Towards Executable Models in Systems Engineering

Antoine Rauzy (NTNU)

Agenda

- **Introduction**
- Models versus Notations
- Mathematical Frameworks
- The S2ML+X Paradigm
- Sigma and WorldLab
- Conclusion

Model-Based Systems Engineering

We entered the era of MBSE but several questions remain to open to a large extent:

- How to make the MBSE process efficient?
- What is a (good) model/modeling language/modeling environment?
- What is the role of **computerized simulation** in MBSE?

These questions are serious and require serious answers.

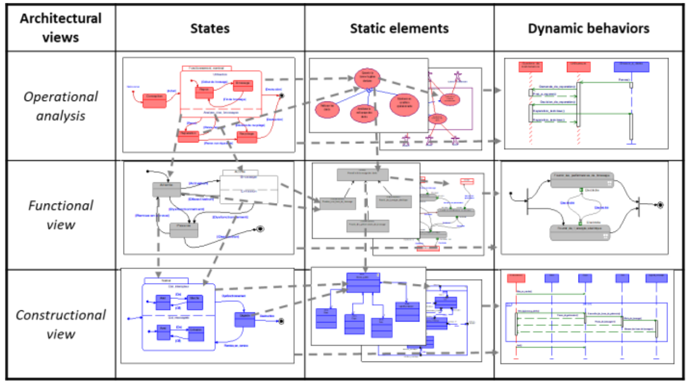
They are related to **scientific foundations** of MBSE.

Agenda

- Introduction
- **Models versus Notations**
- Mathematical Frameworks
- The S2ML+X Paradigm
- Sigma and WorldLab
- Conclusion

