CSDM 2019 – Paris 13/12/2019

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

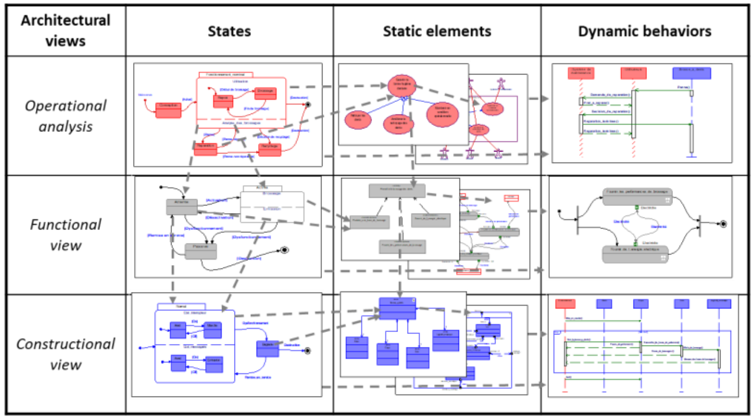
&

Chair Blériot-Fabre
CentraleSupélec/SAFRAN
Paris, France

Systems Engineering vs Reliability Engineering

Systems Engineering

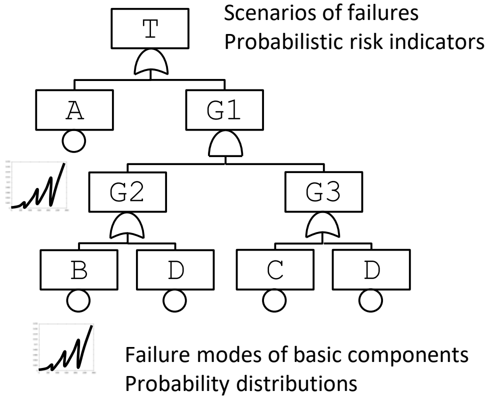
What the system should do?
What the system should be?



