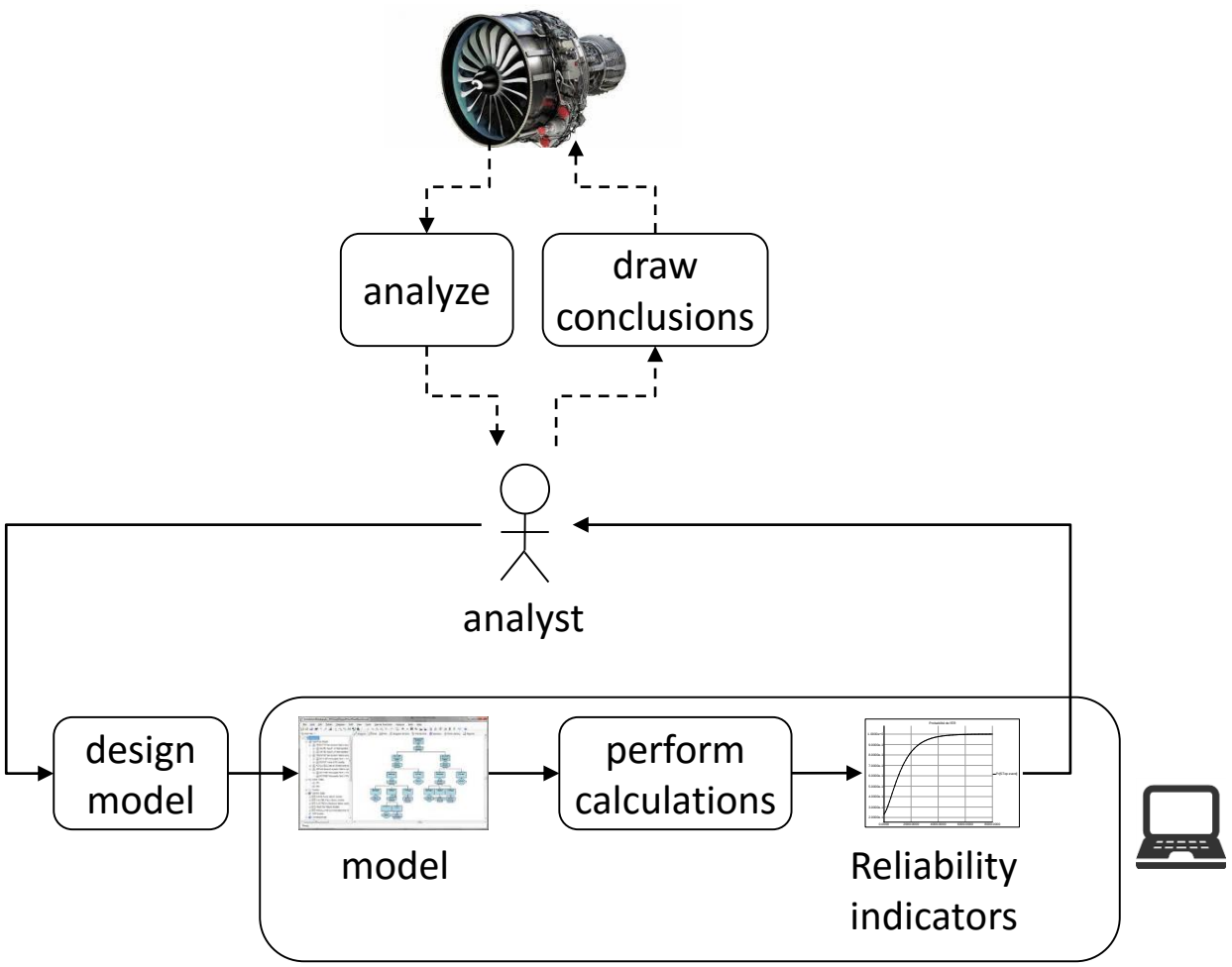
Part 1. The reliability assessment process

Part 2. Fundamental complexity results (the bad news)

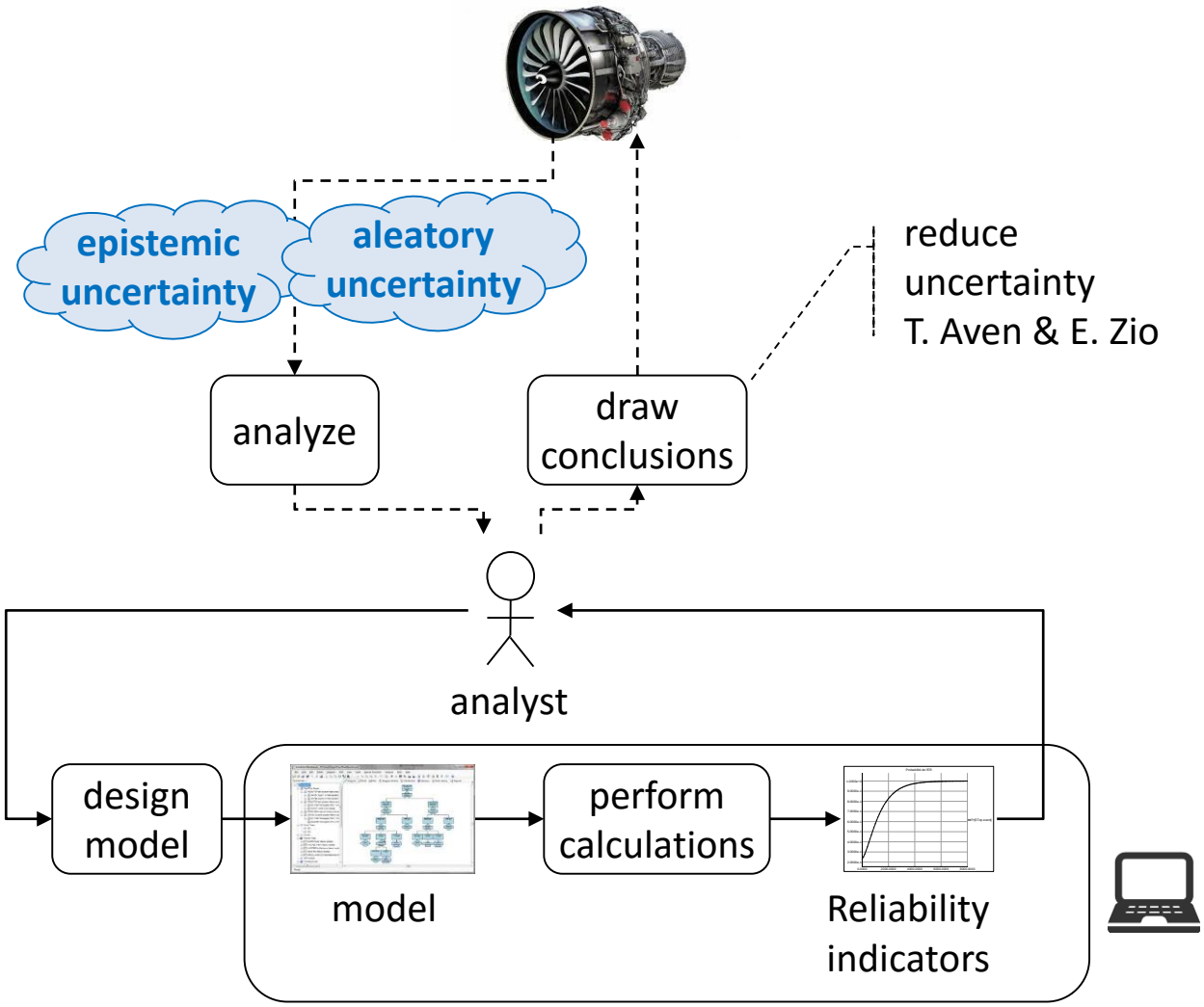
Part 3. Model-Based Systems Engineering (the good news)

Wrap-Up & Challenges

The Reliability Assessment Process

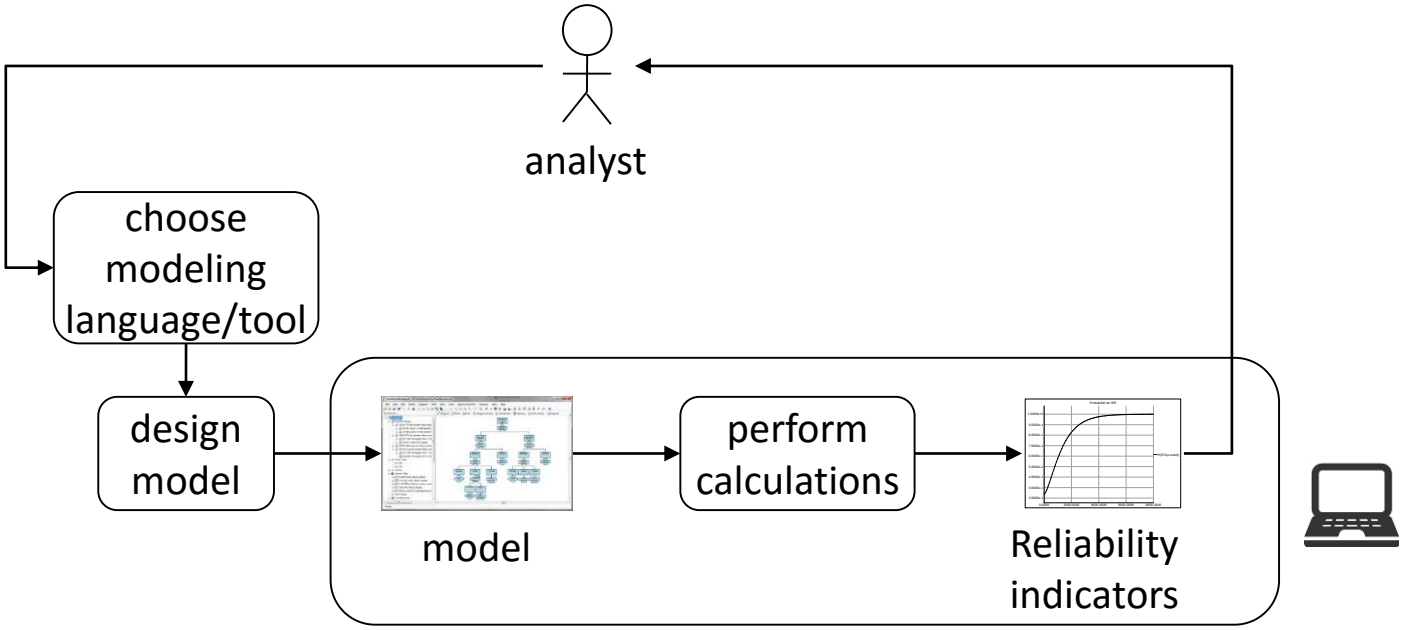


Uncertainties



Modeling Languages and Tools

To design a model, we need a **modeling language** (would it be graphical), just as to design a program, we need a programming language.