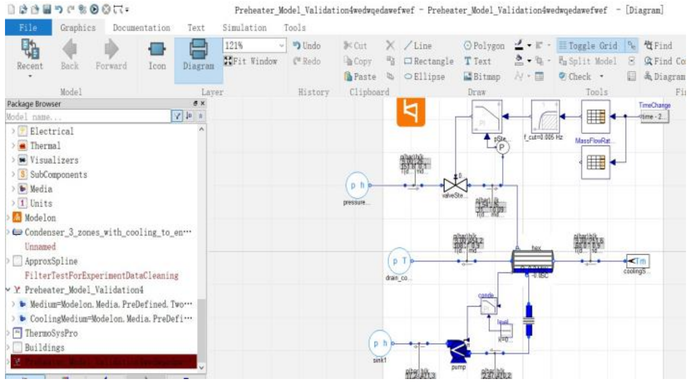
Pragmatic versus Formal Models

“Models” to communicate
amongst stakeholders



Pragmatic “models”
Standardized graphical notations

“Models” to calculate
performance indicators



Formal “models”
Differential equations, Automata

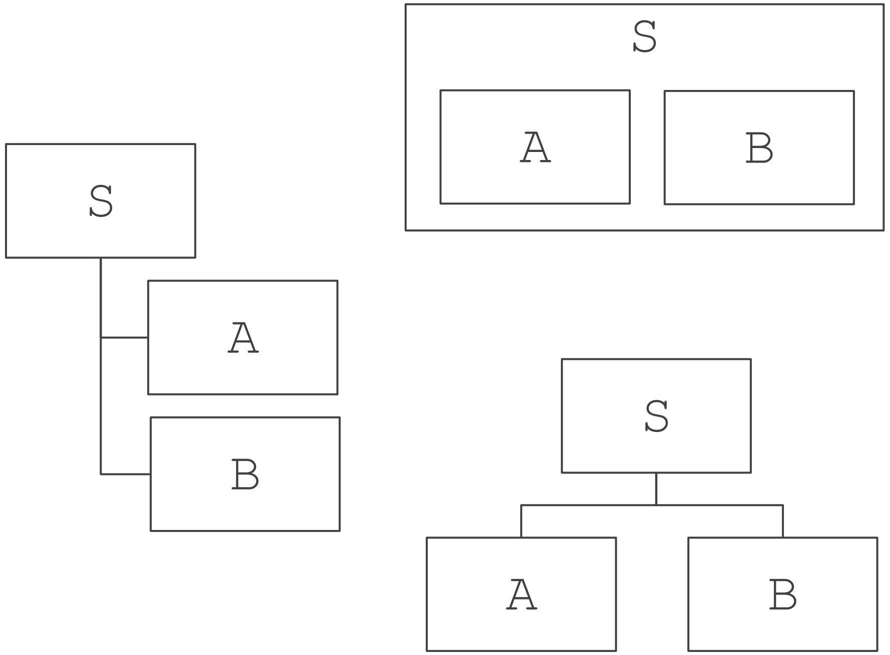
Epistemic gap

Diagrams Are Not Models

Models are **mathematical objects**

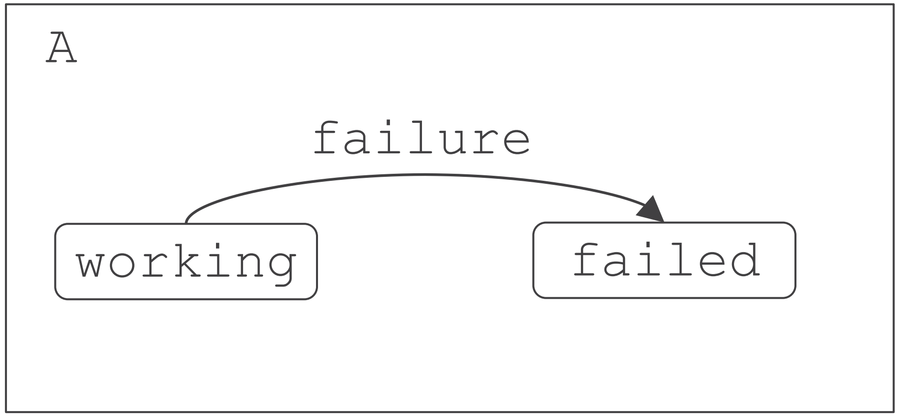
```
block S
   block A
     …
   end
   block B
     …
   end
end
```

Diagrams are (or more exactly should be) **graphical representations** of models



Formal Models Have a Syntax

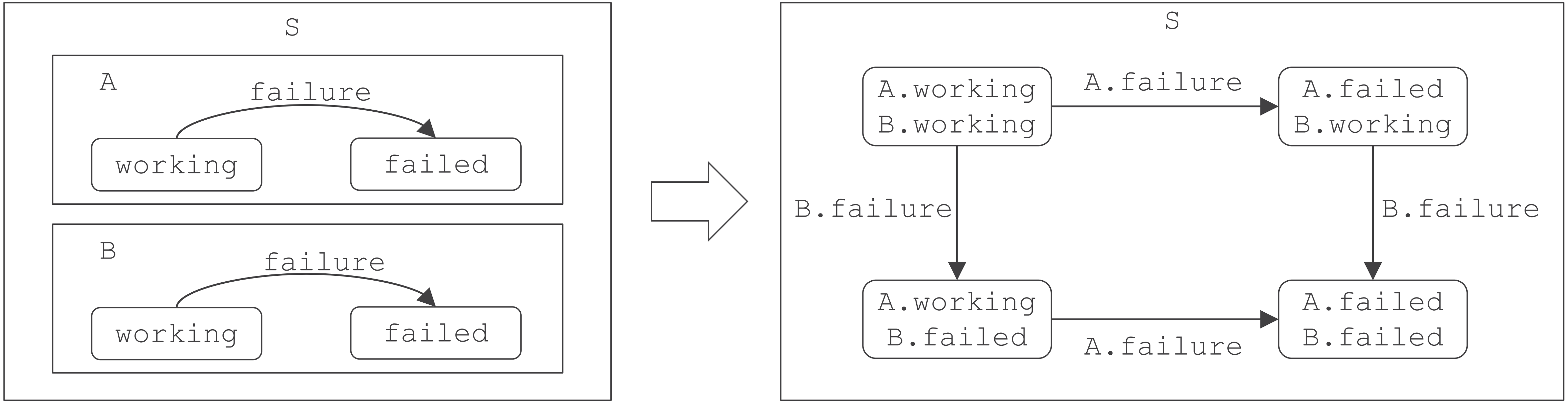
```
block A
   state working;
   state failed;
   transition
      failure: working -> failed;
end
```



Models are written in **modeling languages**.
There should be a unambiguous means to determine whether a given text (or diagram) is a correct a model or not. This means is called the **syntax** of the models, often described by means of a **grammar**.

```
Block ::= block Identifier StateDeclaration* Transition* end
StateDeclaration ::=  state State ;
Transition ::= transition Event : State -> State ;
Event ::= Identifier
State ::= Identifier
```