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

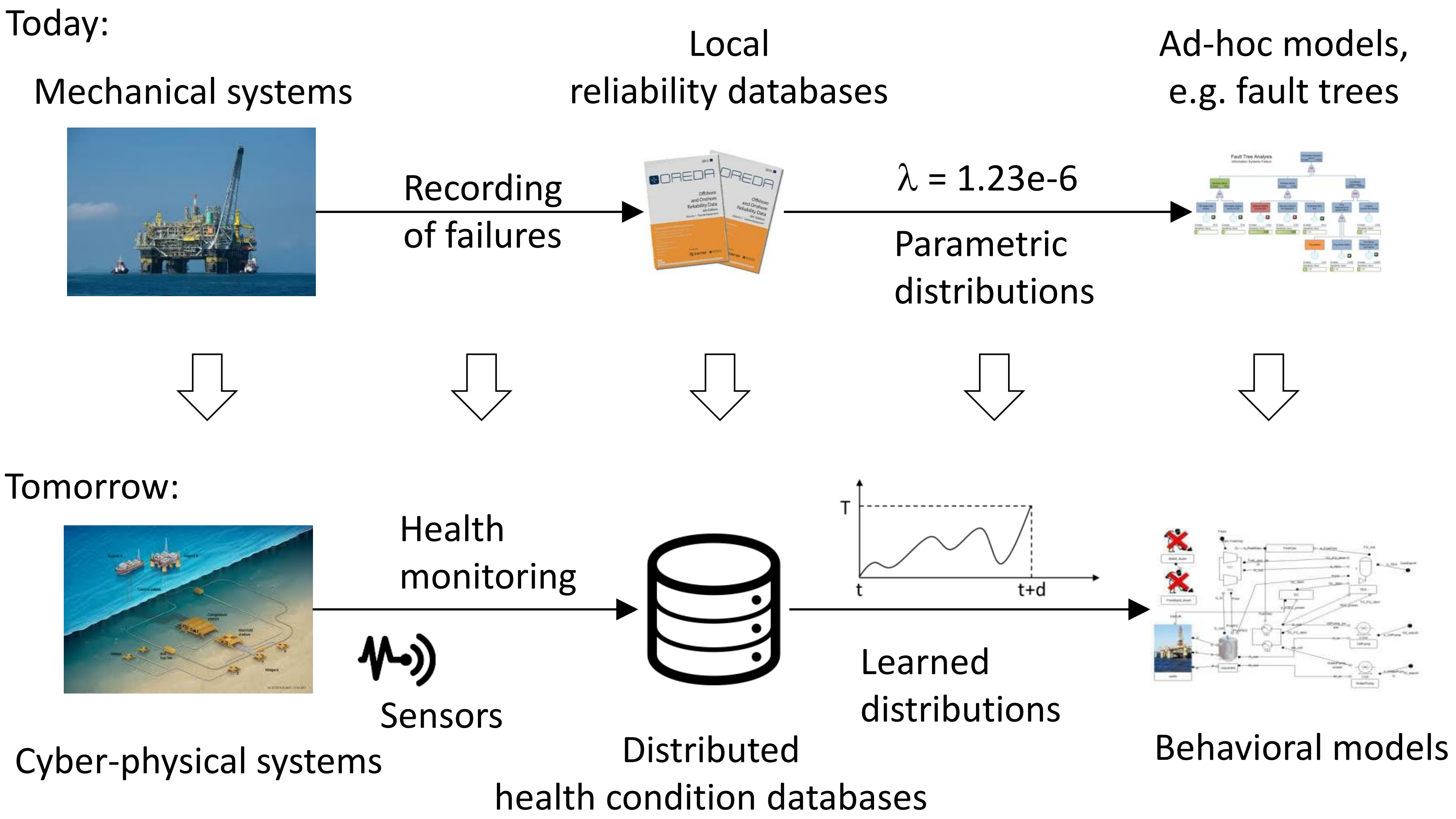
Reliability Engineering

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


T
Scenarios of failures
Probabilistic risk indicators
A
G1
G2
G3
B
D
C
D
Failure modes of basic components
Probability distributions

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

(R)evolution in Reliability Engineering



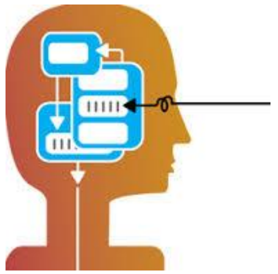
Agenda

- 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

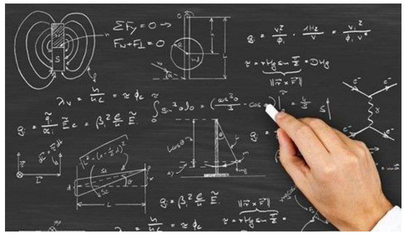
Agenda

- **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?



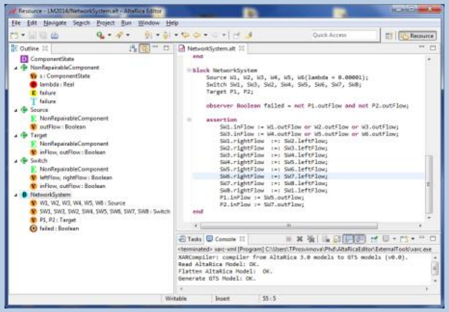
Cognitive Model



Mathematical Model

Models *in abstracto*

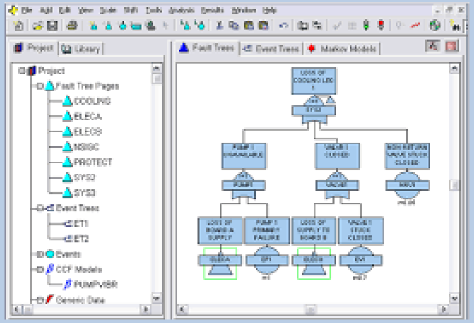
Models *in silico*



```
end

block NetworkSystem
    Source W1, W2, W3, W4, W5, W6(lambda = 0.00001);
    Switch SW1, SW3, SW2, SW4, SW5, SW6, SW7, SW8;
    Target P1, P2;

    observer Boolean failed = not P1.outFlow and not P2.outFlow;

    assertion
        SW1.inFlow := W1.outFlow or W2.outFlow or W3.outFlow;
        SW3.inFlow := W4.outFlow or W5.outFlow or W6.outFlow;
        SW1.rightFlow  :=: SW2.leftFlow;
        SW2.rightFlow  :=: SW3.leftFlow;
        SW3.rightFlow  :=: SW4.leftFlow;
        SW4.rightFlow  :=: SW5.leftFlow;
        SW5.rightFlow  :=: SW6.leftFlow;
        SW6.rightFlow  :=: SW7.leftFlow;
        SW7.rightFlow  :=: SW8.leftFlow;
        SW8.rightFlow  :=: SW1.leftFlow;
        P1.inFlow := SW5.outFlow;
        P2.inFlow := SW7.outFlow;
end
```

```
<terminated> xarc-xml [Program] C:\Users\TProsvirnova\Phd\AltaRicaEditor\ExternalTools\xarc.exe
XARCompiler: compiler from AltaRica 3.0 models to GTS models (v0.0).
Read AltaRica Model: OK.
Flatten AltaRica Model:  OK.
Generate GTS Model: OK.
```

