Hong Kong City University 16/10/2019

Some Recent Developments on 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

&

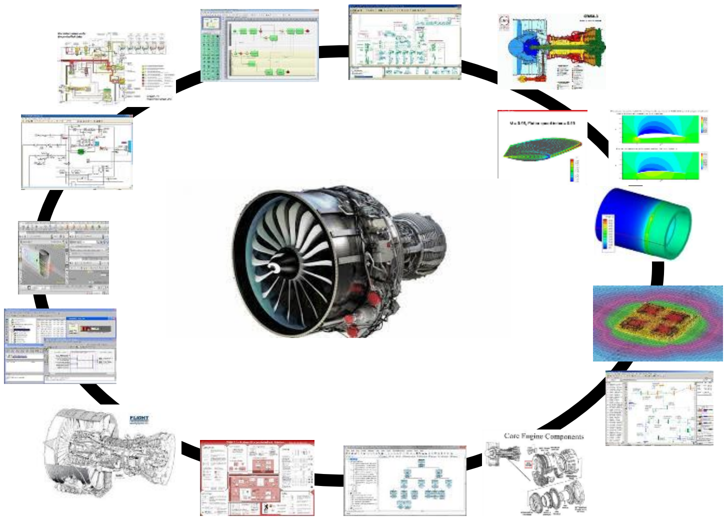
Chair Blériot-Fabre
CentraleSupélec/SAFRAN
Paris, France

Agenda

- Introduction
- Behavioral Models
- Experiments *in Silico* and their Complexity
- Reuse Modeling Components and Patterns
- Model Synchronization
- Conclusion

Model-Driven Engineering

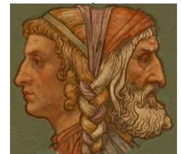
We entered in the era of **model-based systems engineering**: models are the only way to master the steadily increasing **complexity** of technical and socio-technical systems.



complexity → simplexity

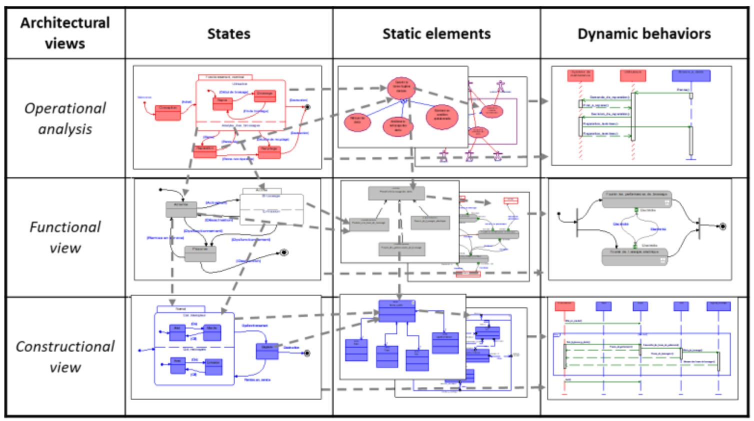
Models must be taken seriously and considered as **first-class citizens**.
We need to establish the **scientific foundations** of model-driven engineering.

Disciplines



System Architecture

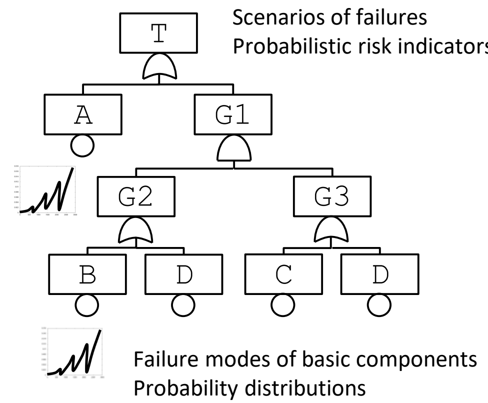
What the system should do?
What the system should be?