Formal Models Have a Semantics

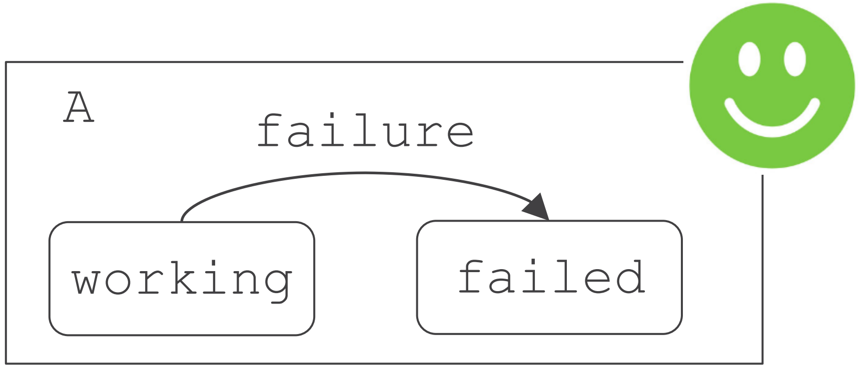


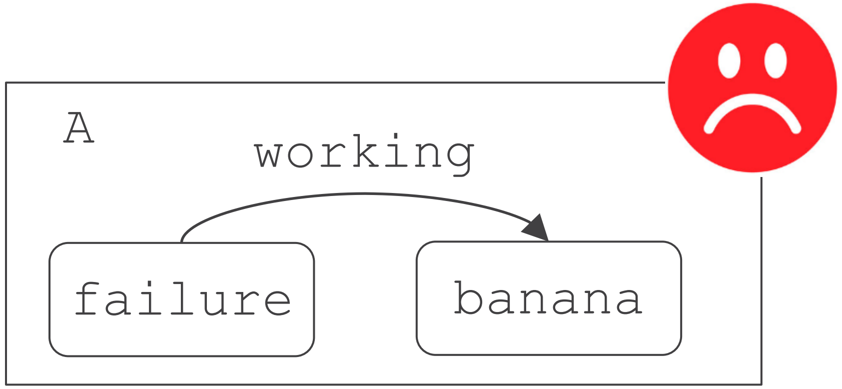
There should be an unambiguous way to interpret models into mathematical objects. This interpretation is the **semantics** of the model.

A formal semantics is the only way to justify computerized operations on models

Syntax and semantics are **domain independent**.

All Models Have a Pragmatics





Properties of models are interpreted into **properties of real systems**.
This interpretation is called the **pragmatics** of models.

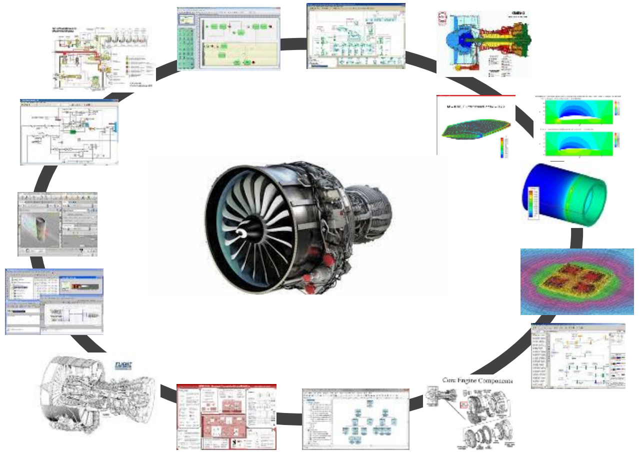
Facts about the pragmatics of models:

- It is at the very **core of the modeling process**.
- It is impossible to formalize as in requires a huge and **domain dependent** knowledge about systems.
- It is cultural and as such source of **ambiguities**.
- For these reasons, it should never be mixed up with the syntax and the semantics.

Agenda

- Introduction
- Models versus Notations
- **Mathematical Frameworks**
- The S2ML+X Paradigm
- Sigma and WorldLab
- Conclusion

Mathematical Frameworks



A model is always an **abstraction** of the system and is of interest because it is an abstraction.

The **properties of the system** to be studied determine the **mathematical framework** that should be used for the model.

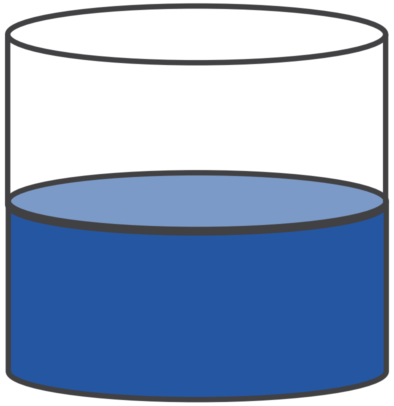
Experiments performed on the model have a **cost**. This cost is a key driver for the choice of the mathematical framework and the level of abstraction of the model.

The design of a model results always of a **tradeoff** between the **accuracy** of the description and the **cost** of experiments (calculations, simulations).

This is the reason why the **diversity** of models is **irreducible**.