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.

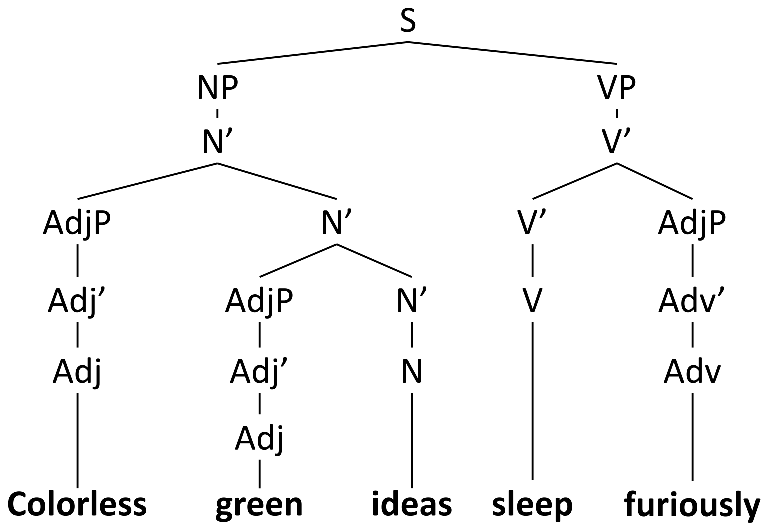
3 A well-understood **pragmatics**:

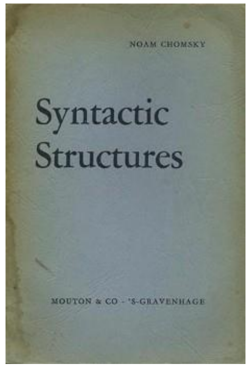
Analysts must be able to relate the mathematical object encoded by the model with the actual behavior of the system under study (and to agree on this relation).

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**.





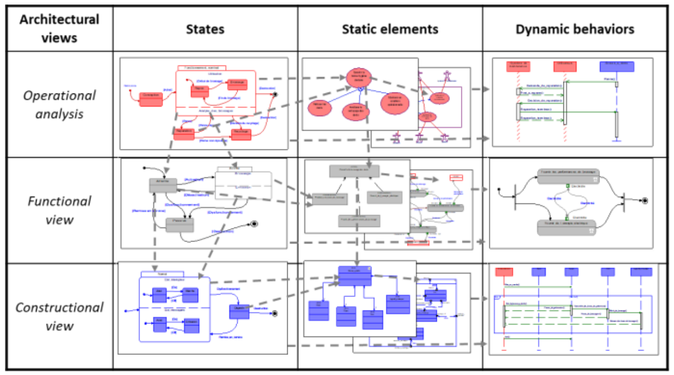
Noam Chomsky (1957)

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

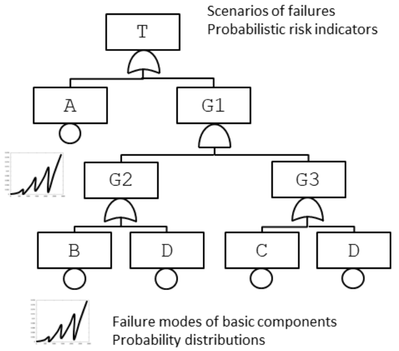
Models to communicate amongst stakeholders



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

Reliability Engineering

Models to calculate performance indicators


Scenarios of failures
Probabilistic risk indicators
T
A
G1
G2
G3
B
D
C
D
Failure modes of basic components
Probability distributions