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

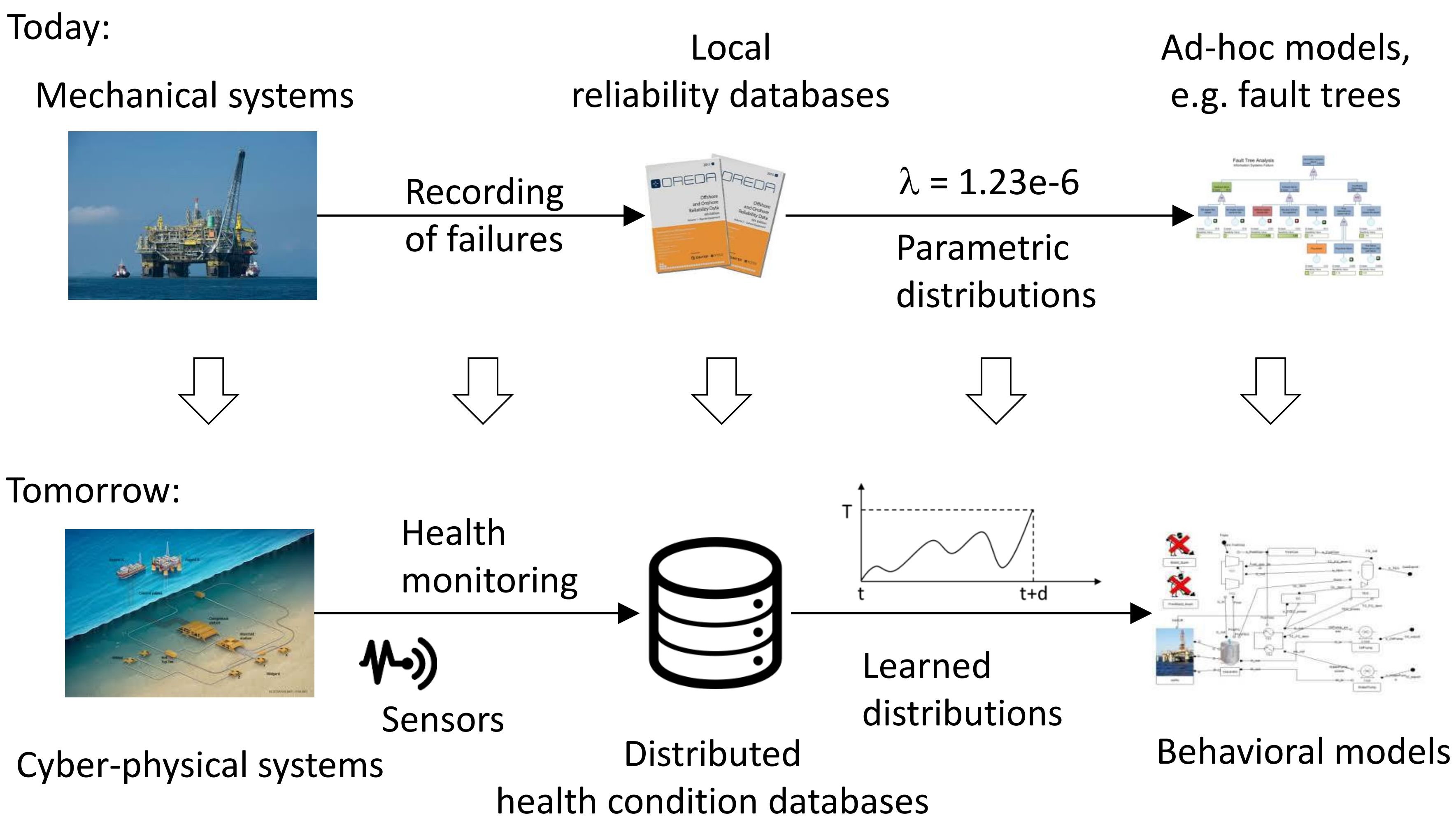
Reliability Engineering

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.