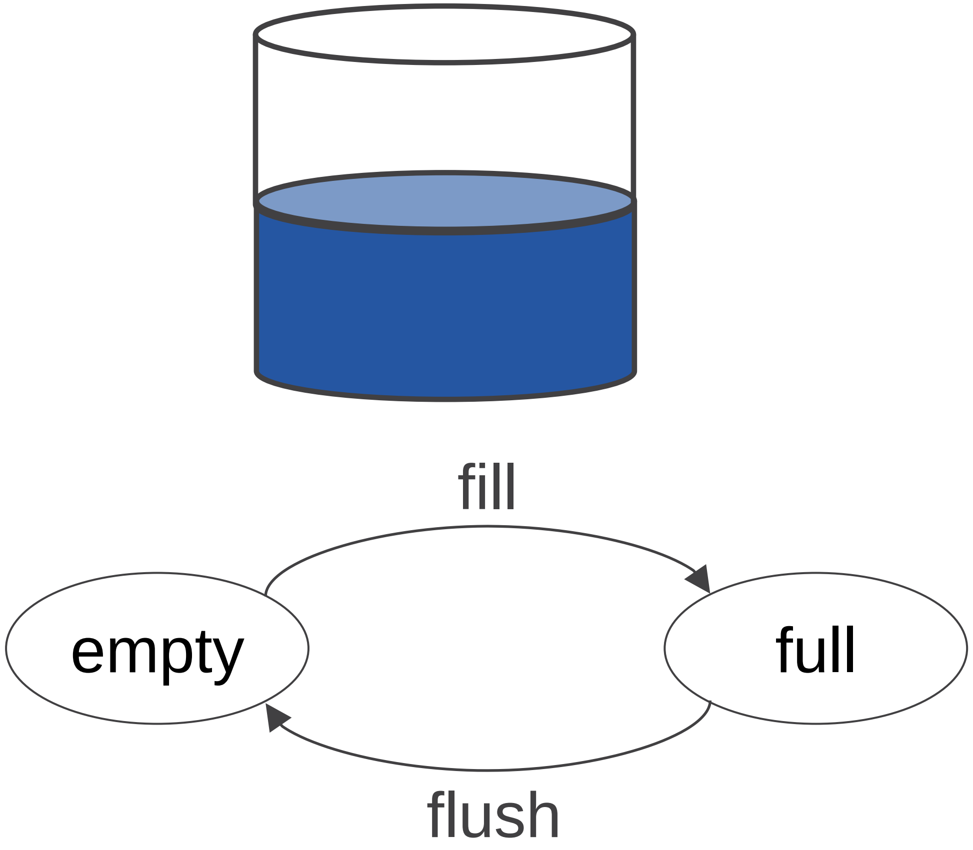
Mathematical Frameworks (bis)

Differential Equations



$$Qin - Qout = A\frac{dh}{dt}$$

Discrete Events



Mathematical Frameworks (ter)

Differential Equations

- At the core of physics
- Well mastered
- Powerful tools (Matlab/Simulink, Modelica)
- Very rich libraries of domain specific reusable modeling components
- Attempts to model systems (systems dynamics)
- Cost of simulations

Discrete Events

- More abstract vision (computer science)
- Better handling of non-deterministic and stochastic behaviors
- More efficient assessment tools

Classes of Event-Based Frameworks

The example of reliability engineering:

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

States Automata
- Markov chains
- Dynamic Fault Trees
- Stochastic Petri Nets
- AltaRica
- …

Agent-Based Models
- Process algebras
- High level Petri nets
- …

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 →

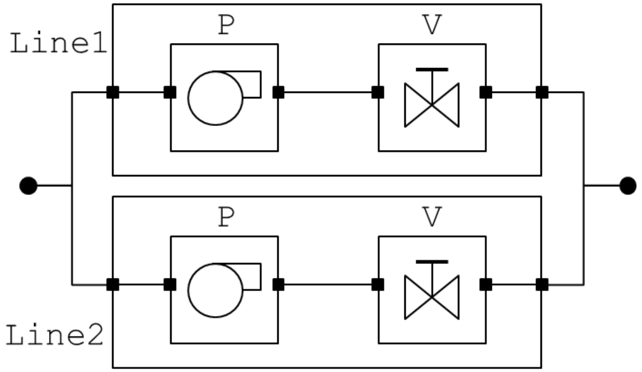
Agenda

- Introduction
- Models versus Notations
- Mathematical Frameworks
- **The S2ML+X Paradigm**
- Sigma and WorldLab
- Conclusion