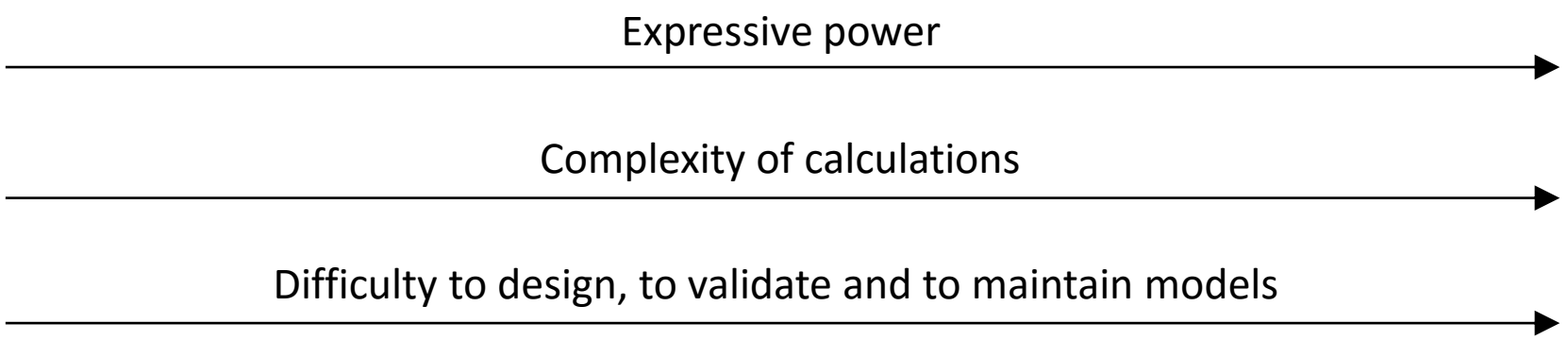


Categories of Modeling Languages

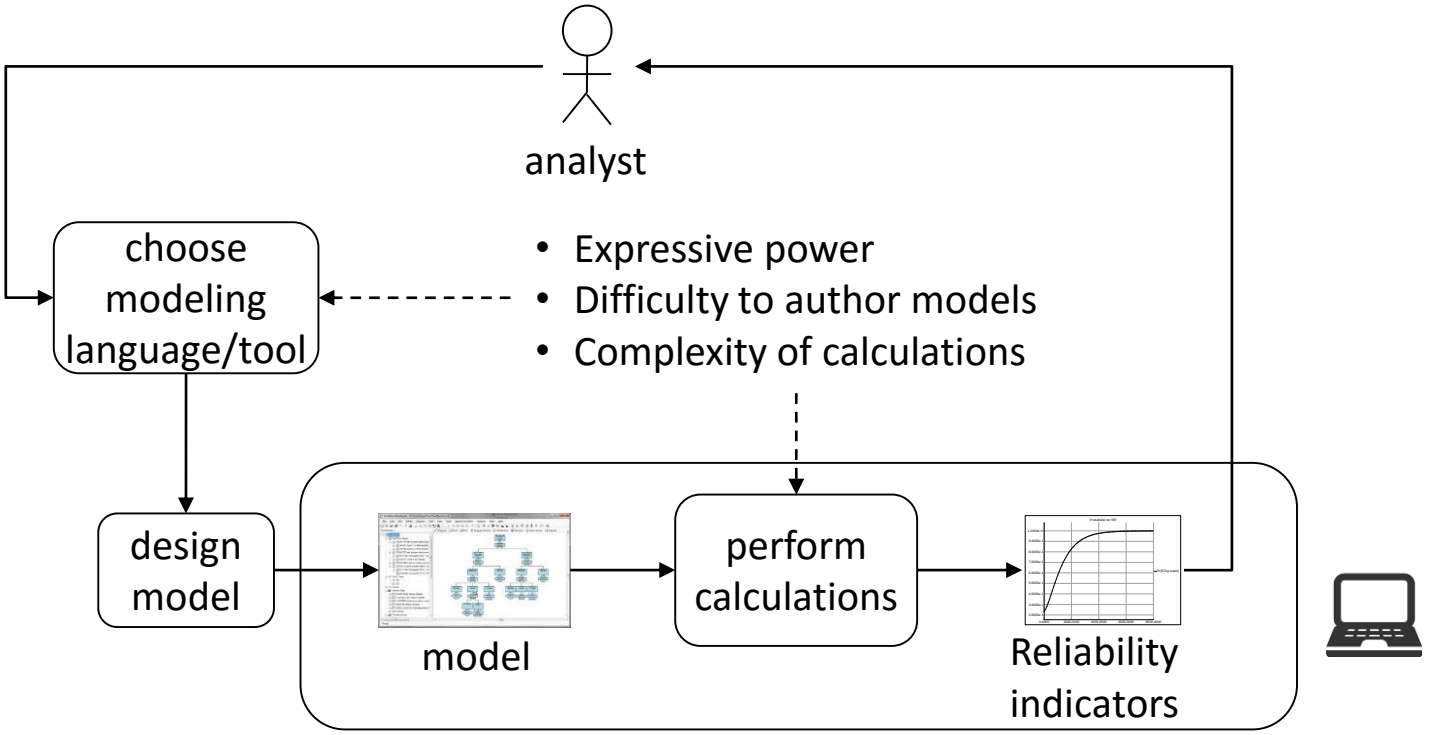
Boolean Formalisms
- Fault Trees
- Event Trees
- Reliability Block Diagrams

Transitions Systems
- Markov Chains
- Dynamic Fault Trees
- Stochastic Petri Nets
- …

Universal Languages
- Agent Based Models
- Matlab
- Java/C++
- …



Tradeoffs



Calculations in reliability assessment are **provably** extremely **resource consuming**.

Consequence 1: Tools perform **approximate calculations**.

Consequence 2: Models always result from a **tradeoff** between the **accuracy of the description** and the **ability to perform calculations** and to **validate results**

Uncertainties

epistemic uncertainty

aleatory uncertainty

analyze

draw conclusions

Herbert Simon
(1916-2001)
Bounded Rationality

analyst

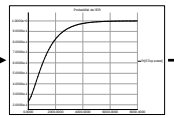
computational uncertainty

choose modeling language/tool

design model

model

perform approximate calculations



Reliability indicators

Agenda

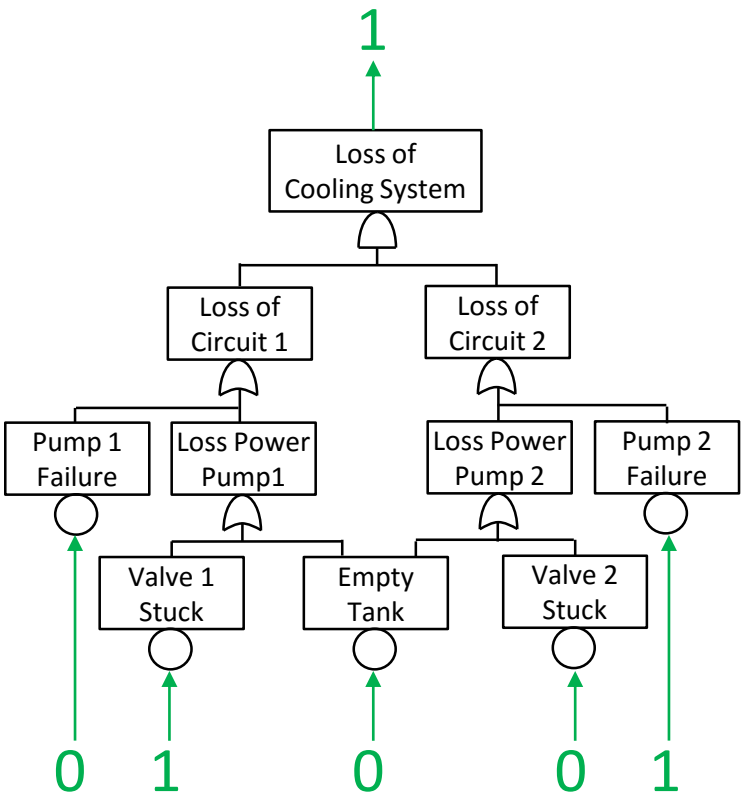
Part 1. The reliability assessment process

Part 2. Fundamental complexity results (the bad news)

Part 3. Model-Based Systems Engineering (the good news)

Wrap-Up & Challenges

State Spaces



The **state space** described by a fault tree with **n basic events** contains **2^n scenarios**.

There is an **exponential blow-up** of its size w.r.t. the size of the model.

The situation gets indeed even worse for more expressive formalisms.

A small exercise…

Take a sheet of paper. Fold it. Fold it again. And again.

Each time you fold it, it gets twice thicker.

?

Question: how many times do you need to fold the sheet to reach the moon?

Solution

A packet of 500 sheets is about 5 cm thick.

Let us start:

- After 10 folding, we get a $2^{10} \approx 1.000$ sheets (10 cm) stack
- After 20 folding, we get a 2^{10} x 10 cm $\approx$ 1.000 x 10 cm = 100 m stack
- After 30 folding, we get a 2^{10} x 100 m $\approx$ 1.000 x 100 m =100 km stack
- After 40 folding, we get a 2^{10} x 100 m $\approx$ 1.000 x 100 km = 100.000 km stack

World record at MIT 13 folding!

- The **distance from the earth to the moon varies between** 356.375 km and 406.720 km. On average it is 384.400 km.
- In **42 folding**, we get a stack of 2^2 x 100.000 km = 400.000 km which is way enough to go to the moon…
- … and if we fold it once more, we can come back

Computation Times

Computation times to evaluate all configurations with k components out of n on a computer that evaluates 10^6 components per second.

n/k	2	3	4	5	6	7	10
50	0.001”	0.02”	0.2”	2”	20”	2h	2h50’
100	0.005”	0.2”	4”	1’20”	21’	4h	6 months
200	0.02”	1”	1’	40’	21h	26 days	7 centuries
1000	0.5”	2’50”	10h30’	6 days	40 years	60 centuries	$8\ 10^7$ centuries

An Old and Fascinating Story

19th century: discovery of various mathematical paradoxes.