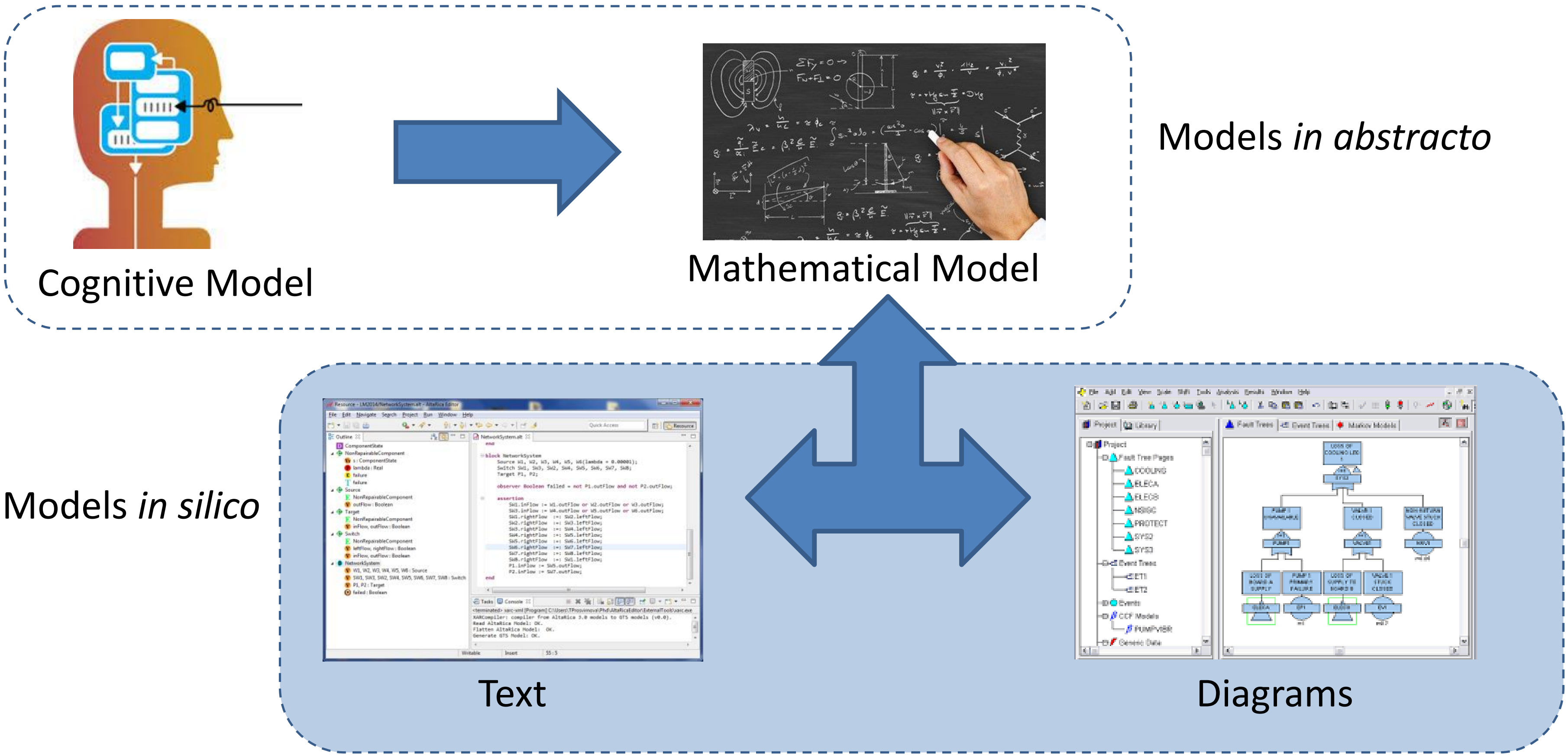
(R)evolution in Reliability Engineering



Agenda

- Introduction
- Behavioral Models
- Experiments *in Silico* and their Complexity
- Reuse Modeling Components and Patterns
- Model Synchronization
- Conclusion

Behavioral Models of Technical Systems



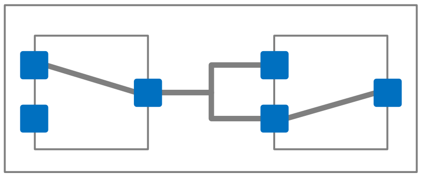
Characteristics of Behavioral Models

- Models are **well-defined mathematical objects** written in a **well-defined syntax**.
  - (More or less standardized) notations are not models.
  - Graphical/diagrammatic representations (of models) are not models.