Characteristics of Behavioral Models

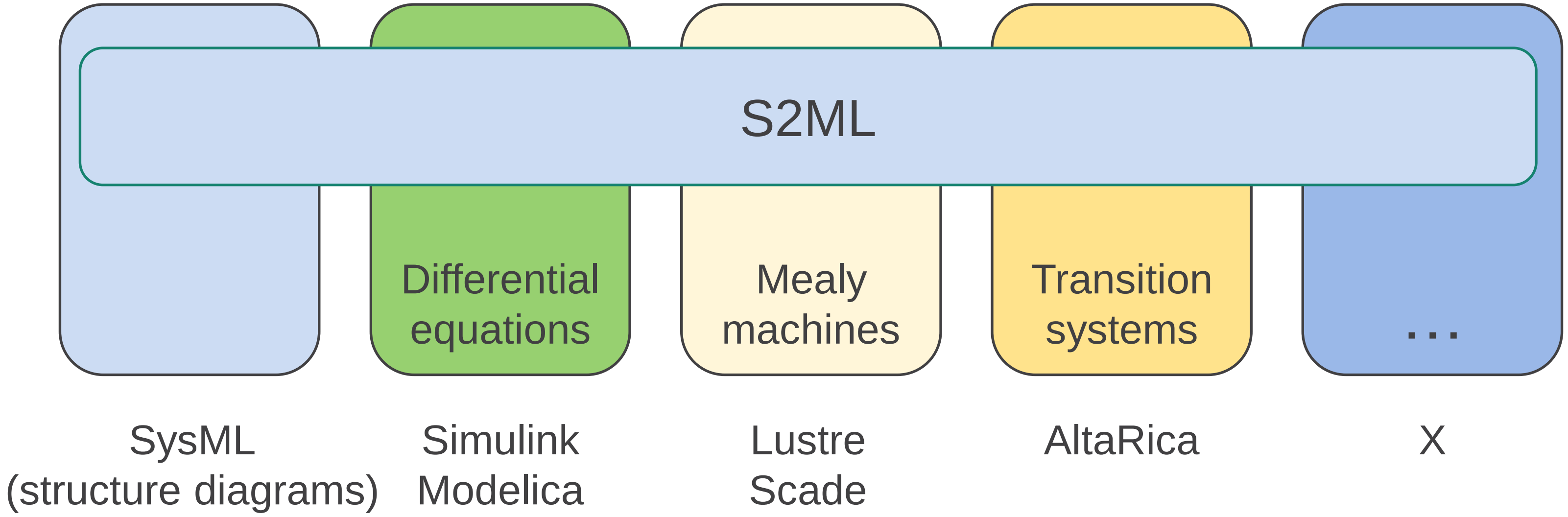
Behavior + Architecture = Model

- Any modeling language is the combination of a **mathematical framework** to describe the behavior and a **structuring paradigm** to organize the model.
- The choice of the **suitable mathematical framework** depends on which aspect of the system we want to study
- **Structuring paradigms** are to a very large extent **independent** of the chosen mathematical framework.



The S2ML+X Promise

S2ML (System Structure Modeling Language): a coherent and versatile set of **structuring constructs** for any behavioral modeling language.



- The **structure of models** reflects the **structure of the system**, even though to a **limited extent.**
- **Structuring** helps to design, to debug, to share, to maintain and to align heterogeneous models.

Models as Scripts

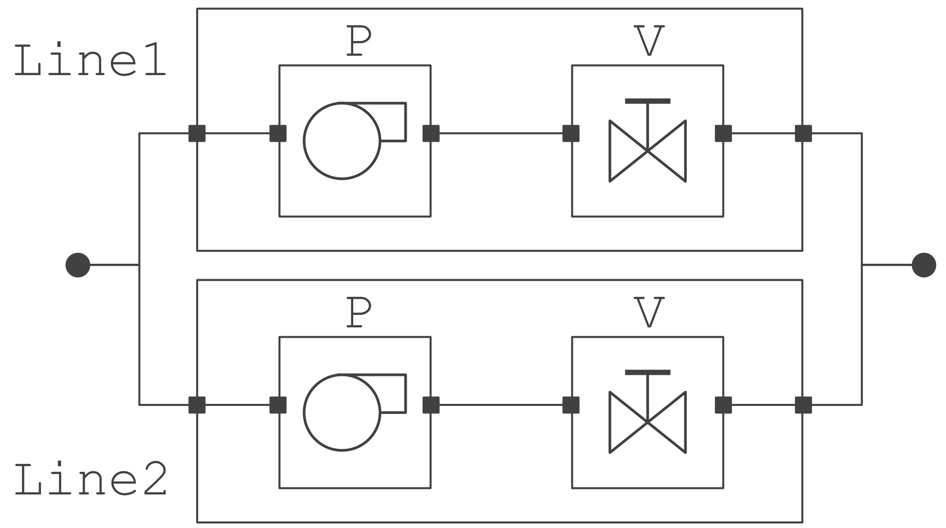
The **model "as designed"** is a script to build the **model "as assessed"**.

```
domain WF {WORKING, FAILED} WORKING<FAILED;

operator Series arg1 arg2 =
  return if state1==WORKING and state2==WORKING then WORKING else FAILED;

class Component
  WF state(init = WORKING);
  WF in, out(reset = WORKING)
  probability state FAILED = (exponentialDistribution lambda (missionTime));
  parameter Real lambda = 1.0e-3;
  assertion
    out := Series(in, state);
end
```

Complex models can be built using **libraries** of **reusable modeling components** and **modeling patterns**.