Beginning of 20th century: David Hilbert's program:

Formalization of all mathematics.

David Hilbert
(1862-1943)

1931: Kurt Goedel's incompleteness theorem:

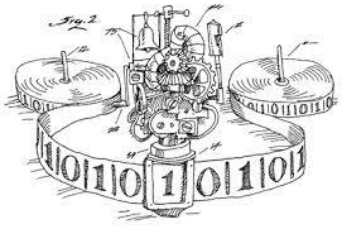
There is no algorithm to decide the truth (or provability) of statements in any consistent extension of Peano arithmetic (set theory).

Kurt Goedel
(1906-1978)

1936: Alan Turing's seminal articles about calculation

There exist universal calculators.

There is no algorithm to decide whether a calculator halts on a given input.

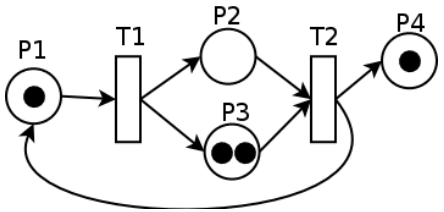


Turing machine

Alan Turing
(1912-1954)

Consequences for the Validation of Models

The **marking** of a **Petri net** is a function that associates with each place the number of tokens in that place.



A Petri net

The **reachability problem** consists in determining whether a given marking M is reachable from a given initial marking M_0.

Regular Petri Nets	Petri nets with inhibitor arcs
Decidable (Mayr, 1981)	**Undecidable** (see Esparza & Nielsen 1995)

And also:

- Program halting problem: **undecidable**
- Program equivalence problem: **undecidable**
- …

From Decidability down to Complexity

Turing machines are not only a model of calculator, they are also very helpful to characterize the **complexity of a calculation**.

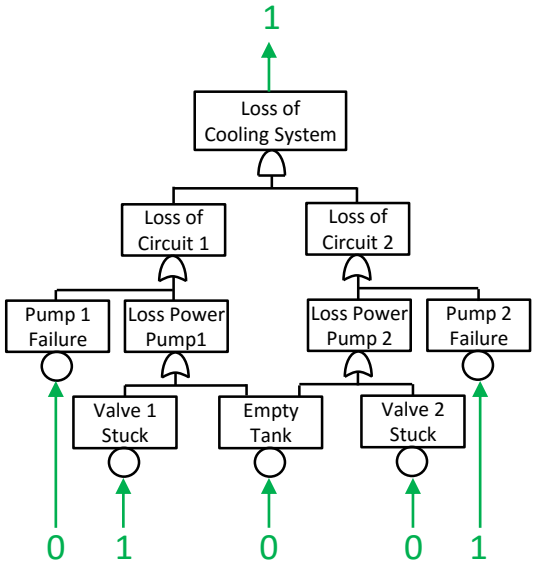
The **complexity of an algorithm** is in **f(n)** if the maximum number of steps of that algorithm on an input of size n is f(n).

The **complexity of a problem** is in **f(n)** if the complexity of the fastest algorithm to solve that problem is in f(n).

Complexity of problems seen by mathematicians (and computer scientists):

Easy	Hard
Complexity in g(n) where g is a polynomial	Complexity in g(n) where g is not bounded by any polynomial

The SAT problem



SATisfiability **Problem**:

Given a Boolean formula F (e.g. a Fault Tree with a possibly negated events) over variables (basic events) V_1,…, V_n, is there a valuation of the Vi's that satisfies F (realizes the top event of the tree).

Three fundamental characteristics:

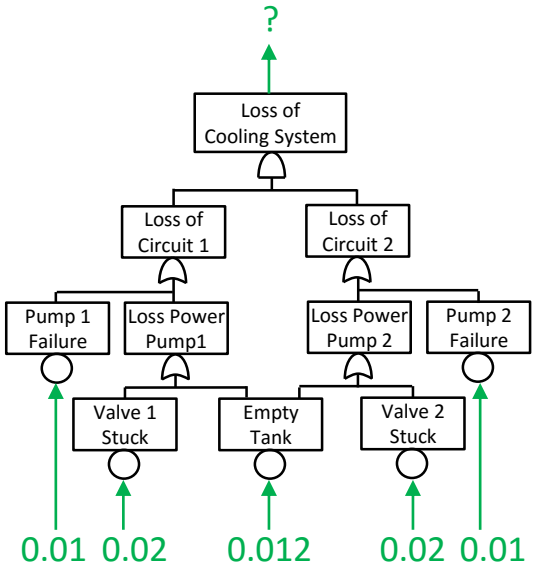
- There are **2^n candidate valuations**.
- It is **easy to check** (linear time) whether a valuation satisfies F or not.
- If the function is **monotone** function, it suffices to check V_1=1,…, V_n=1

SAT is representative of a large class of problems having the same characteristics: **Non-deterministic Polynomial** decision problems (NP).
It has been shown **NP-complete**, i.e. as hard and as easy of all of the problems of this class (Cook, 1971).

One million US dollars question: **P $\overset{?}{=}$ NP** (believed to be false)

Stephen Cook (1939-…)

The RELIABILITY problem



RELIABILITY Problem:

Given a Boolean formula F (e.g. a Fault Tree with a possibly negated events) over variables (basic events) $V_1,\ldots, V_n$, and probability $p(V_i)$ for each V_i, what is the probability of F?

Two fundamental characteristics:

- At least as hard as **#SAT**, the problem of counting the number of solutions of F.
- Same complexity whether F is monotone or not.

RELIABILITY and #SAT are representative of a large class of problems having the same characteristics: **#P** problems.
They have been shown **#P-complete**, i.e. as hard and as easy of all of the problems of this class (Valiant, 1979).

Leslie Valiant
(1949-…)

戸田 誠之助
(1959-…)

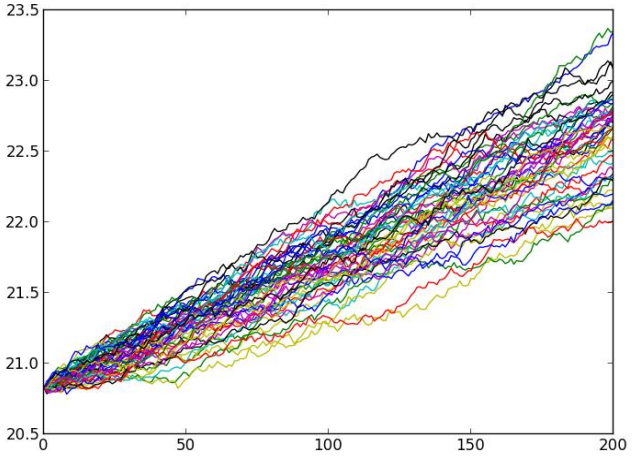
Strongly believed to be **intractable** (Toda's theorem, 1991)

What about Approximations?

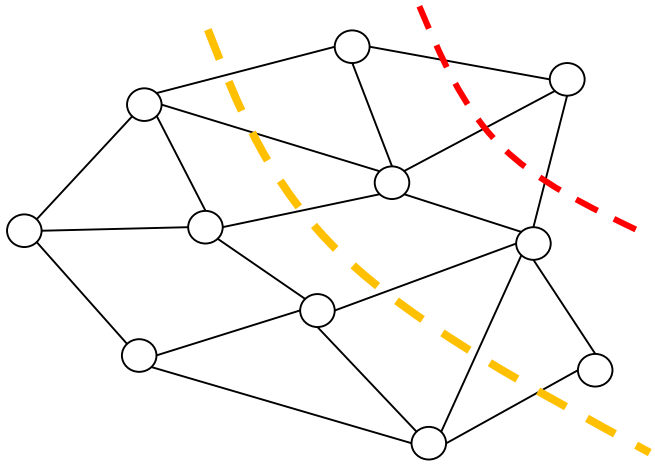
Valiant and Toda's results apply to **exact probability calculations** but there are so many approximations in models (due to epistemic and aleatory uncertainties) that we can **tolerate approximations in calculations** as long as they are **reasonably conservative**.

Two approaches to get approximations:

Sampling Approach



Cutoff Approach



Sampling Approach

Principle: Monte-Carlo simulation:

1) Sample variable valuations according to variable probabilities.
2) Approximate p(F) as the number of times s the drawn valuation is a solution of F divided by the size n of the sample (empirical mean $\mu = s/n$ and standard deviation σ).

According to the **Weak Law of Large Numbers**, μ **converges in probability** to p(F).

Confidence interval at x%: $\left(\mu - z_x \frac{\sigma}{\sqrt{n}},\ \mu + z_x \frac{\sigma}{\sqrt{n}}\right)$

x	Z_x
99%	2.576
98%	2.326
95%	1.96
90%	1.645

Issues:

Cost:

When p(F) is small the size of the sample must very large to get accurate enough results.

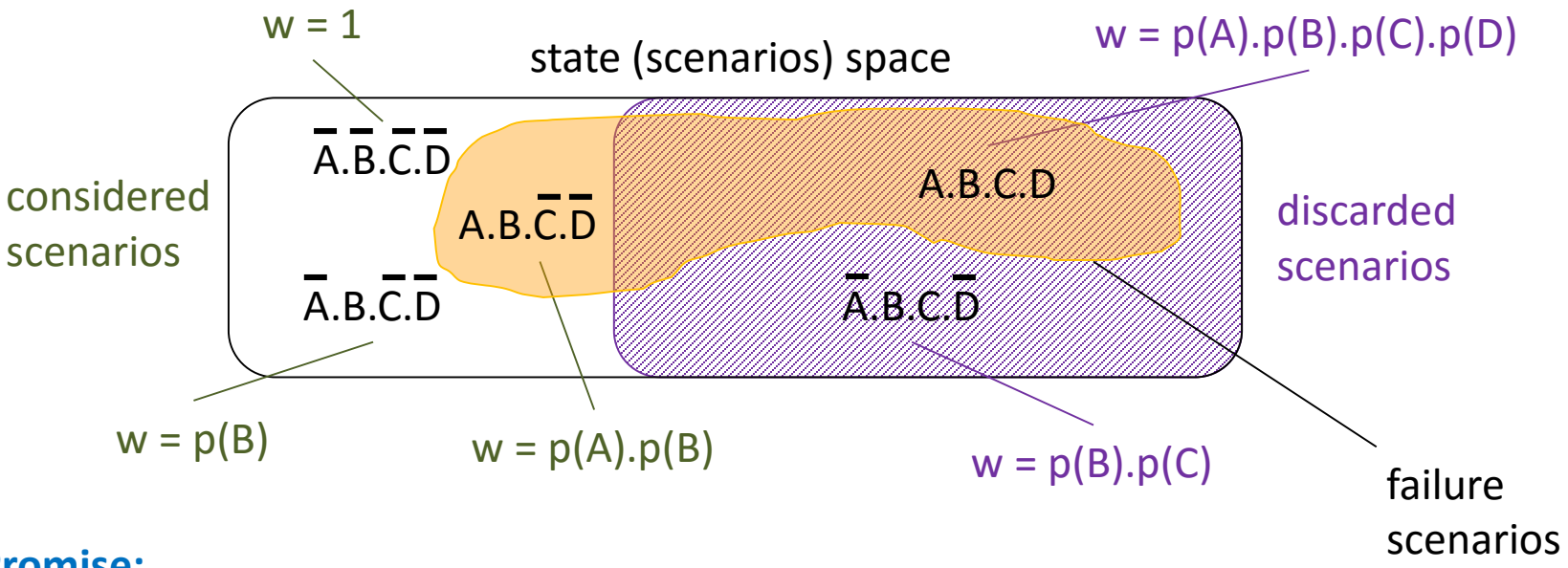
Validation:

Obtaining a raw number is of no help to validate the model (and the assessment tool).