- **Behaviors + Structures = Models**
  - 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.

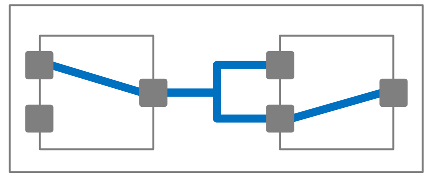
Ontology/Meta-Model of Behavioral Models

Port



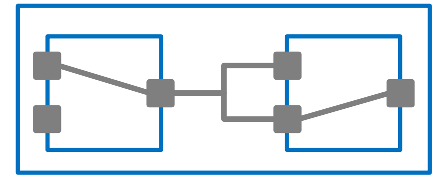
Variable, event…

Connection



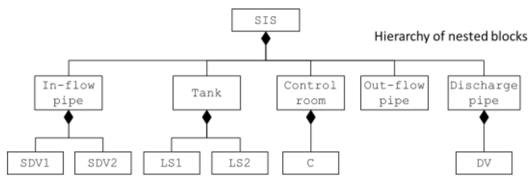
Equation, transition…

Container



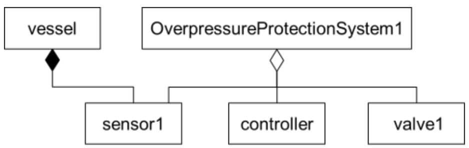
Model, component…

Composition



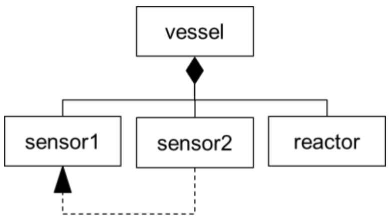
Is-part-of

Aggregation

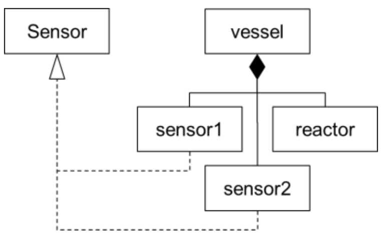


Uses

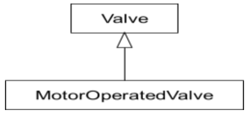
Prototype/Cloning



Class/Instantiation



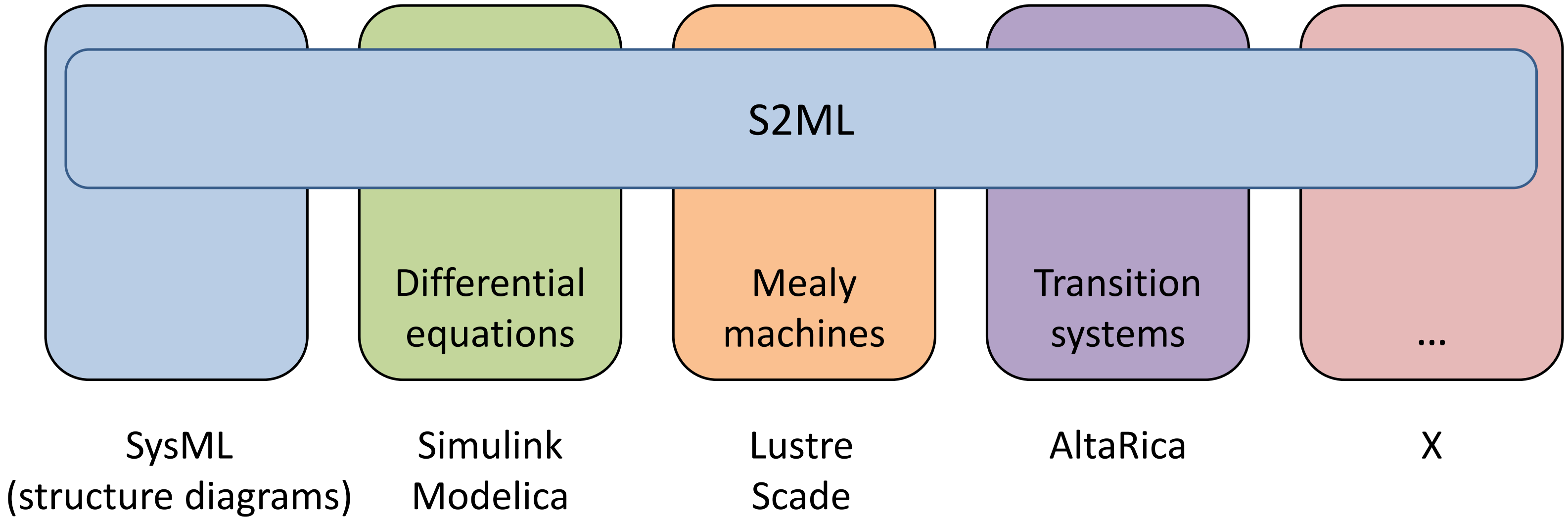
Inheritance



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.

