ONERA, Toulouse 22/11/2019

On Four Questions about Model-Based Systems Engineering and Model-Based Reliability Engineering

Prof. Antoine B. Rauzy

Department of Mechanical and Industrial Engineering
Norwegian University of Science and Technology
Trondheim, Norway

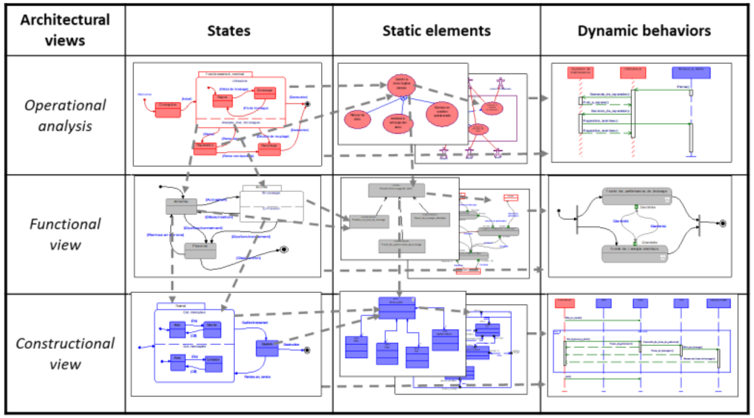
&

Chair Blériot-Fabre
CentraleSupélec/SAFRAN
Paris, France

Systems Engineering vs Reliability Engineering

Systems Engineering

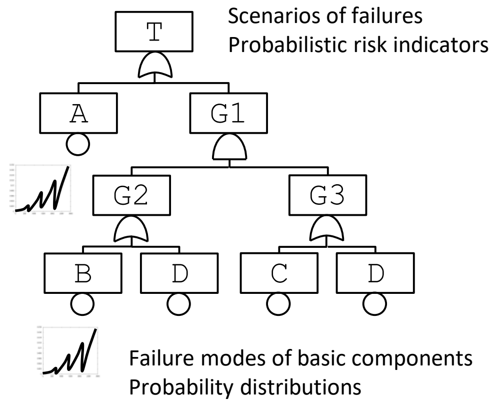
What the system should do?
What the system should be?



Proof that there exists a system that meets the given specification.

Reliability Engineering

What can go wrong?
What is the severity of consequences?
What is the likelihood?



Proof that the specified system is reliable enough to be operated.

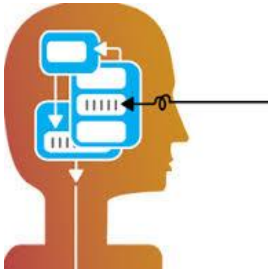
Agenda

- Essential differences between models designed by systems engineers and those designed by reliability engineers.
- Specificities of models designed by reliability engineers.
- Potential commonalities between models designed by systems engineers and those designed by reliability engineers.
- Alignment of models designed by systems engineers and models designed by reliability engineers.
- Concluding remarks

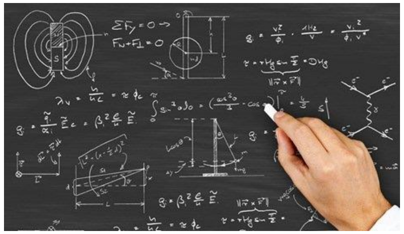
Agenda

- **Essential differences between models designed by systems engineers and those designed by reliability engineers.**
- Specificities of models designed by reliability engineers.
- Potential commonalities between models designed by systems engineers and those designed by reliability engineers.
- Alignment of models designed by systems engineers and models designed by reliability engineers.
- Concluding remarks

What Do We Call a Model?



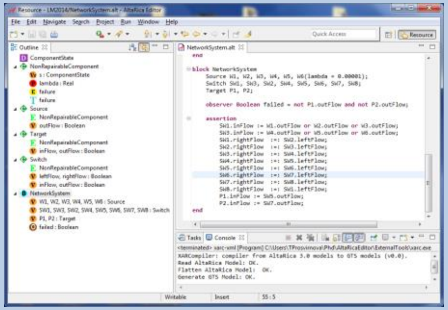
Cognitive Model



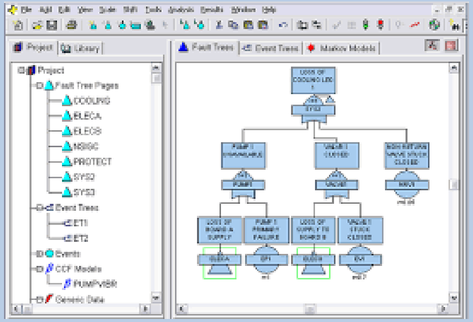
Mathematical Model

Models *in abstracto*

Models *in silico*



Text



Diagrams

Models versus Notations

A **behavioral model *in silico***, would it be authored graphically, must have:

1. A well-defined **syntax**:

   It must be possible to determine automatically whether the model is syntactically correct, i.e. if it obeys the grammar of its modeling language.

2. A well-defined **semantics**:

   Each model must be associated without ambiguity with a mathematical object of some algebra. Computerized operations on models must be justified by the properties of the operators of this algebra.