Cutoff Approach

Principle:

1) Associate a **weight** to each potential scenario.
   E.g w(σ) = product of probabilities of basic events showing up positively in the scenario σ
2) **Discard scenarios** with a weight lower than a predefined **threshold**

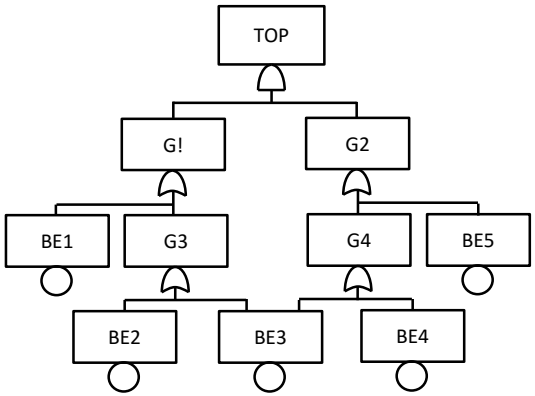


Promise:

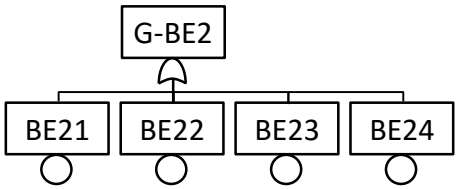
Maintain the set of scenarios to look at to a **manageable size**

The Refinement Paradox

abstract model



refined model



Failure scenarios

BE2.BE4	w > threshold
BE21.BE22.BE4	w < threshold
BE21.BE23.BE4	w < threshold
BE21.BE24.BE4	w < threshold
BE22.BE23.BE4	w < threshold
BE22.BE24.BE4	w < threshold
BE23.BE24.BE4	w < threshold

The more refined (accurate) the model, the lower (less accurate) the measure of risk

Comparison of the Two Approaches

	Sampling approach	Cutoff Approach
Resource consumption	✗	✗
Versatility	✓	✗
Rare events	✗	✓
Refinement	✓	✗
Validation	✗	✓

Agenda

Part 1. The reliability assessment process

Part 2. Fundamental complexity results (the bad news)

Part 3. Model-Based Systems Engineering (the good news)

Wrap-Up & Challenges

Categories of Modeling Languages

Boolean Formalisms
- Fault Trees
- Event Trees
- Reliability Block Diagrams

Transitions Systems
- Markov Chains
- Dynamic Fault Trees
- Stochastic Petri Nets
- …

Universal Languages
- Agent Based Models
- Matlab
- Java/C++
- …

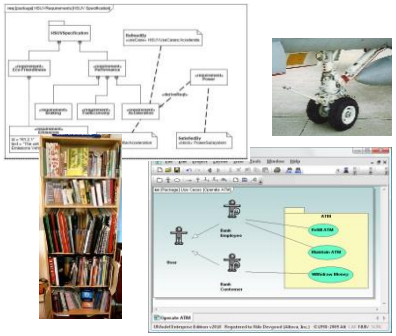
- Is there a generic/unified **modeling framework** to assess reliability of complex systems?

  Guarded Transitions Systems as implemented in AltaRica

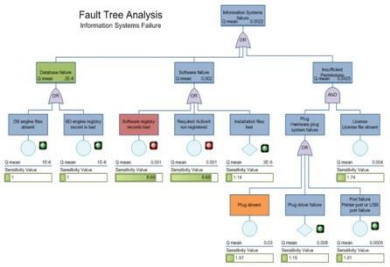
- What are the available **algorithmic tools** to assess reliability of complex systems?

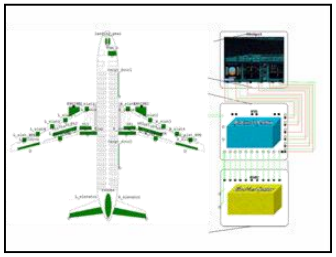
Model-Based Reliability Assessment

Systems Specifications



Models





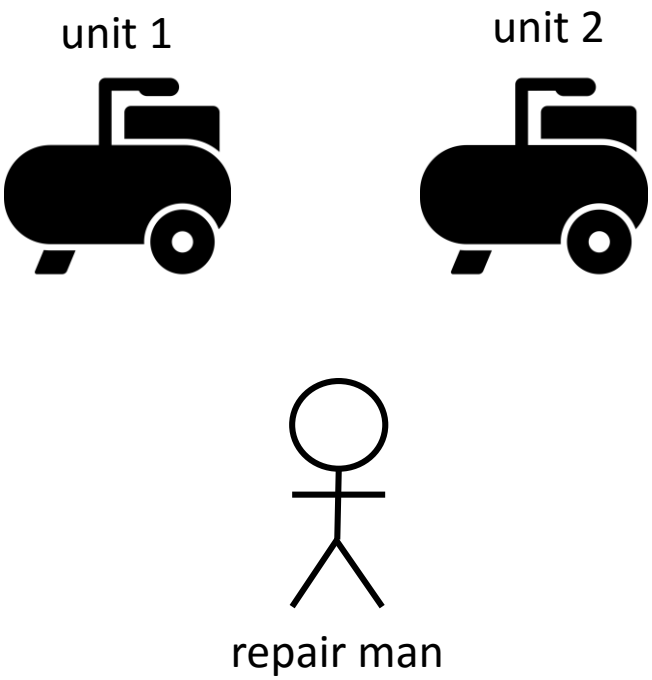
```
class HydraulicPump
  Boolean working (init = false);
  event failure (delay = exponential(lambda));
 transition
    failure: working -> working := false;
end
```

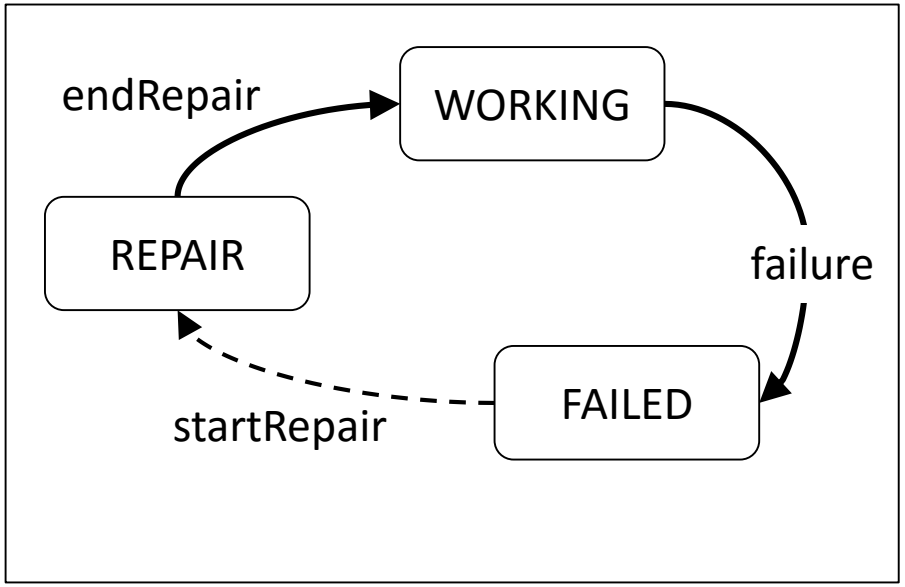
AltaRica

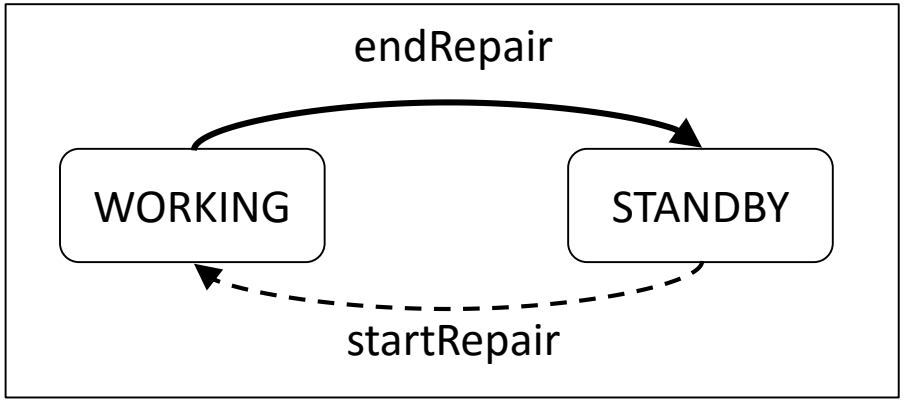
Promise:

- Ability to **animate/simulate** models: **model validation**, **discussions with stakeholders**;
- One model, several safety goals: to ease **versioning**, **configuration**;
- One model, several assessment tools: **versatility**, **quality-assurance**;
- Fine grain analyses: to **avoid over-pessimism**.