Models as Scripts

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

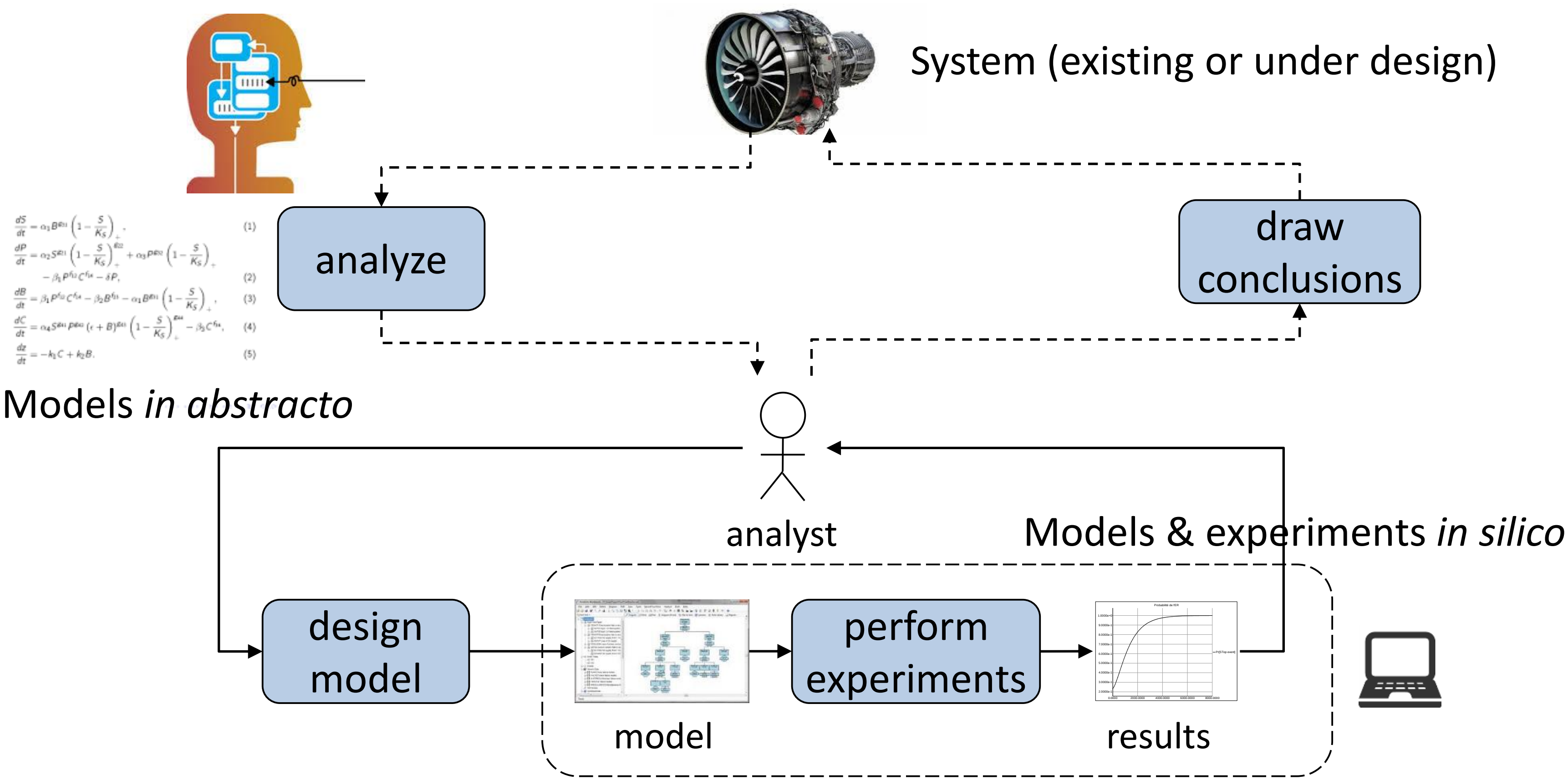
```
domain WF {WORKING, FAILED} WORKING<FAILED;

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

- Introduction
- Behavioral Models
- Experiments *in Silico* and their Complexity
- Reuse Modeling Components and Patterns
- Model Synchronization
- Conclusion

Experiments in Silico



A model results always of a **tradeoff** between the **accuracy of the description** and the **computational cost** of Experiments in Silico.

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