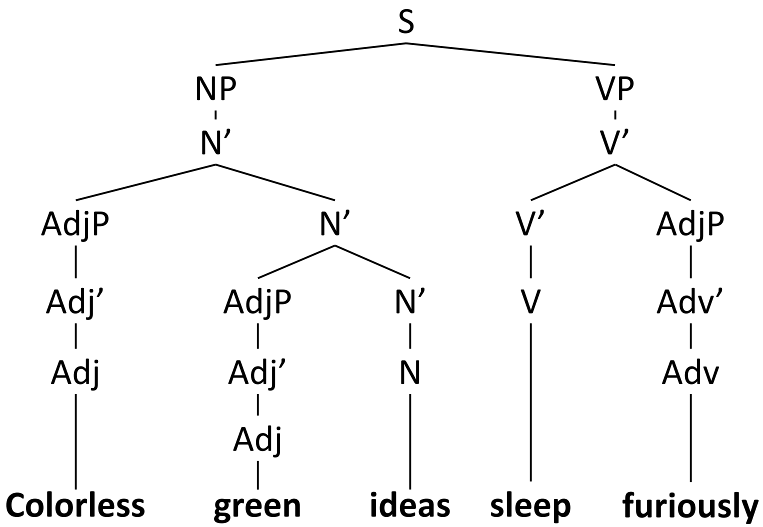
3. A well-understood **pragmatics**:

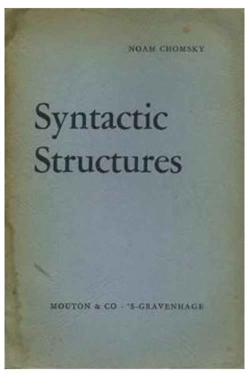
   Analysts must be able to relate the mathematical object encoded by the model with the actual behavior of the system under study (and to agree on this relation).

Texts and/or diagrams that do not have the above properties should not be called models, but simply **notations**.
More or less standardized **notations play an extremely important role** in engineering, but they do not have the epistemic status of models.

Pragmatics

In linguistics and semiotics, **pragmatics** designates studies about how the **context** of a discourse contributes to its **meaning**.





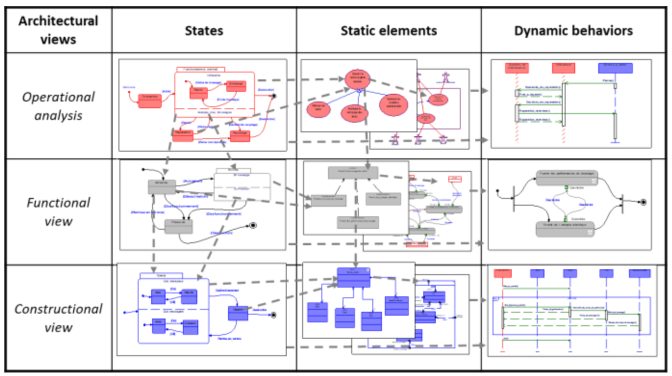
Noam Chomsky (1957)

In model-driven engineering, the **pragmatics of a model** is the body of **implicit knowledge** that is used to author and to use this model. This body of knowledge is hopefully **shared by the stakeholders** who, for this very reason, do not need to make it explicit in the model.

Pragmatic versus Formal Models

Systems Engineering

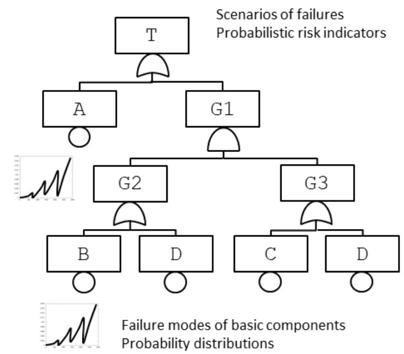
Models to communicate
amongst stakeholders



Pragmatic proof that there exists a system that meets the given specification.

Reliability Engineering

Models to calculate
performance indicators



Formal proof that the specified system is reliable enough to be operated.