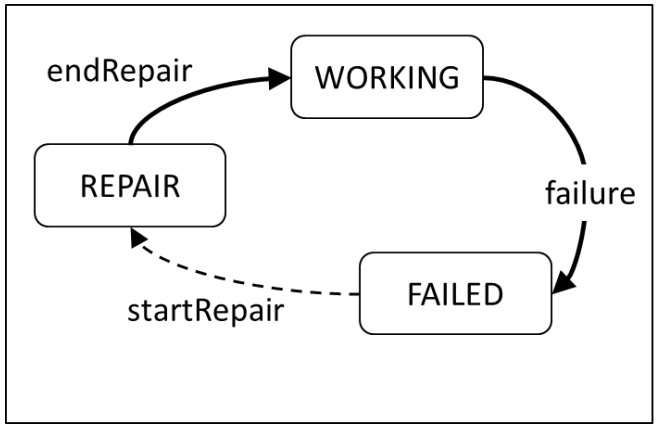
State Automata





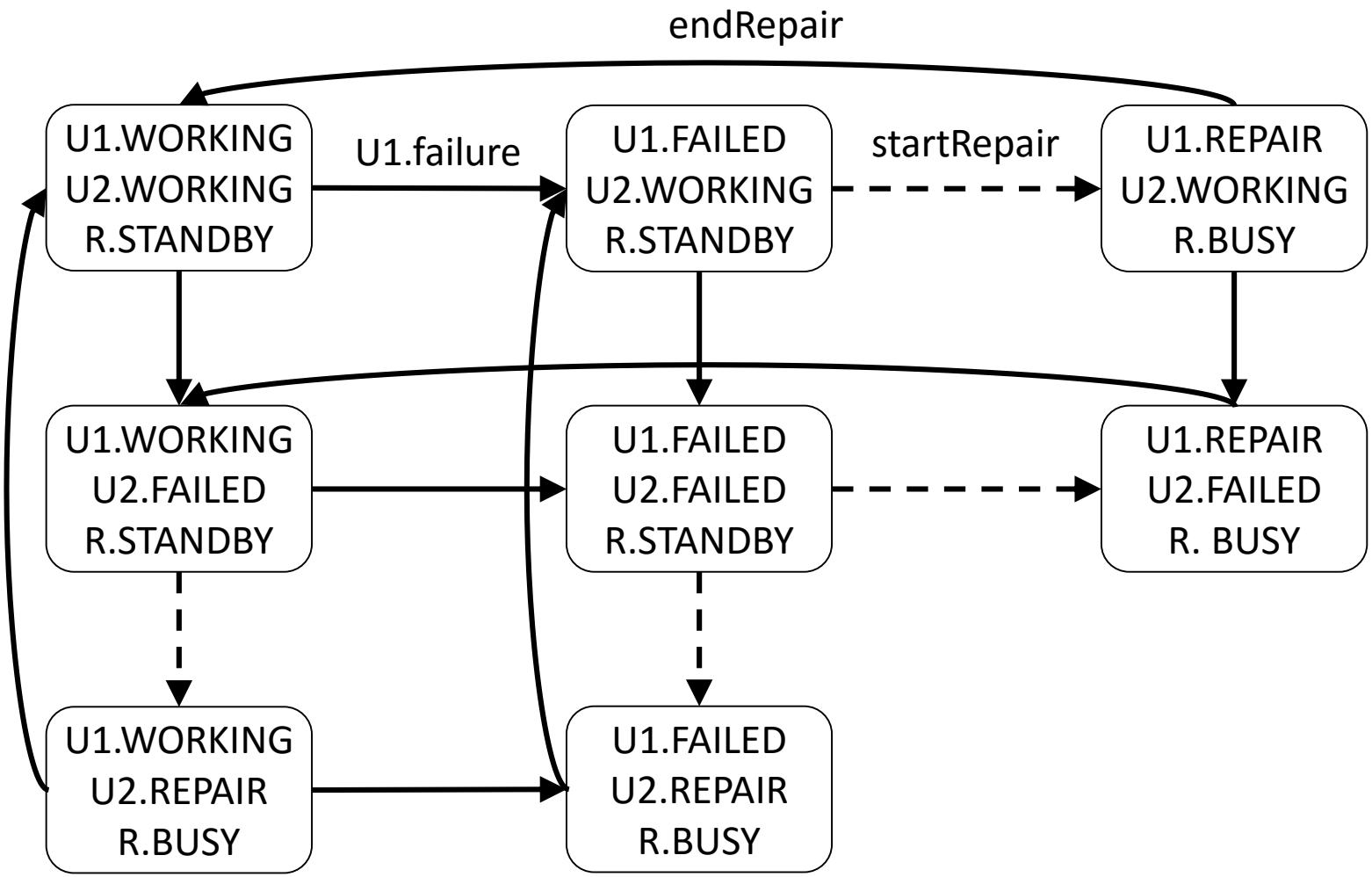


Guarded Transitions Systems

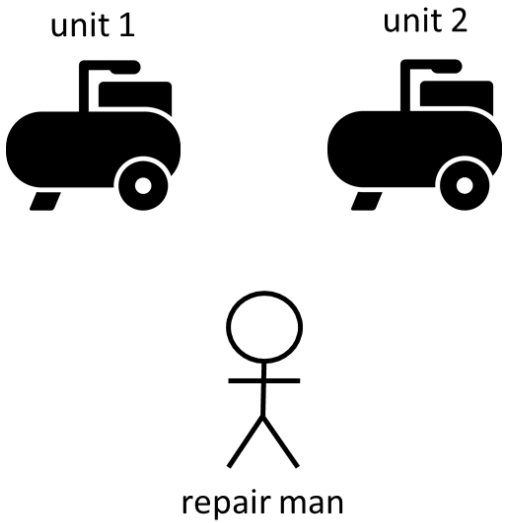



```
domain UnitState {WORKING, FAILED, REPAIR}

block Unit
    UnitState state (init = WORKING);
    event failure;
    event startRepair;
    event endRepair;
    transition
        failure: state==WORKING -> state:=FAILED;
        startRepair: state==FAILED -> state:=REPAIR;
        endRepair: state==REPAIR -> state:=WORKING;
end
```

Synchronized Products

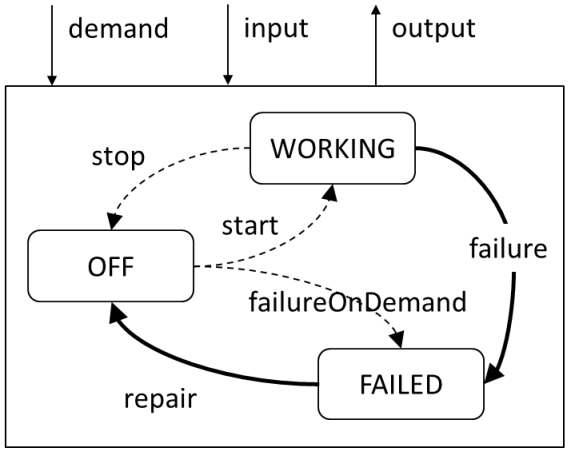


Composition



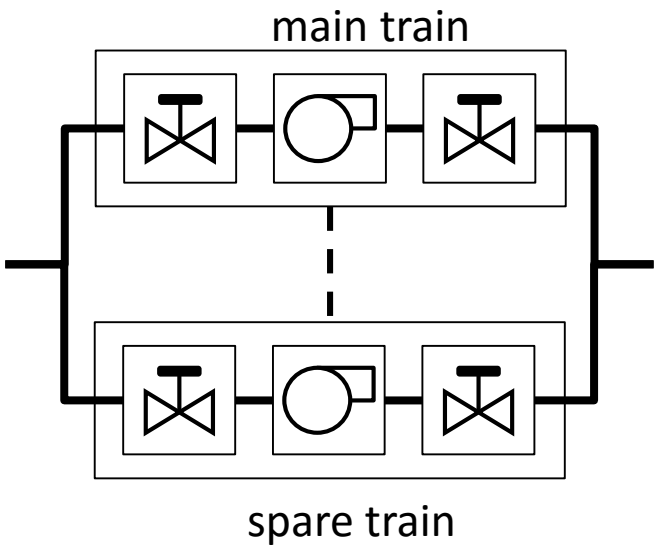

```
domain UnitState {WORKING, FAILED, REPAIR}
domain RepairManState {STANDBY, BUSY}


block System
    block U1 … end
    block U2 … end
    block R … end
    event startRepair;
    event endRepair;
    transition
        startRepair: U1.startRepair & R.startRepair;
        startRepair: U2.startRepair & R.startRepair;
        endRepair: U1.endRepair & R.endRepair;
        endRepair: U2.endRepair & R.endRepair;
end
```

Flows




```
domain PumpState {WORKING, FAILED, REPAIR}

block Pump
    PumpState state (init = OFF);
    Boolean demand (reset = false);
    Boolean input (reset = false);
    Boolean ouput (reset = false);
    event start;
    …
    transition
        start: state==OFF and demand -> state:=WORKING;
        …
    assertion
        output := state==WORKING and input;
end
```

Composition



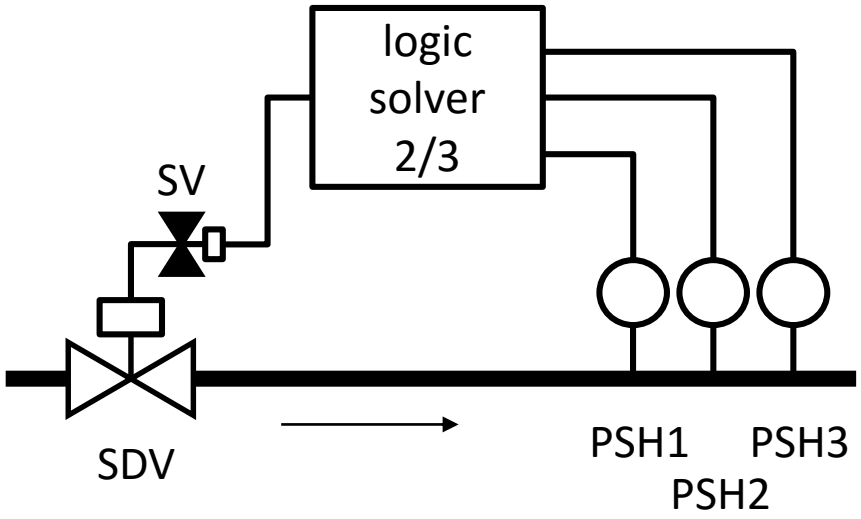

```
block System
    block MT
        block V1 … end
        block V2 … end
        block P1 … end
    end
    …
    assertion
        MT.P1.input := MT.V1.output;
        MT.V2.input := MT2.P1.output;
        …
        ST.demand := not MT.output;
        …
end
```