S2ML + Stochastic Boolean Equations

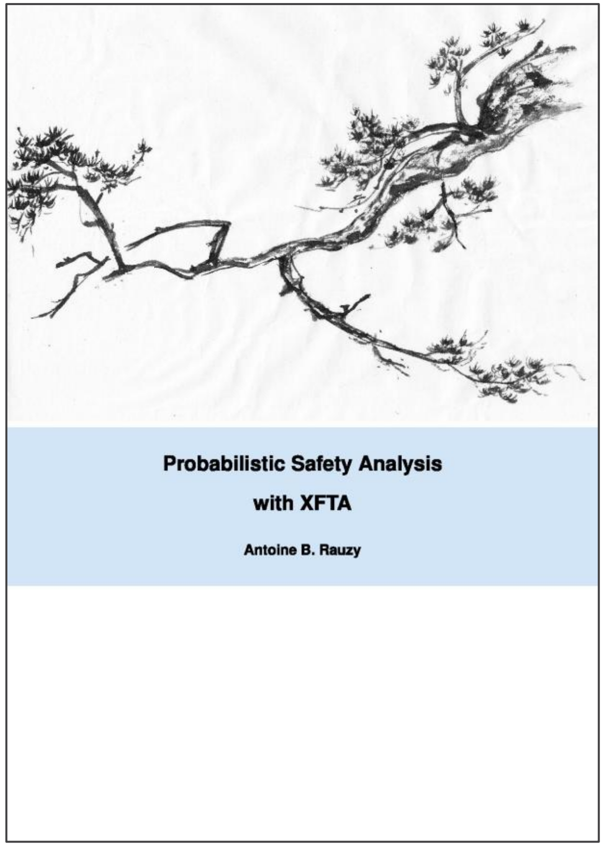
Enhancing classical **reliability models** (fault trees, reliability block diagrams) with the **expressive power of object-orientation** at **no algorithmic cost**




```
class Pump
      extends RepairableUnit
      …
end

block System
      block Line1
            Pump P;
            …
      end
      clones Line1 as Line2;
      …
end
```

```
Line1.in := in;
Line1.P.in := Line1.in;
Line1.P.out := Line1.P.in and not Line1.P.failed;
…
```

XFTA 2 + XFTA Book



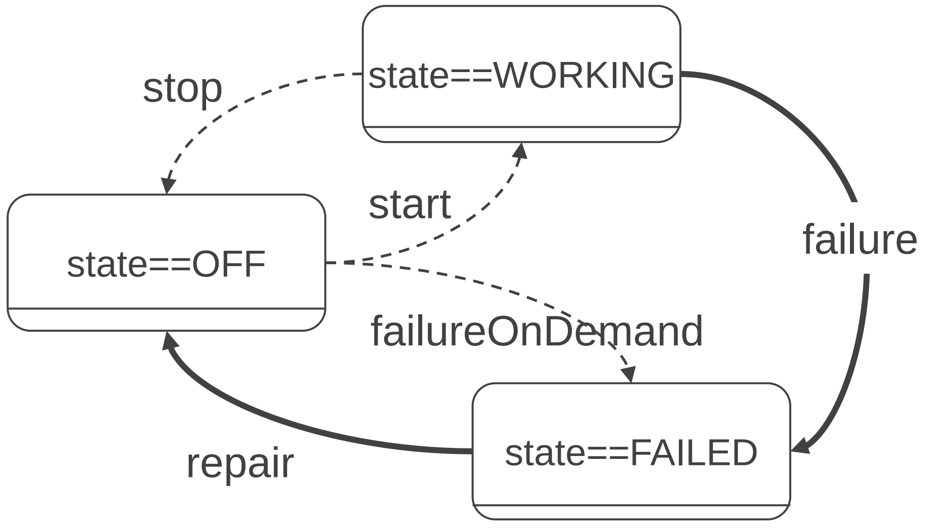
XFTA 2:

- Calculation engine for fault trees and related models.
- Input language: S2ML+SBE
- State of the art assessment algorithms: as of today, the most efficient calculation engine
- Calculation of all usual risk indicators:
  - Top event probability
  - Importance factors
  - Sensitivity analyses
  - Approximation of system reliability
  - Safety integrity levels
- Free of use.

AltaRica 3.0 (S2ML + Guarded Transitions Systems)

Guarded Transitions Systems:

- Are a probabilistic Discrete Events System framework.
- Are a compositional formalism.
- Generalize existing other mathematical framework, e.g. Petri nets.
- Take the best advantage of existing assessment algorithms.