Epistemic gap

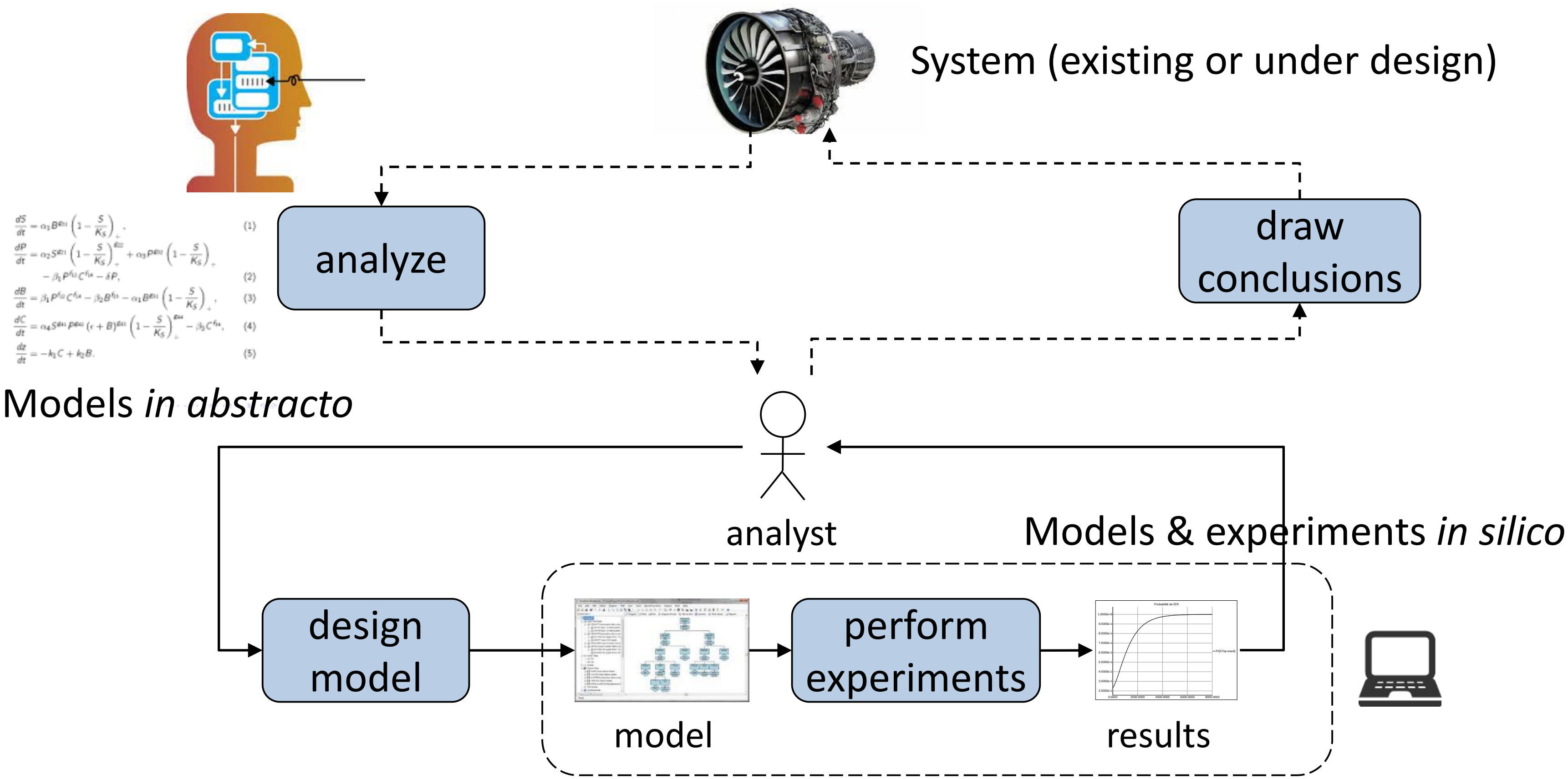
Agenda

- Essential differences between models designed by systems engineers and those designed by reliability engineers.
- **Specificities of models designed by reliability engineers.**
- Potential commonalities between models designed by systems engineers and those designed by reliability engineers.
- Alignment of models designed by systems engineers and models designed by reliability engineers.
- Concluding remarks

Complexity of Virtual Experiments

In reliability engineering, a model results always of a **tradeoff** between the **accuracy of the description** and the **computational complexity** of calculations.

Computational complexity issues **determine ultimately** the modeling process (an embodiment of Simon's concept of **bounded rationality**).



Classes of Modeling Languages

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
- Netlogo
- …

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 →

(R)evolution in Reliability Engineering

Today:

Mechanical systems

Recording of failures

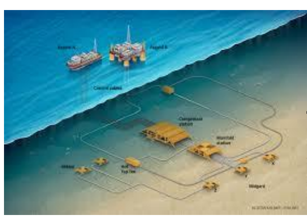
Local reliability databases

λ = 1.23e-6

Parametric distributions

Ad-hoc models, e.g. fault trees

Tomorrow:

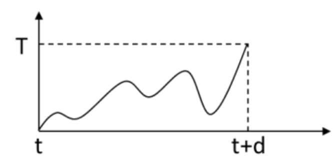


Cyber-physical systems

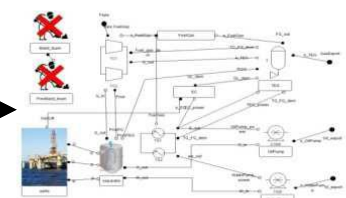
Health monitoring

Sensors

Distributed health condition databases



Learned distributions



Behavioral models

Agenda

- Essential differences between models designed by systems engineers and those designed by reliability engineers.
- Specificities of models designed by reliability engineers.
- **Potential commonalities between models designed by systems engineers and those designed by reliability engineers.**
- Alignment of models designed by systems engineers and models designed by reliability engineers.
- Concluding remarks

Behaviors + Structures = Models

The behavioral model of a complex system cannot be simple. The complexity cannot vanish. Modeling aims at **simplexity** (in Berthoz's sense).

Any behavioral modeling language is the combination of a **mathematical framework** in which the behavior is described and a **structuring paradigm** to organize the model.

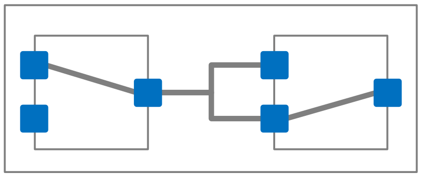
The choice of the **suitable mathematical framework** depends on which aspect of the system we want to study.

Structuring constructs help to design, to understand, to share and to maintain models through the life-cycles of systems.

Structuring paradigms are to a very large extent **independent** of the chosen mathematical framework.

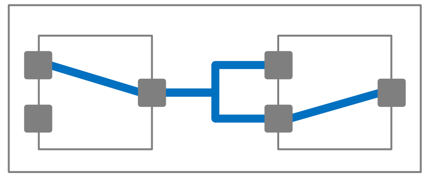
Ontology/Meta-Model of Behavioral Models