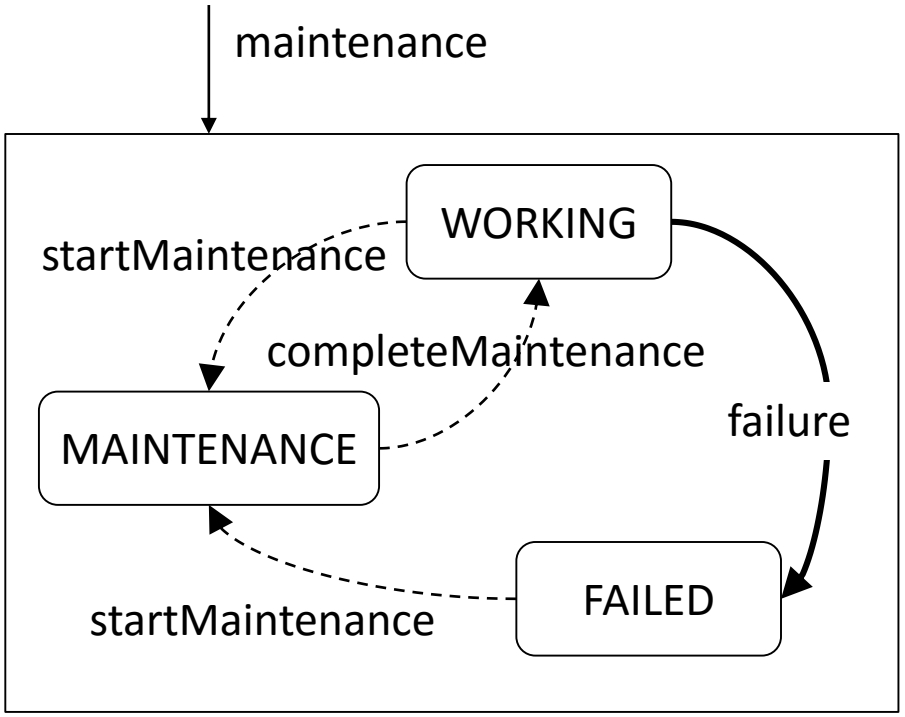
Classes

It is often convenient to have a common definition for different elements of a model (e.g. the four wheels of a car). A **class** is a separately defined, reusable (on the shelf) **block**.

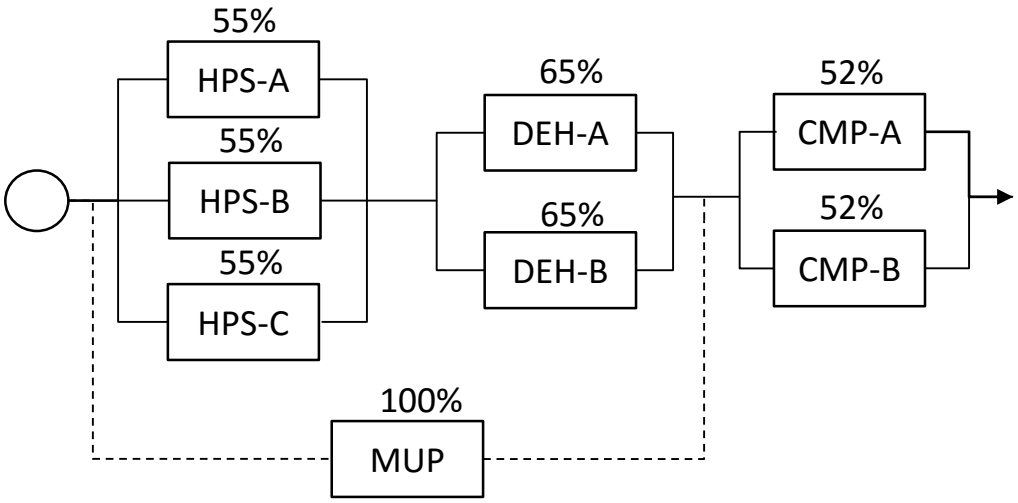
```
class Pump
      // elements of the pump
end

class Valve
     // elements of the valve
end

block System
     Pump P;
     Valve V;
     /* elements to connect
        the pump P and the valve  V*/
end
```