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

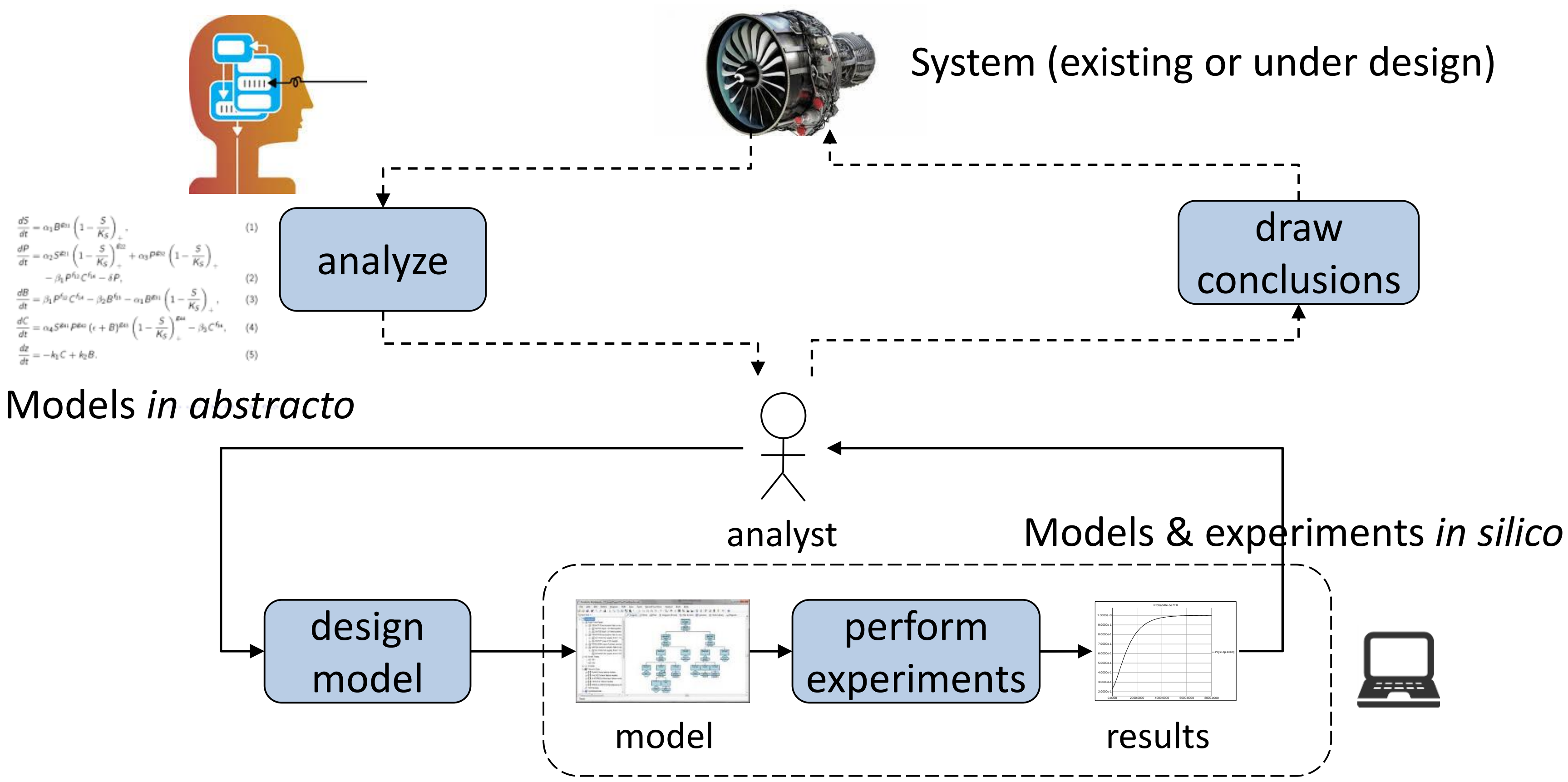
Epistemic gap

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	States Automata	Agent-Based Models
• Fault Trees • Event Trees • Reliability Block Diagrams • Finite Degradation Structures	• Markov chains • Dynamic Fault Trees • Stochastic Petri Nets • AltaRica • …	• 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 →

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.