Port



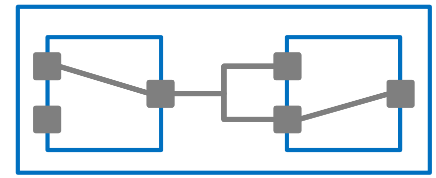
Variable, event…

Connection



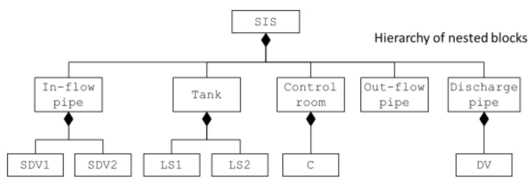
Equation, transition…

Container



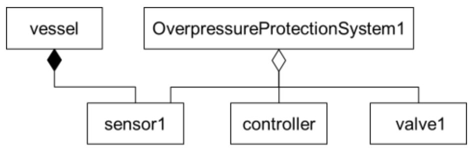
Model, component…

Composition



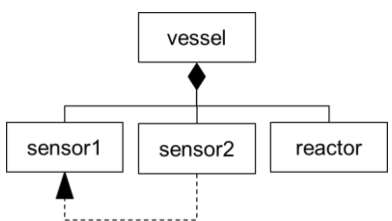
Is-part-of

Aggregation

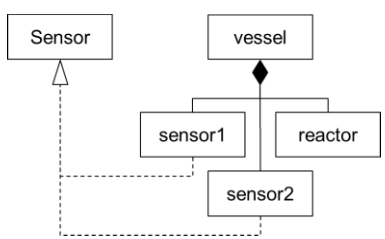


Uses

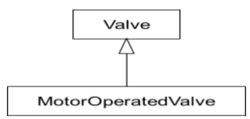
Prototype/Cloning



Class/Instantiation



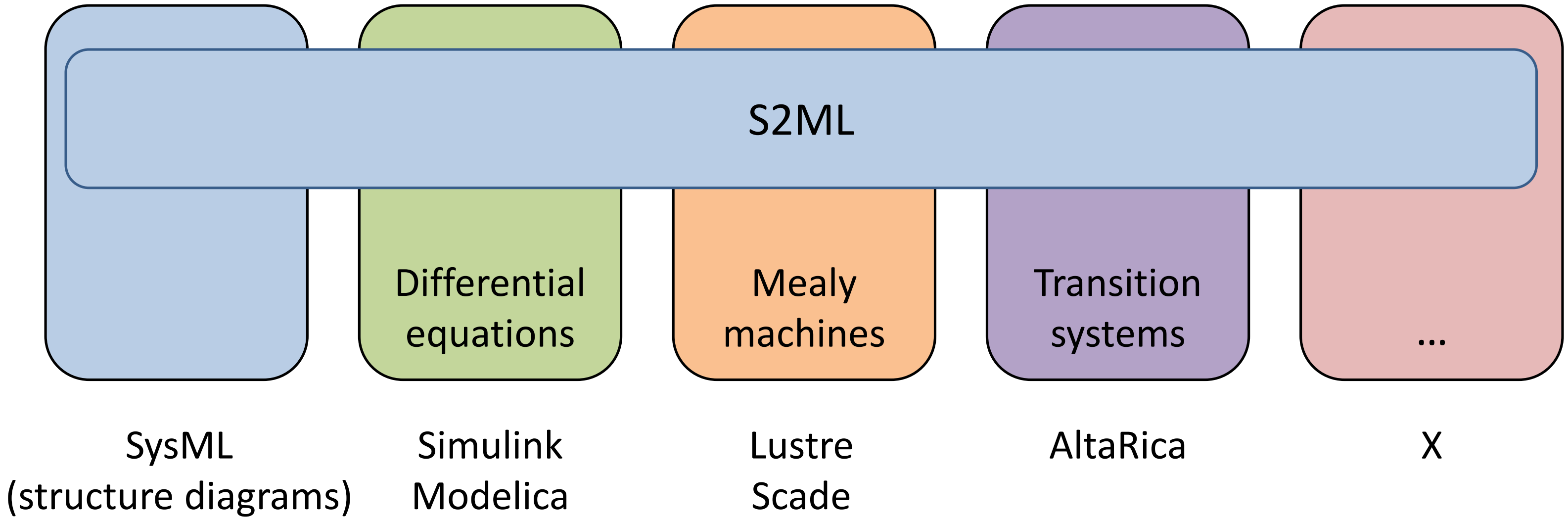
Inheritance



Is-a

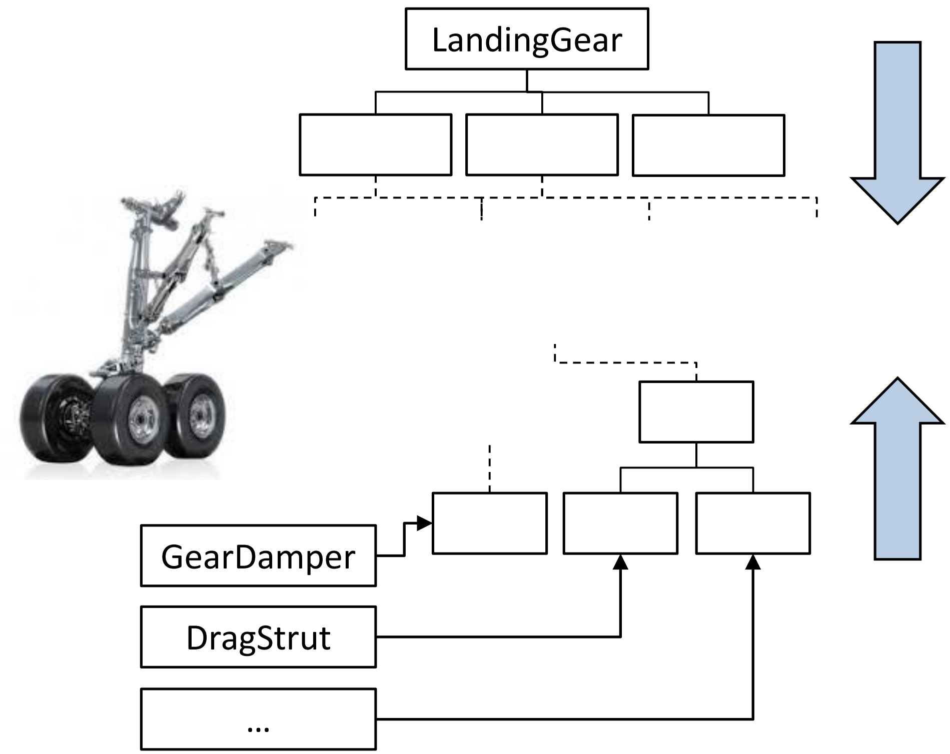
The S2ML+X Promise

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.