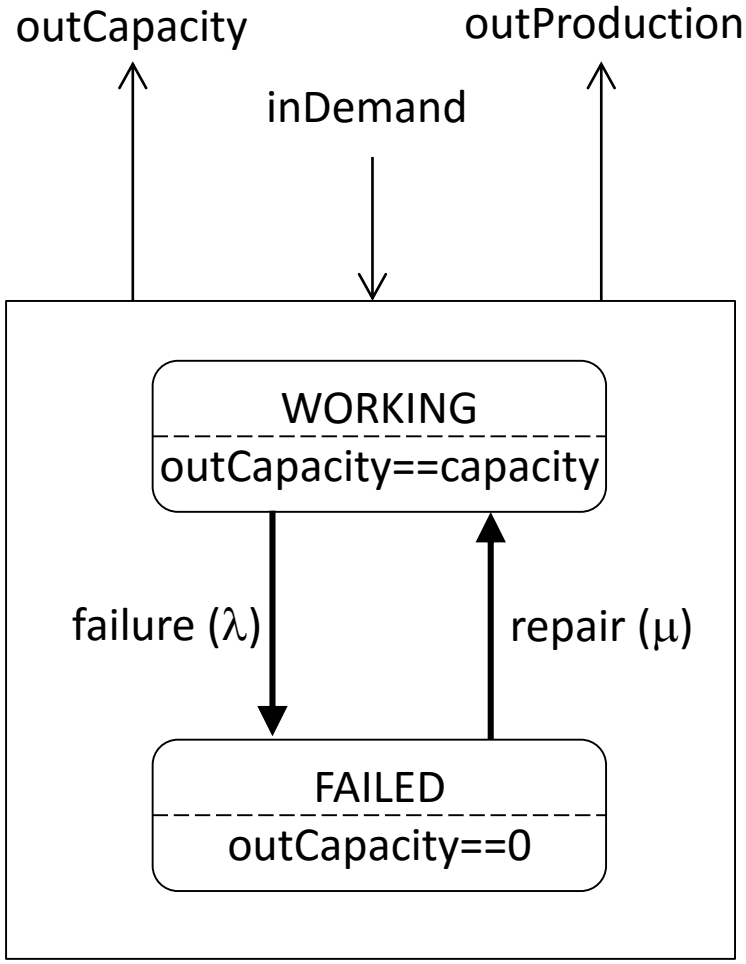
Maintenance Strategies





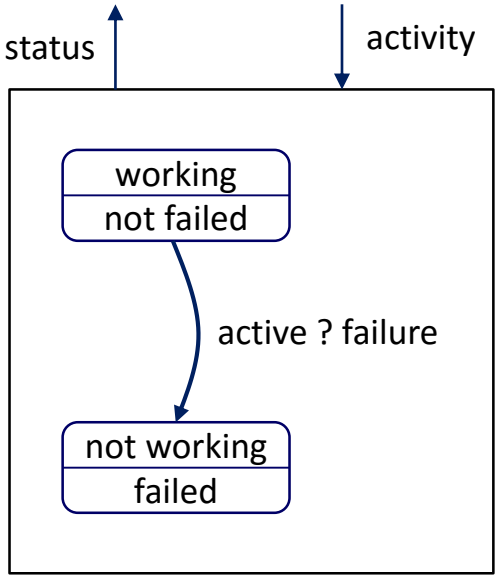
Production Levels



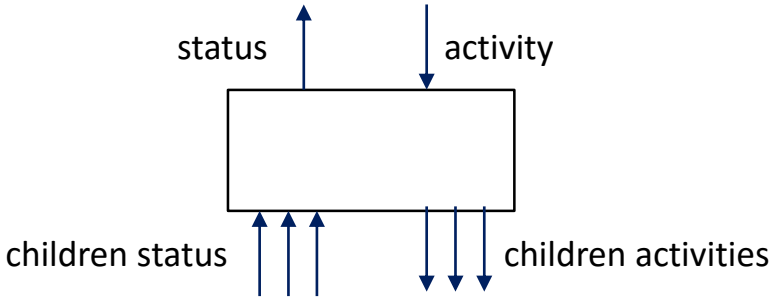


AltaRica versus Dynamic Fault Trees

Basic Event



Gates

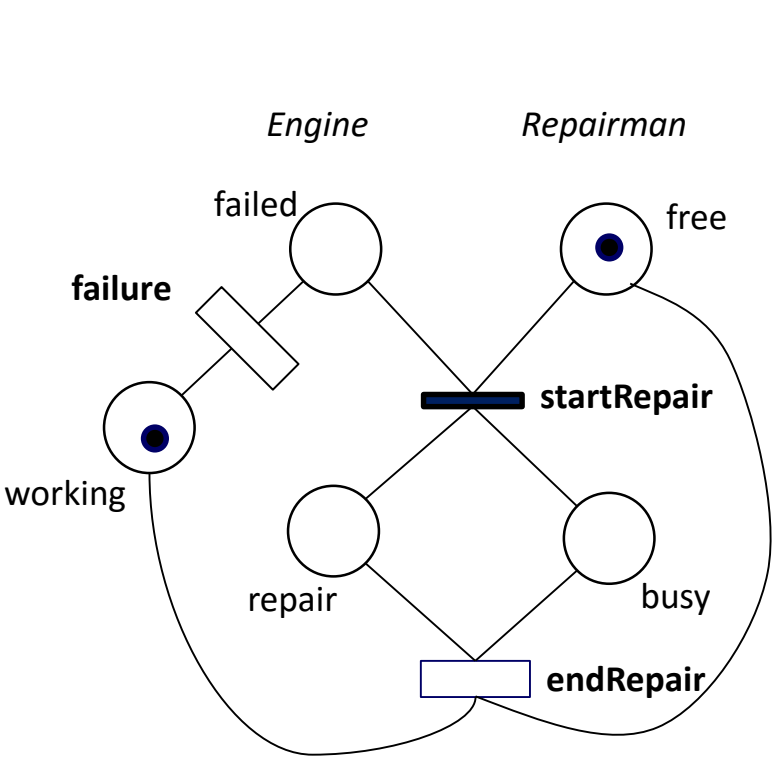


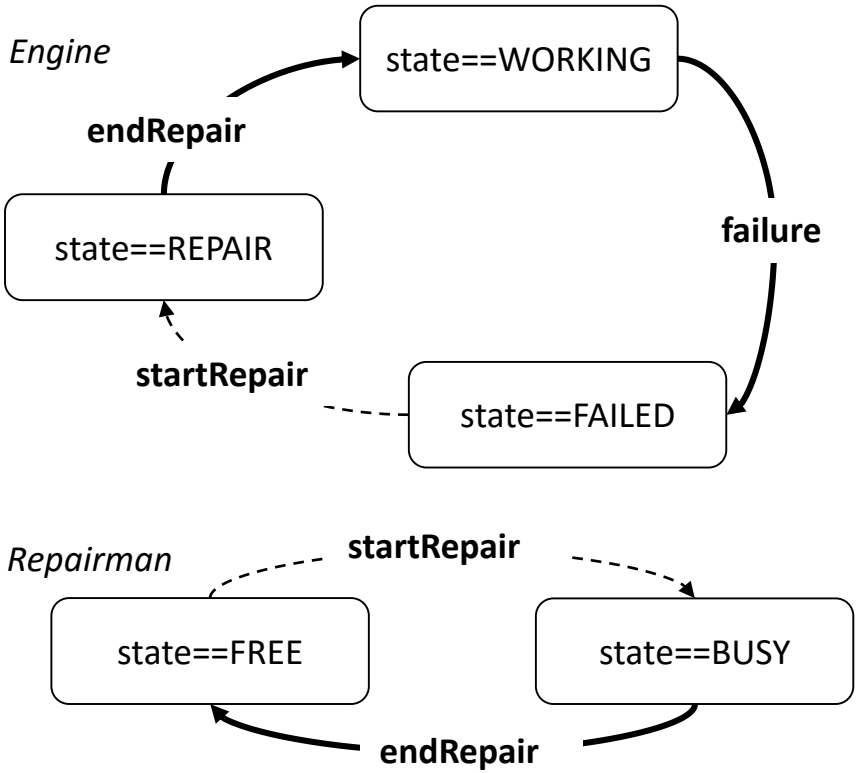
Idea: Basic Events and Gates

- calculate their status (working or failed) bottom-up;
- are activated top-down (in regular Fault Trees, basic events and gates are always active).

AltaRica generalizes (at no cost) Dynamic Fault Trees

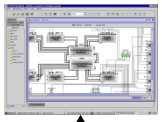
AltaRica versus Stochastic Petri Nets





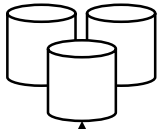
AltaRica generalizes (at no cost) Stochastic Petri Nets (and various extensions of).

The AltaRica 3.0 Project



Graphical User Interfaces
for model authoring and simulation

Libraries of reuseable
modeling components
and patterns

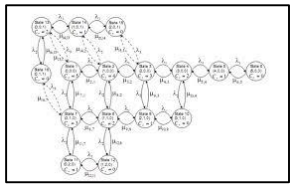


AltaRica 3.0

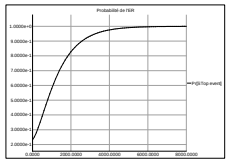
```
class Pump
...
end
```

compiler to
Fault Trees

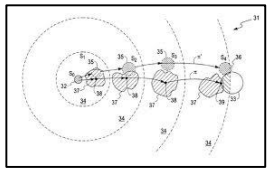
compiler to
Markov Chains



stochastic
simulator



model checker



A Central Thesis of Model-Based Systems Engineering

Behaviors + Structures = Models*

Meaning and practical consequences:

- Any modeling language is the combination of a **mathematical framework** to describe the behavior of the system under study and a **structuring paradigm** to organize the model.
- The choice of the **appropriate mathematical framework** for a model depends on the **characteristics of the system** one wants to study.
- **Structuring paradigms** are to a very large extent **independent** of the chosen mathematical framework. They can be studied on their own.

(*) In reference to Wirth's seminal book "Algorithms + Data Structures = Programs"

Niklaus Wirth
(1934-…)

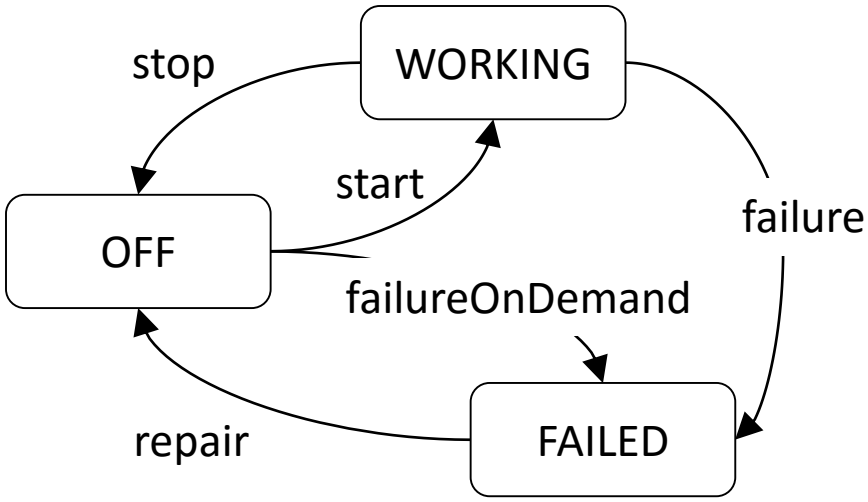
Expected Properties of a Mathematical Framework (1)

1. Event-Centered Descriptions

The system can be in a number of **states** and changes of states only under the occurrence of **events**.

The state of the system is described by means of one or more variables.

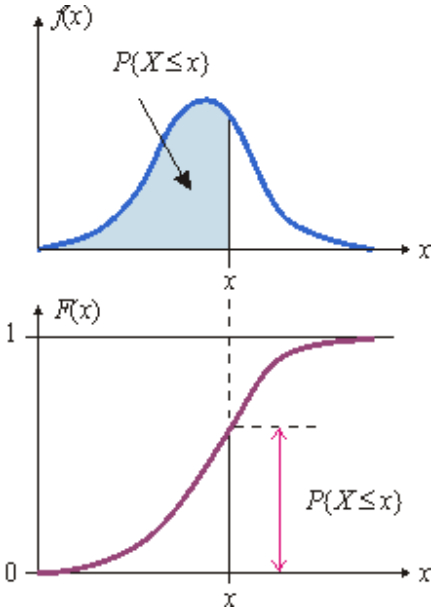
Rewards can be associated with each state.

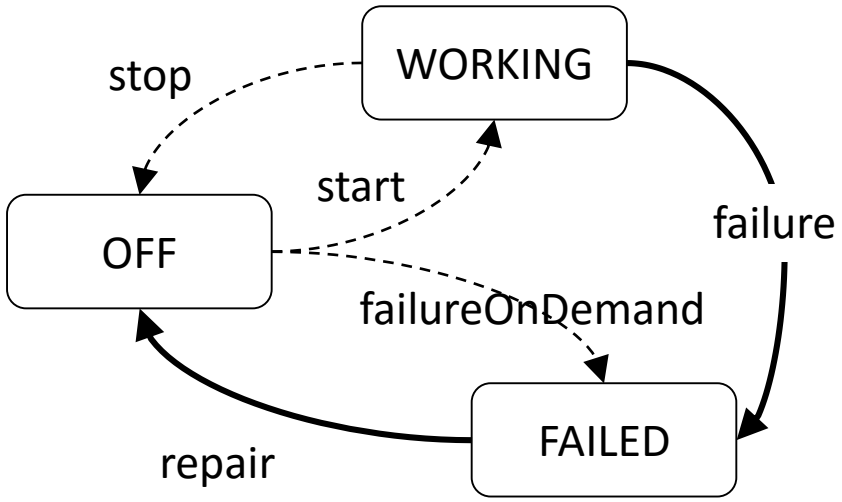


Expected Properties of a Mathematical Framework (2)

1. **Event-Centered Descriptions**
2. **Stochastic Descriptions**

Events are associated with **deterministic or stochastic delays** and/or **probabilities (weights)**.



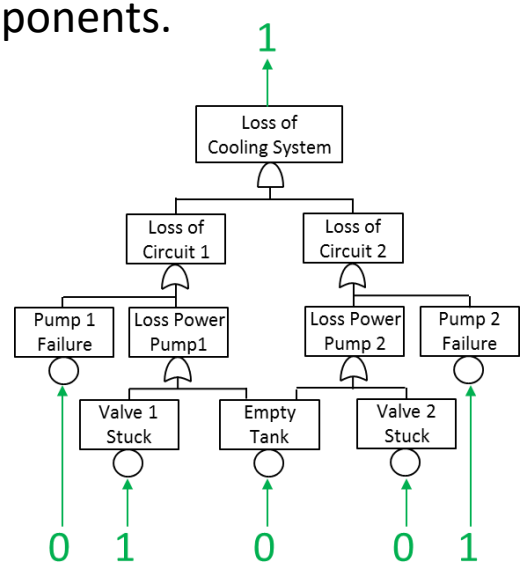


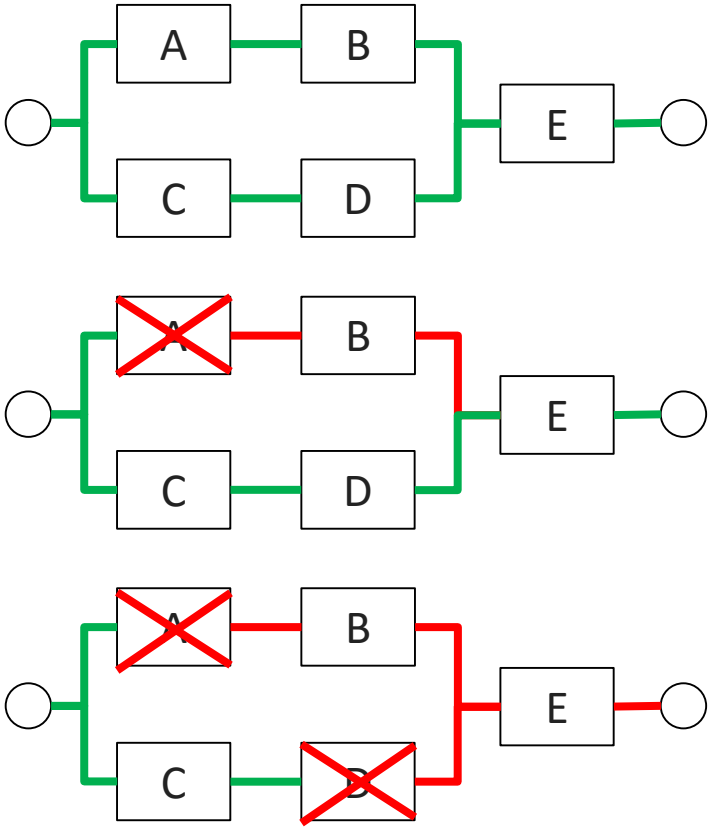
event	rate	probability
failure	λ	
repair	μ	
start		$1-\gamma$
failureOnDemand		γ
stop		1

Expected Properties of a Mathematical Framework (3)

1. **Event-Centered Descriptions**
2. **Stochastic Descriptions**
3. **Flow Propagation**

After each transition firing, we need a mechanism to propagate the change of state through the network of components.