- …

Process Algebras
- Agent-based models
- Process algebras
- Python/Java/C++
- …

Expressive power →

States | States + transitions | Deformable systems

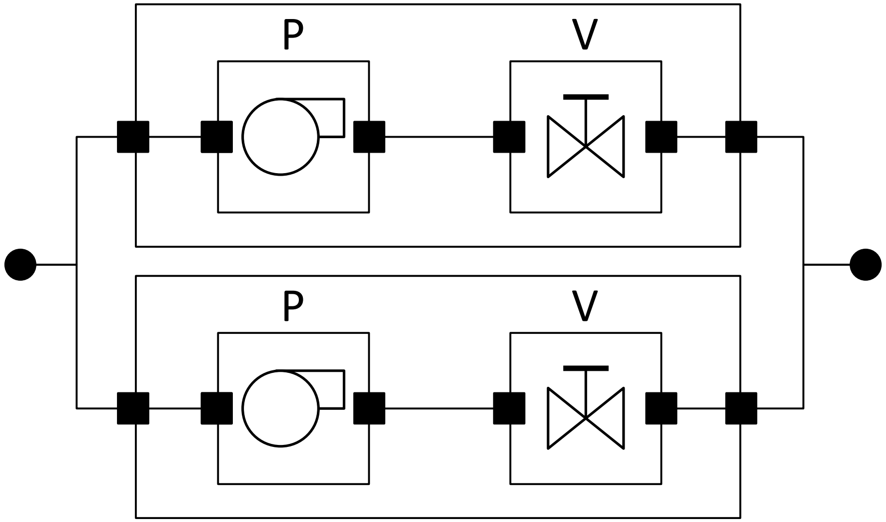
Complexity of assessments →

#P-hard but reasonable polynomial approximation | PSPACE-hard | Undecidable

Difficulty to design, to validate and to maintain models →

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**

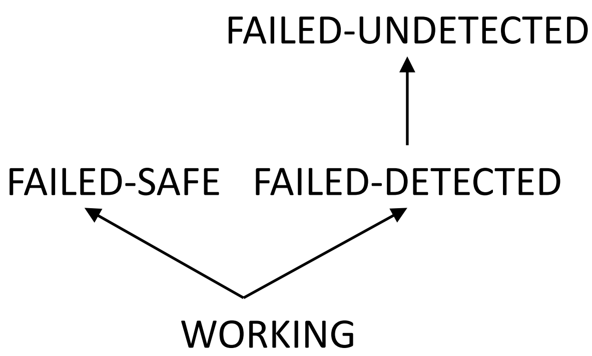



```
class Pump
    extends Component
    …
end

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**.