Modeling Approaches



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

system architecture

safety

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

Multiphysics simulation

These conceptual foundations echo results obtained in **cognitive science**, e.g. Lakoff categories of thoughts, in **management science**, e.g. Hatchuel's C-K theory, and of course in **software engineering** via **programming paradigms** and the notion of **design patterns**.

Models as Scripts

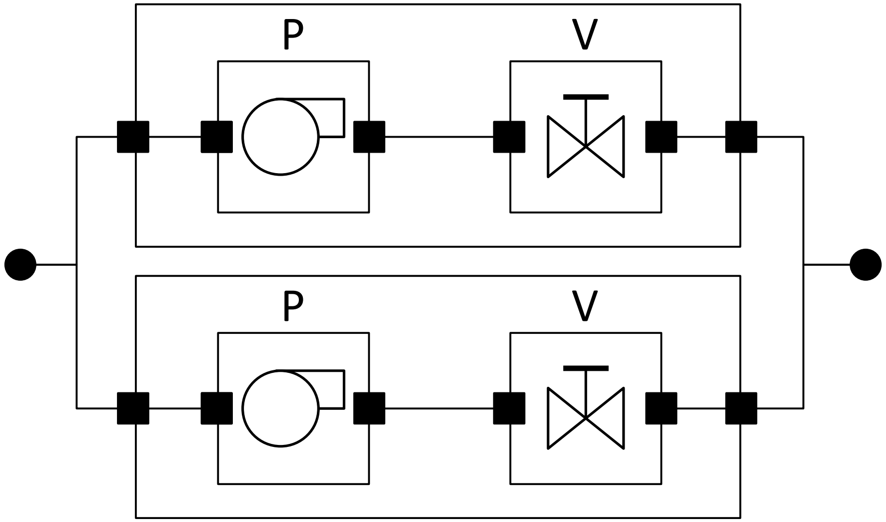
The **model "as designed"** is a script to build the **model "as assessed"**.

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

operator Series arg1 arg2 =
   (if (and (eq state1 WORKING) (eq state2 WORKING))
      WORKING
      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**.

Open-PSA V4 (S2ML + Boolean Equations)

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

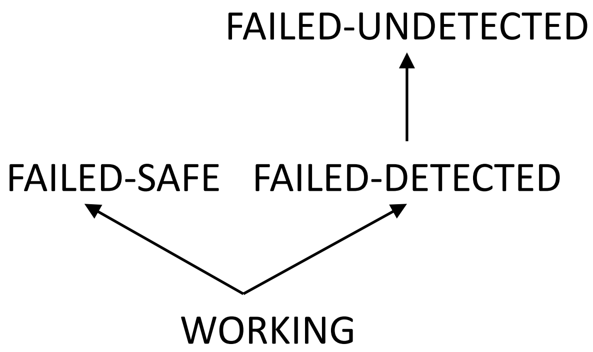



```
class Pump
    extends Component
    …
end

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;
…
```

S2ML + Finite Degradation Structures

Lifting-up all classical concepts of reliability engineering to **multi-valued logics** and giving these logics the **expressive power of object-orientation**.



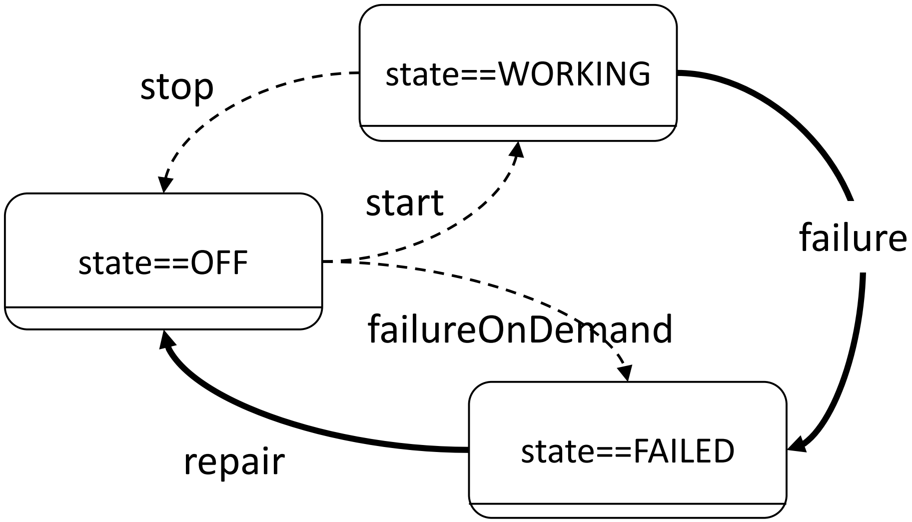
||	W	Fs	Fd	Fu
W	W	W	W	W
Fs	W	Fs	Fs	Fs
Fd	W	Fs	Fd	Fd
Fu	W	Fs	Fd	Fu

```
domain IEC61508
    {WORKING, FAILED_SAFE,
     FAILED_DETECTED,
     FAILED_UNDETECTED}
    WORKING<FAILED_SAFE,
    WORKING<FAILED_DETECTED,
    …

operator Parallel
    …
end
```