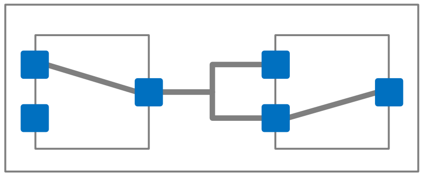
The choice of the **suitable mathematical framework** depends on which aspect of the system we want to study.

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.

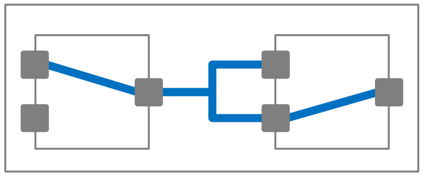
Ontology/Meta-Model of Behavioral Models

Port



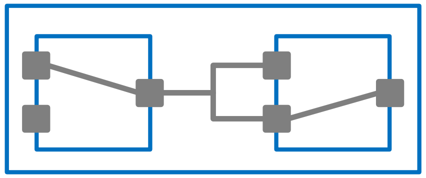
Variable, event…

Connection



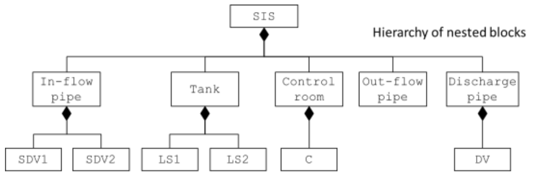
Equation, transition…

Container



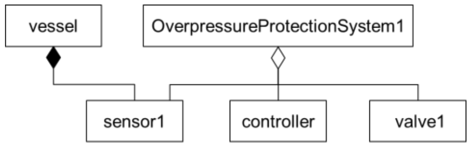
Model, component…

Composition



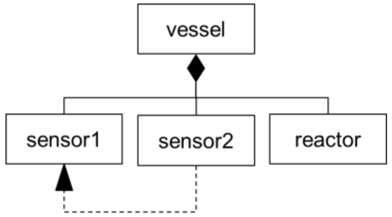
Is-part-of

Aggregation