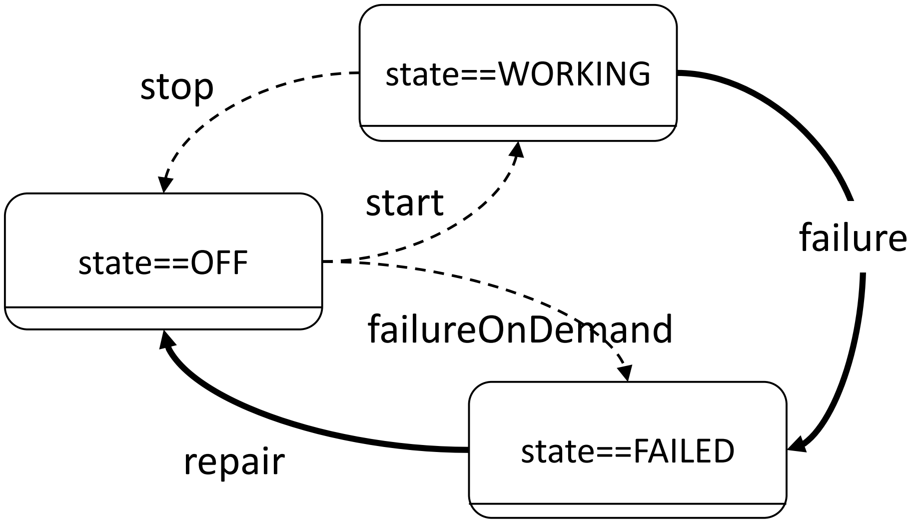
||	W	Fs	Fd	Fu
W	W	W	W	W
Fs	W	Fs	Fs	Fs
Fd	W	Fs	Fd	Fd
Fu	W	Fs	Fd	Fu

```
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.



OpenAltaRica SYSTEM SAFETY ANALYSIS TECHNOLOGY | SystemX INSTITUT DE RECHERCHE TECHNOLOGIQUE

SAFRAN

THALES

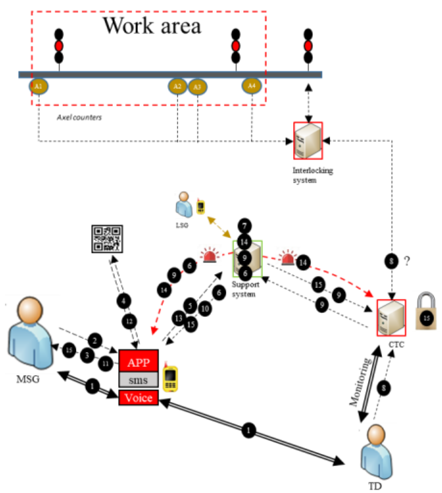
AIRBUS

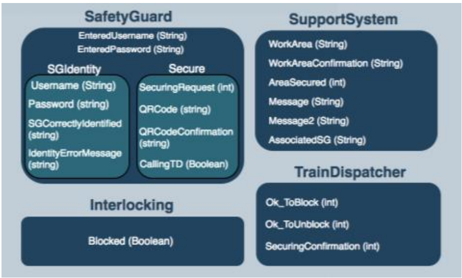
Scola (S2ML + Process Algebra)

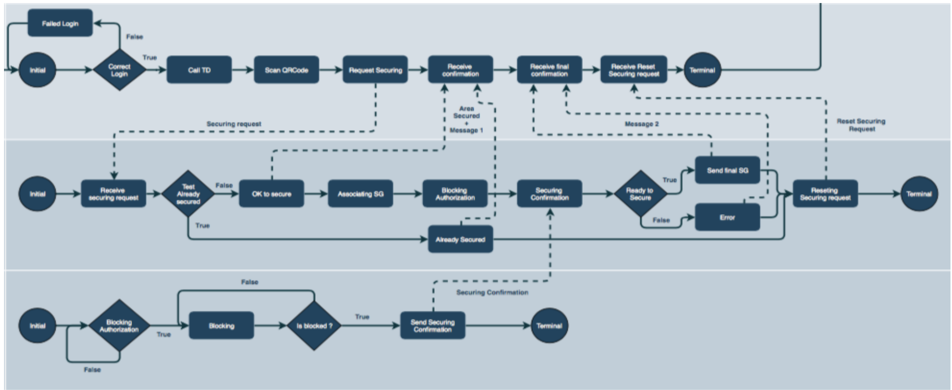
Scenario-oriented modeling methodology

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

BANE NOR





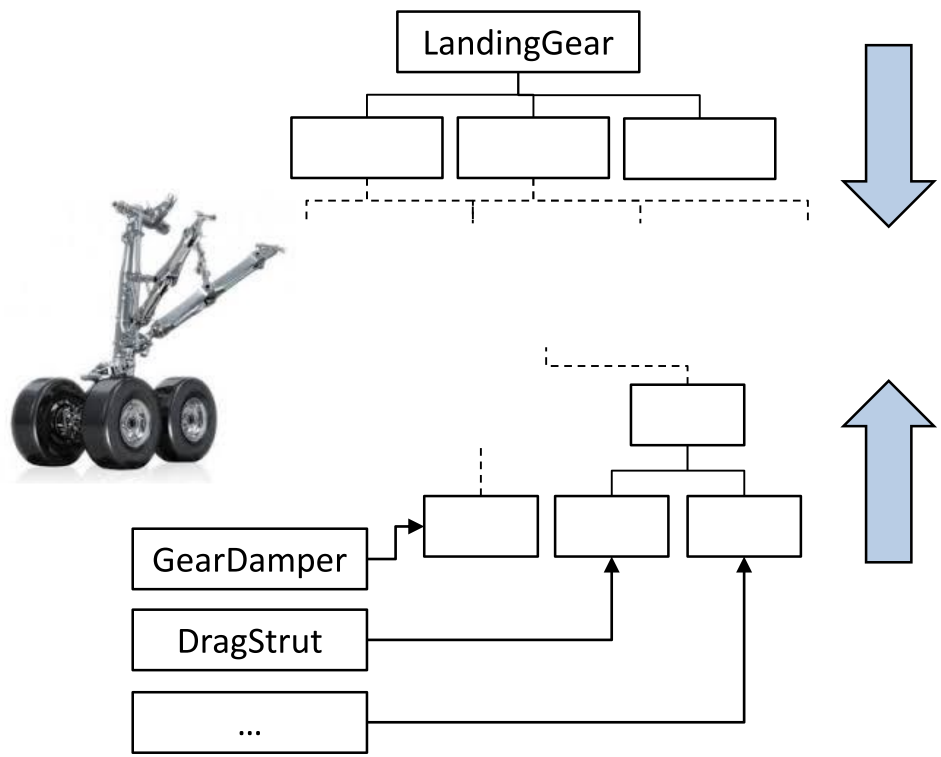


Agenda

- Introduction
- Behavioral Models
- Experiments *in Silico* and their Complexity