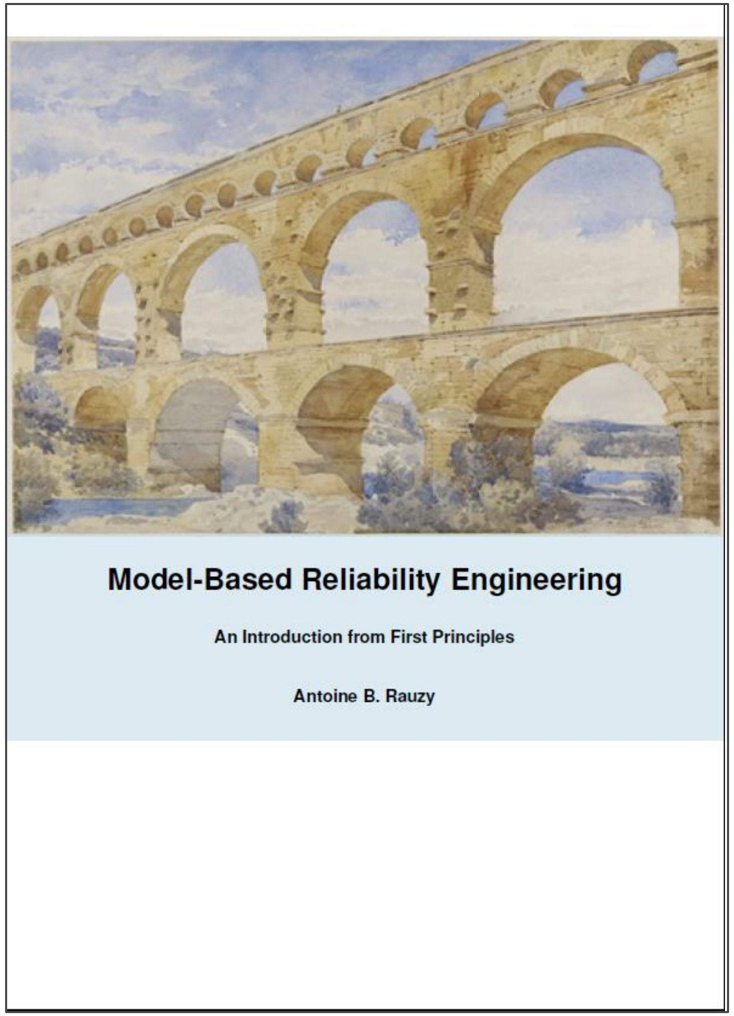
OpenAltaRica SYSTEM SAFETY ANALYSIS TECHNOLOGY | SystemX INSTITUT DE RECHERCHE TECHNOLOGIQUE

SAFRAN

THALES

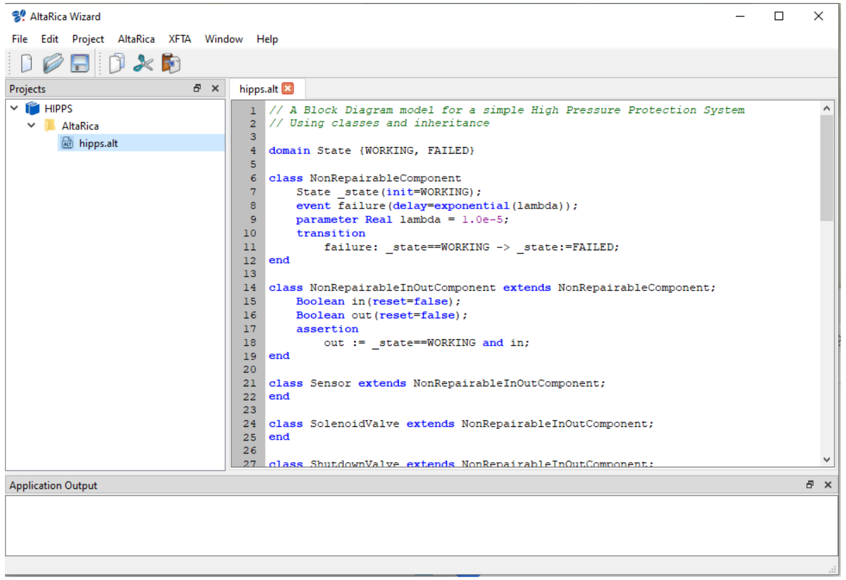
AIRBUS

AltaRica Wizard + MBRE Book



Integrated Modeling Environment AltaRica Wizard:

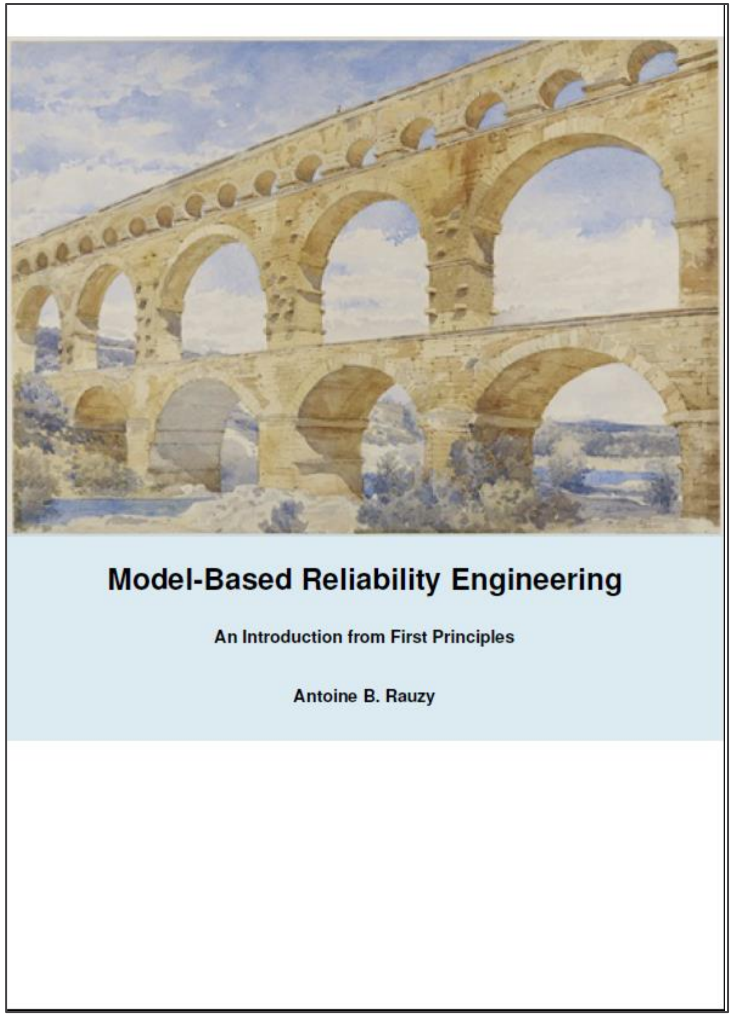
- Supports AltaRica 3.0 and S2ML+SBE (XFTA)
- Efficient assessment tools (interactive simulator, compiler to SBE, compiler to Markov chains, generator of critical sequences, stochastic simulator).
- Free of use.