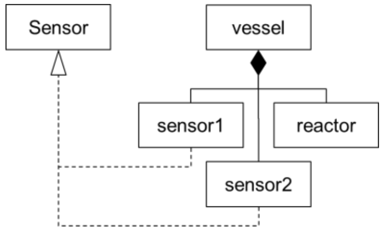


Uses

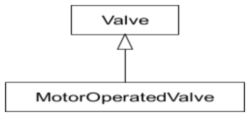
Prototype/Cloning



Class/Instantiation



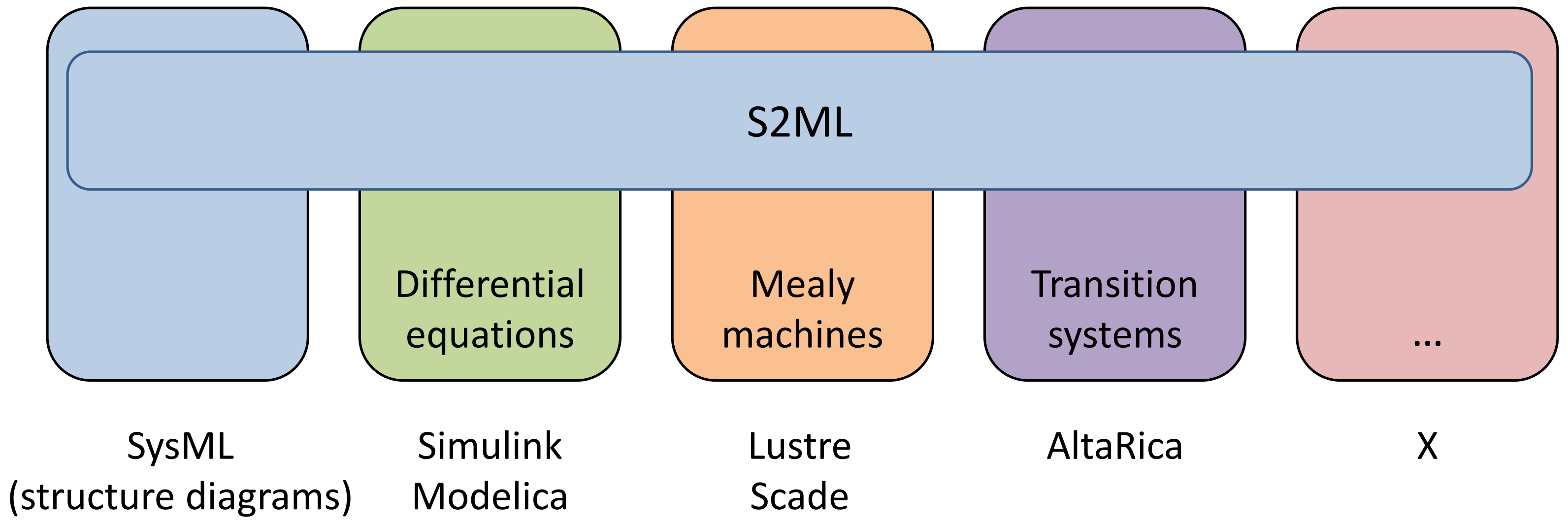
Inheritance



Is-a

The S2ML+X Promise

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

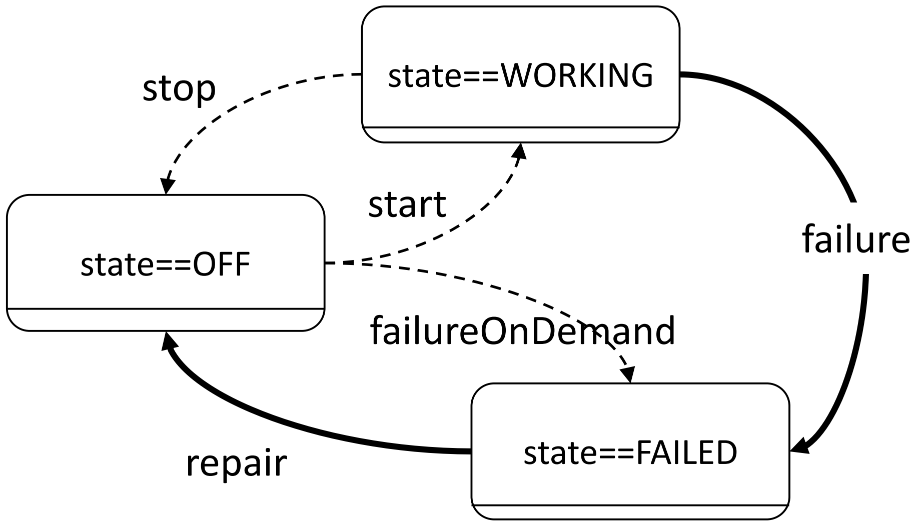


- The **structure of models** reflects the **structure of the system**, even though to a **limited extent.**
- **Structuring** helps to design, to debug, to share, to maintain and to align heterogeneous models.

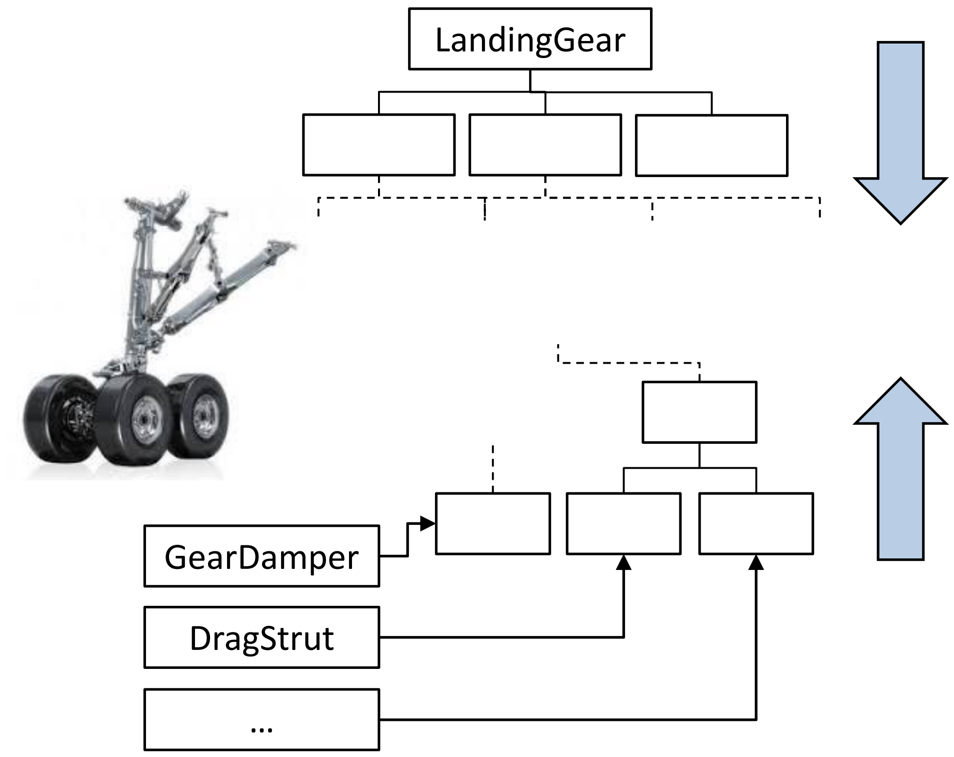
AltaRica 3.0 (S2ML + Guarded Transitions Systems)