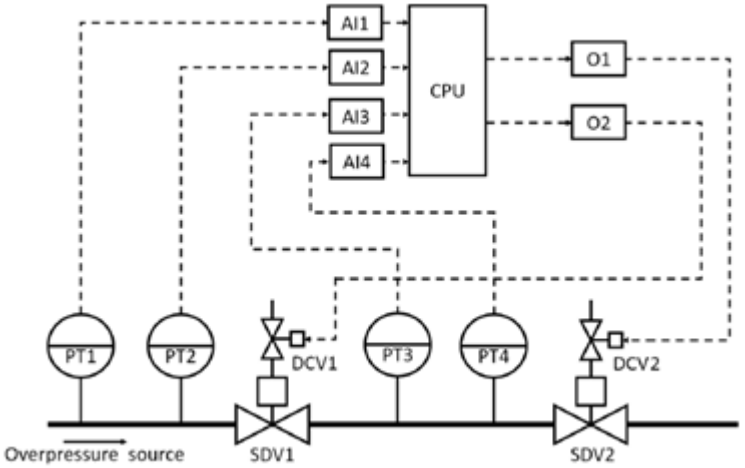


Flow circulating through the network can be of different types

Expected Properties of a Mathematical Framework (4)

1. **Event-Centered Descriptions**
2. **Stochastic Descriptions**
3. **Flow Propagation**
4. **Compositional & Implicit Descriptions**

The model for the system should be obtained by **composing** smaller models of subsystems and components.
This means that the model should be an **implicit** representation of the state space (as opposed to explicit representations as in Markov chains)



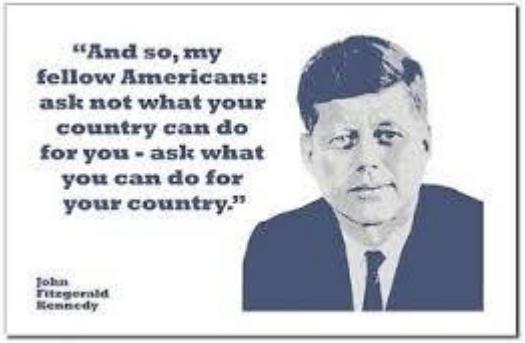
Expected Properties of a Mathematical Framework (5)

1. **Event-Centered Descriptions**
2. **Stochastic Descriptions**
3. **Flow Propagation**
4. **Compositional & Implicit Descriptions**
5. **Algorithm Friendliness**

The number of candidate assessment techniques for transitions systems is rather limited:

- **Compilation into fault trees** then fault tree assessment technique
- **Compilation into Markov chains** (possibly multi-phase Markov chains with rewards) then use of numerical simulation techniques.
- **Monte-Carlo simulation**.
- **Model-checking** techniques.

And so, my fellow ESREL delegates:
ask not what algorithms can do for your modeling language…
ask you what your modeling language can do for algorithms.



John Fitzgerald Kennedy
(1917-1963)

Expected Properties of a Structuring Paradigm (1)

Models of complex systems cannot be simple. They need to be structured.

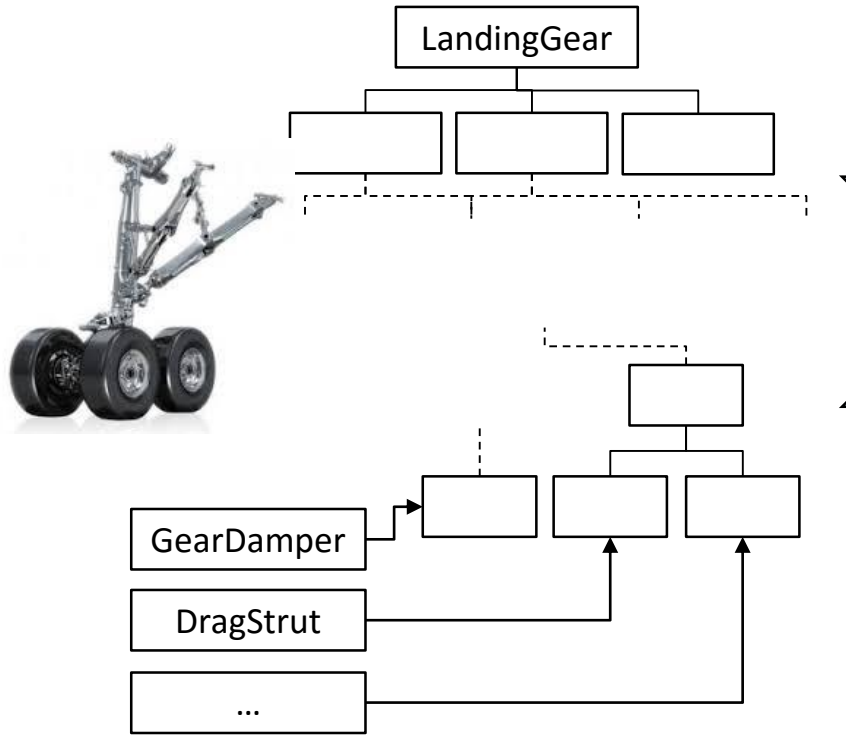
Almost all, if not all, of structural relationships amongst the components/functions of a system are captured by a small sets of relations:

- "**is-part-of**" (composition),
- "**is-a**" (inheritance),
- "**uses**" (aggregation).

These relations are at the very core of **systems architecture** and **models structure**. They can represented graphically in various ways, depending on the needs.

Expected Properties of a Structuring Paradigm (2)

Any high level modeling formalism should provide constructs to implement these relations as well as a mechanism to defined **on-the-shelf modeling component**.



- Top-down model design
- System level
- Reuse of modeling patterns
- Prototype-Orientation

system architecture

- Bottom-up model design
- Component level
- Reuse of modeling components
- Object-Orientation

Multiphysics simulation

Agenda

Part 1. The reliability assessment process

Part 2. Fundamental complexity results (the bad news)

Part 3. Model-Based Systems Engineering (the good news)

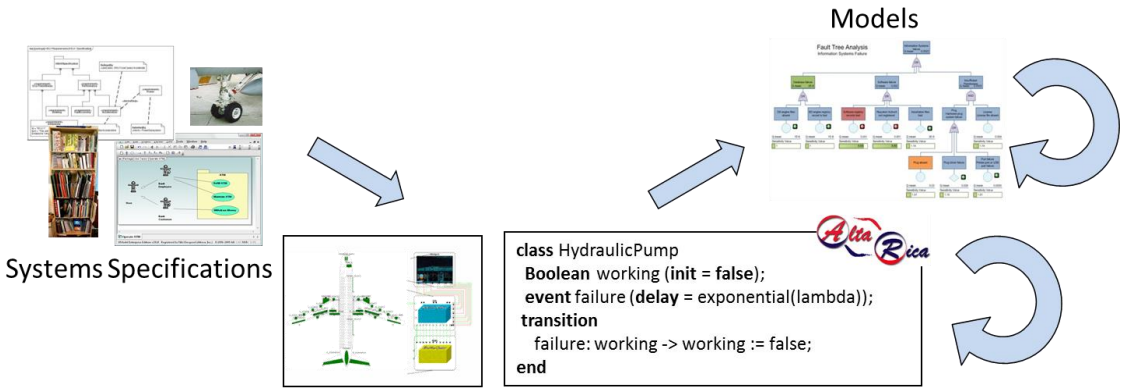
Wrap-Up & Challenges

Issues

Reliability assessment of complex system raises a number of challenging issues:

How to design model efficiently?
How to reuse parts/components of models?
How to validate models?
How to have a quality assurance on the outputs of the modeling process?
How to maintain models throughout the life-cycle of systems?
How to better integrate reliability engineering with other disciplines?
…

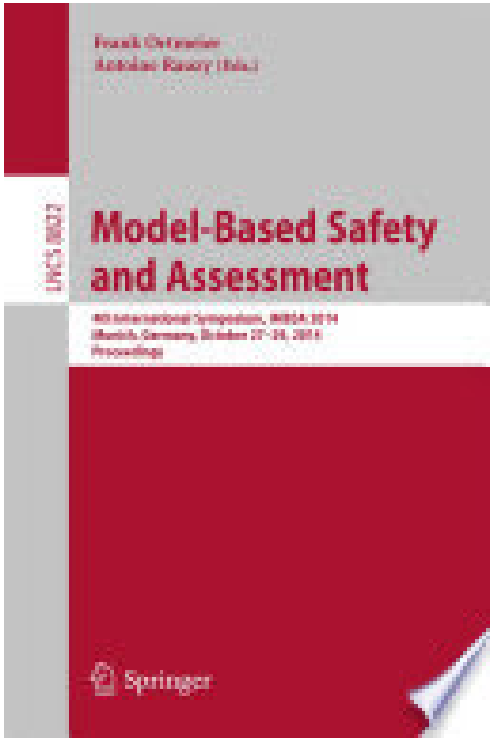
Model-Based Systems Engineering



AltaRica 3.0 = Guarded Transitions Systems + S2ML

4th International Conference
June 2014, Munich, Germany

5th International Conference
September 2017, Trento, Italy



Uncertainties

epistemic uncertainty

aleatory uncertainty

analyze

draw conclusions

Herbert Simon
(1916-2001)
Bounded Rationality

analyst

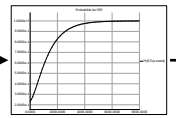
computational uncertainty

choose modeling language/tool

design model

model

perform approximate calculations



Reliability indicators

Challenges

New generations of technical systems are software intensive and interconnected: cyber-physical systems, systems of systems…

We are now facing a number of extremely difficult problems to deal with these new generations of systems as they are:

- **Opaque**: their states can be observed only by indirect means.
- **Reflective**: they embed models of their own behavior and environment.
- **Deformable**: their architecture changes throughout their mission.

We have to forge the concepts to face these issues.