Agenda

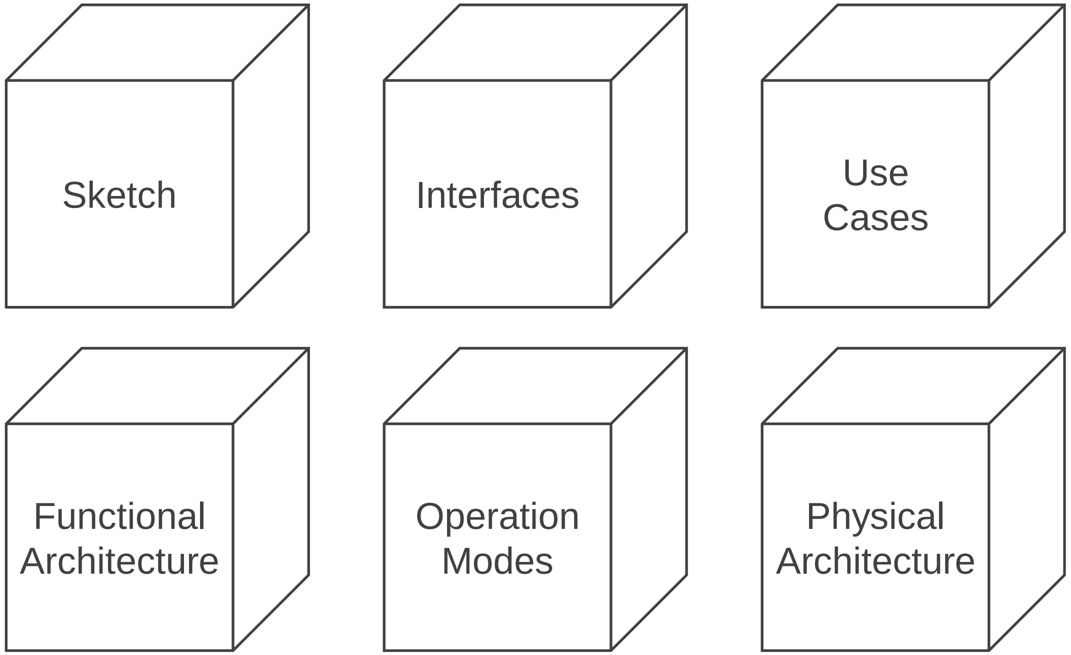
- Introduction
- Models versus Notations
- Mathematical Frameworks
- The S2ML+X Paradigm
- **Sigma and WorldLab**
- Conclusion

Cube Architecture Framework



Architecture frameworks are sets of good practices and guidelines for creating, interpreting, analyzing and using descriptions of systems.

The **cube architecture framework** consists of 6 views:



Sigma (Σ)

A new S2ML+X language dedicated to the design of **systemic digital twins**

```
system World.Supplier
  int rawMaterial(init = 0);
  bool renewing(init = false);
end

activity World.Supplier.RenewRawMaterialStock
  trigger:
    return rawMaterial<=1000 and not renewing;
  start:
    renewing = true;
  completion: {
    renewing = false;
    rawMaterial += 100;
    }
  duration:
    return 30;
end
```

- Σ = S2ML + Activity Algebra
- Discrete events
- Reactive
- Deformable systems (systems of systems)
- Pragmatic proofs

Agenda

- Introduction
- Models versus Notations
- Mathematical Frameworks
- The S2ML+X Paradigm
- Sigma and WorldLab
- **Conclusion**

Conclusion

Huge benefits can be expected from a full-scale deployment of model-based systems engineering. However, this requires:

- To set up solid **scientific foundations**.
- To **bring to maturity** some **key technologies**.

The **biggest challenge** is to **train new generation of scientists and engineers**:

- With skills and competences in **discrete mathematics** and **computer science**,
- With skills and competences in **software engineering**,
- With skills and competences in **system thinking**,
- With skills and competences in **specific application domains**.

CSER 2022

conference on systems engineering research

cser2022.cser.info