AltaRica 3.0 (S2ML + Guarded Transitions Systems)

Guarded Transitions Systems:

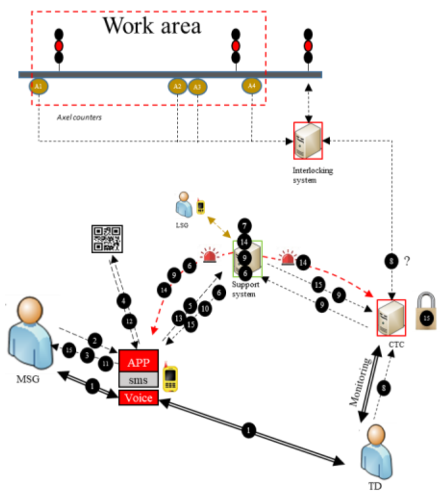
- Are a probabilistic Discrete Events System formalism.
- Are a compositional formalism.
- Generalize existing mathematical framework.
- Take the best advantage of existing assessment algorithms.

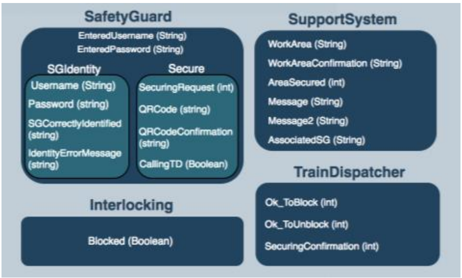


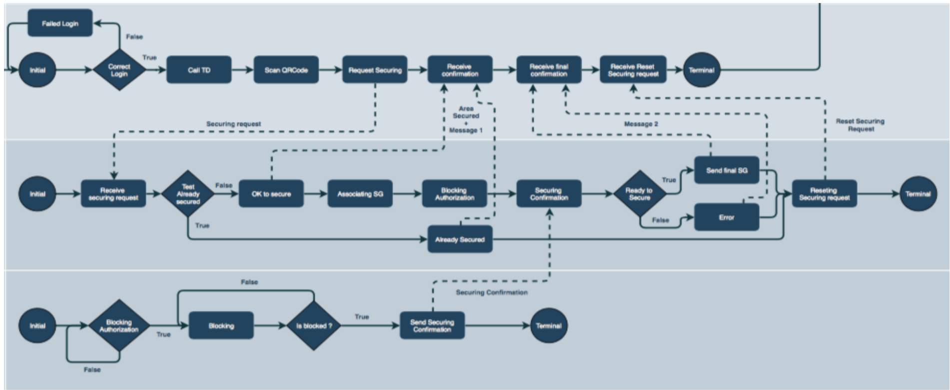
Scola (S2ML + Process Algebra)

Scenario-oriented modeling methodology

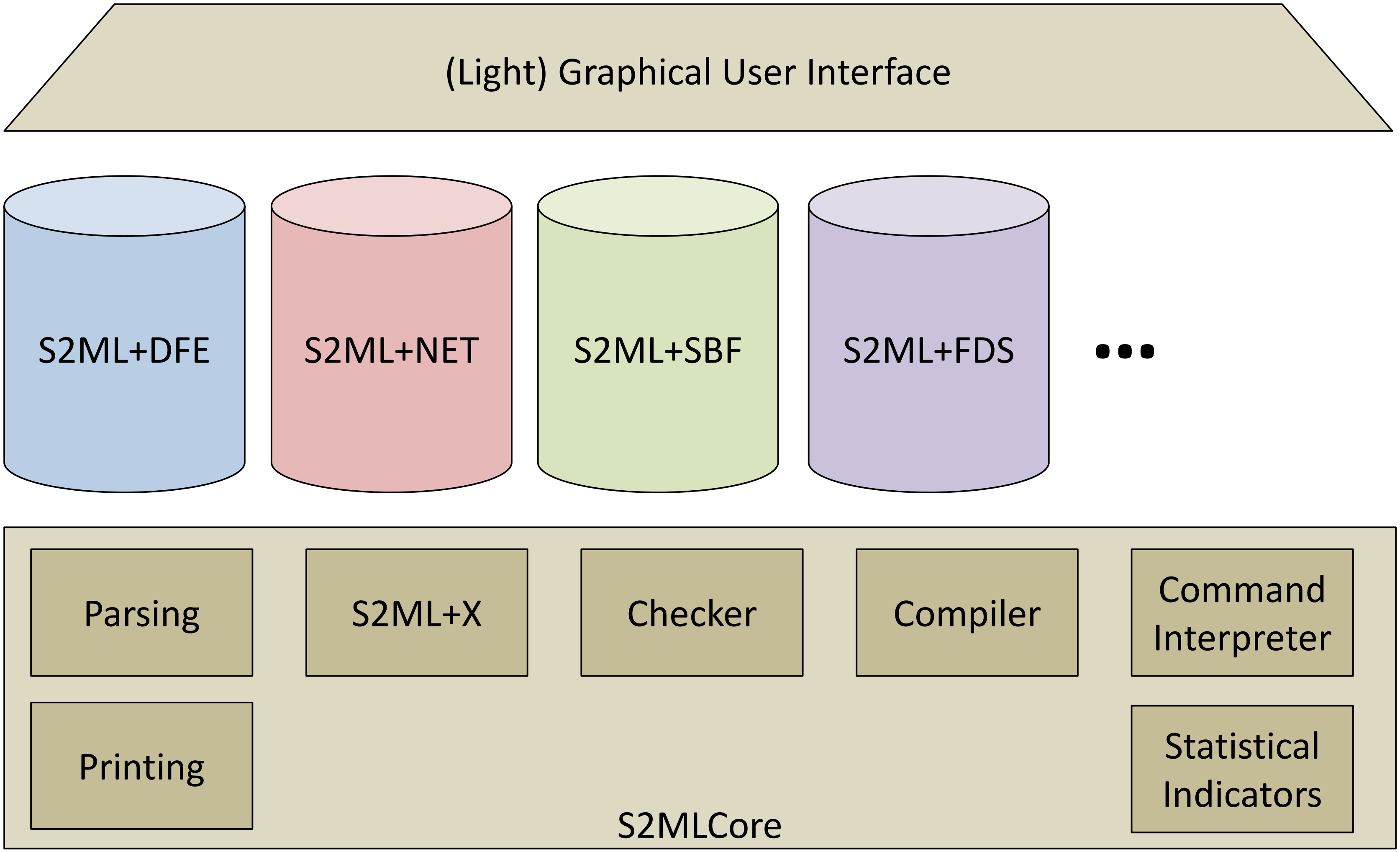
- Architecture description
- Dynamic modification of components
- Moving components
- Dynamic creation/deletion of components