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.



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
```

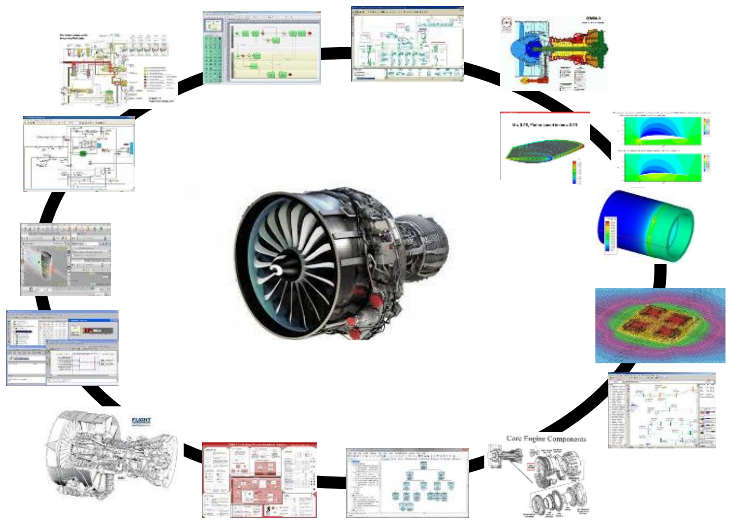
Complex models can be built using **libraries** of **reusable modeling components** and **modeling patterns**.

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**.



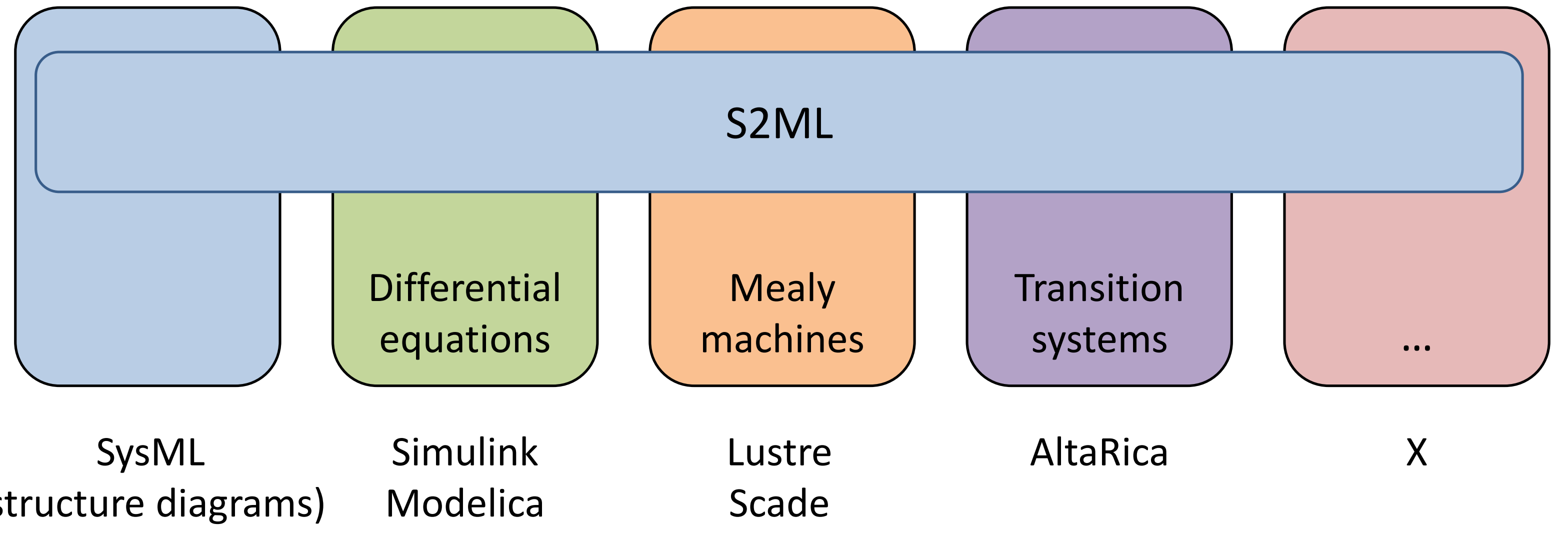
The **diversity** of models is **irreducible.**