S2ML Toolbox (Proof of Concept)



The S2ML+X Paradigm in Pedagogical Action

Versatile set of **domain specific modeling languages**

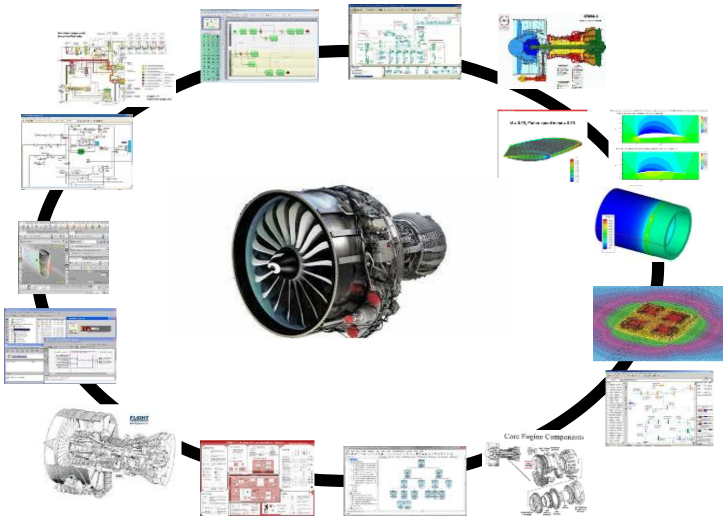
Domain	Language
System architecture (structural diagrams)	S2ML
Stochastic simulation	S2ML + data-flow equations
Combinatorial Optimization	S2ML + constraints
Reliability Engineering	S2ML + stochastic Boolean formulas
Logistics	S2ML + hierarchical graphs
Stochastic processes	S2ML + Markov chains
Model-checking	S2ML + finite state automata
Discrete event systems	S2ML + guarded transition systems (AltaRica)
Business processes	S2ML + process algebra (Scola)
…	…

Agenda

- Essential differences between models designed by systems engineers and those designed by reliability engineers.
- Specificities of models designed by reliability engineers.
- Potential commonalities between models designed by systems engineers and those designed by reliability engineers.
- **Alignment of models designed by systems engineers and models designed by reliability engineers.**
- Concluding remarks

Model Diversity

Models are designed by **different teams** in **different languages** at **different levels of abstraction**, for **different purposes**, making **different approximations**. They have also **different maturities**.



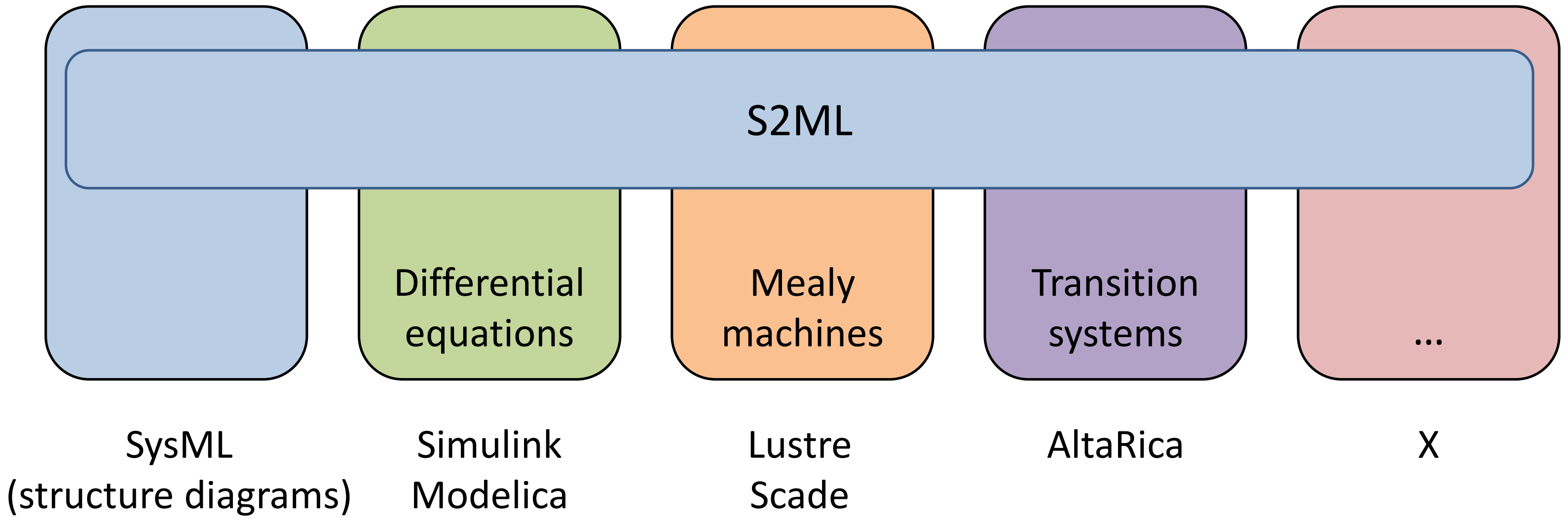
The **diversity** of models is **irreducible.**