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.

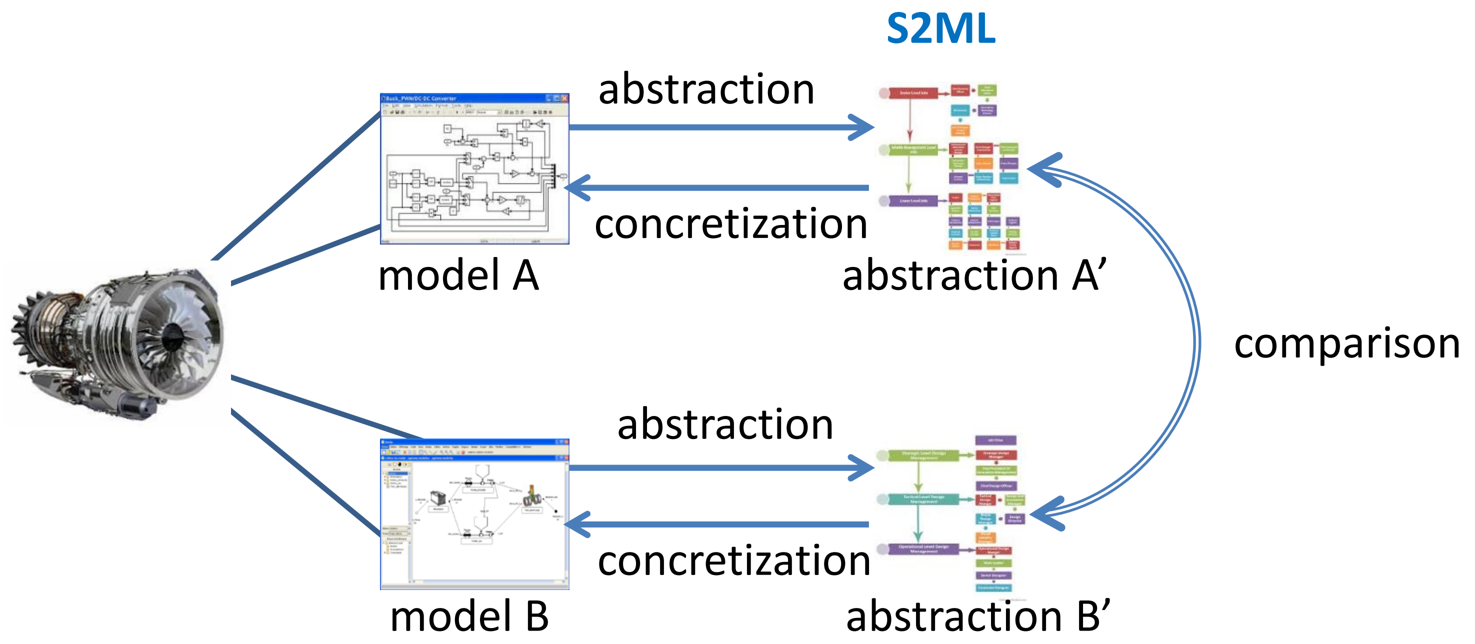
As the **behavioral part** of models is **purpose-dependent**, the main and most often the only way to align models is to compare their **structure**.



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**

Summary

- "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**.

Conclusion

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**.