Alignment of Heterogeneous Models

Models are designed by **different teams** in **different languages** at **different levels of abstraction**, for **different purposes.** They have also **different maturities**.

The question is how to ensure that they are "speaking" about the **same system**, i.e. to **align** them.

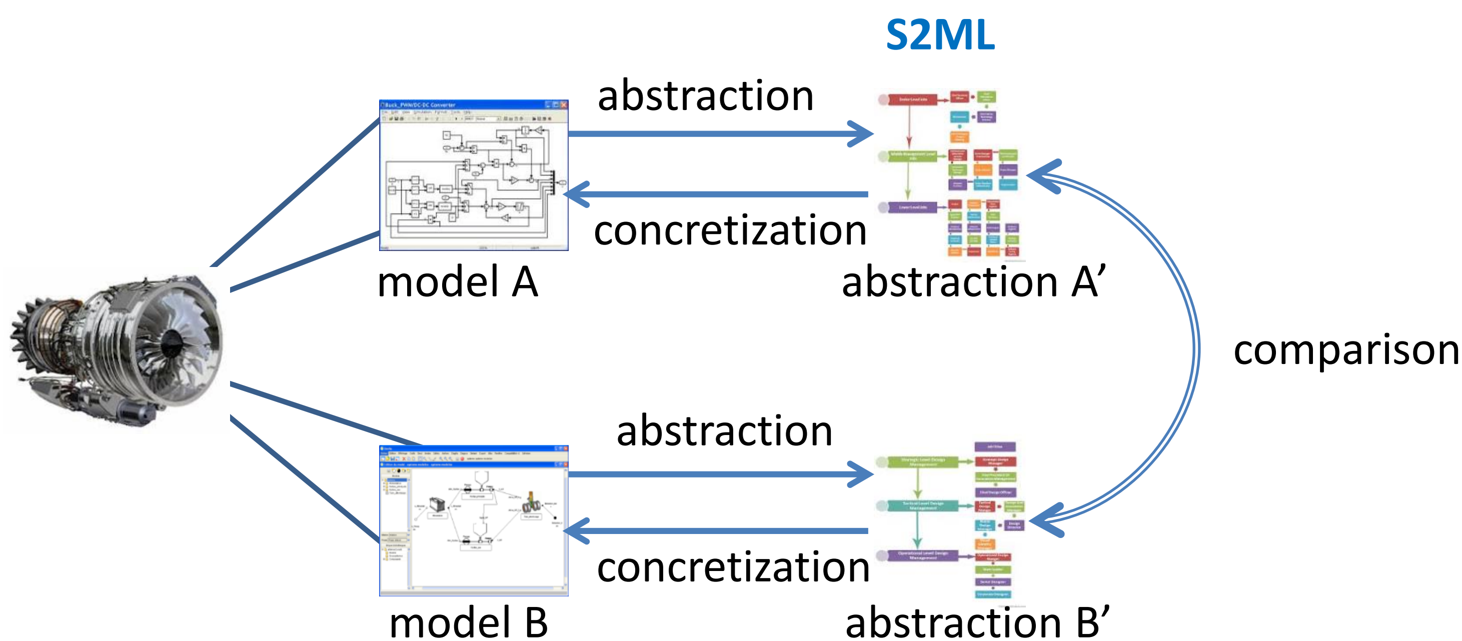
As the **behavioral part** of models is **purpose-dependent**, the main and most often the only way to align models is to compare their **structure**.



Model Synchronization

Model synchronization provides a **formal framework**, inspired from Cousot's **abstract interpretation**, to **align heterogeneous models**.

Abstraction + Comparison = Synchronization



Synchronizing models does not mean making them fully compatible. This would be too ambitious because of the **heterogeneity of concerns**. Rather, the question at stake is **how to agree on disagreements?**

Agenda

- Essential differences between models designed by systems engineers and those designed by reliability engineers.
- Specificities of models designed by reliability engineers.
- Potential commonalities between models designed by systems engineers and those designed by reliability engineers.
- Alignment of models designed by systems engineers and models designed by reliability engineers.
- **Concluding remarks**

Concluding Remarks (1)

- "Traditional" modeling approaches in reliability engineering are **no longer sufficient**:
  - Because the **systems** we are dealing with are **more complex**.
  - Because **new information technologies** open **new opportunities**.
  - Because **reliability models** should be **integrated** with models from other engineering disciplines, especially with those designed by systems engineers.

- **Huge benefits** can be expected from a full-scale deployment of model-based systems engineering. However, this requires:
  - To set up solid **scientific foundations** for **models engineering**.
  - To **bring to maturity** some **key technologies**.

Concluding Remarks (2)

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

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