- Reuse Modeling Components and Patterns
- Model Synchronization
- Conclusion

Modeling Approaches and their Consequences on Reuse

Reuse of modeling elements from models to models is the key modeling **knowledge capitalization** which is itself the key of the modeling **efficiency**.



- 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

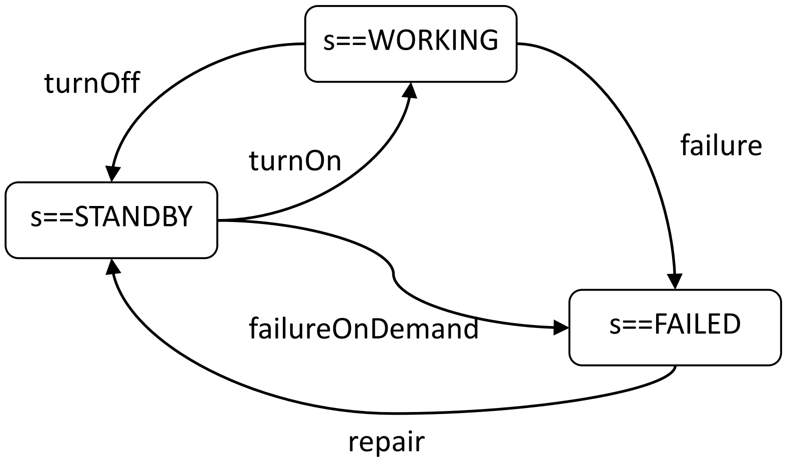
Multiphysics simulation

Reuse of Modeling Components

In bottom-up, object-oriented modeling approach, modeling efficiency relies on the design of generic and domain specific libraries of **on-the-shelf, reusable modeling components**.

Standby component (AltaRica)

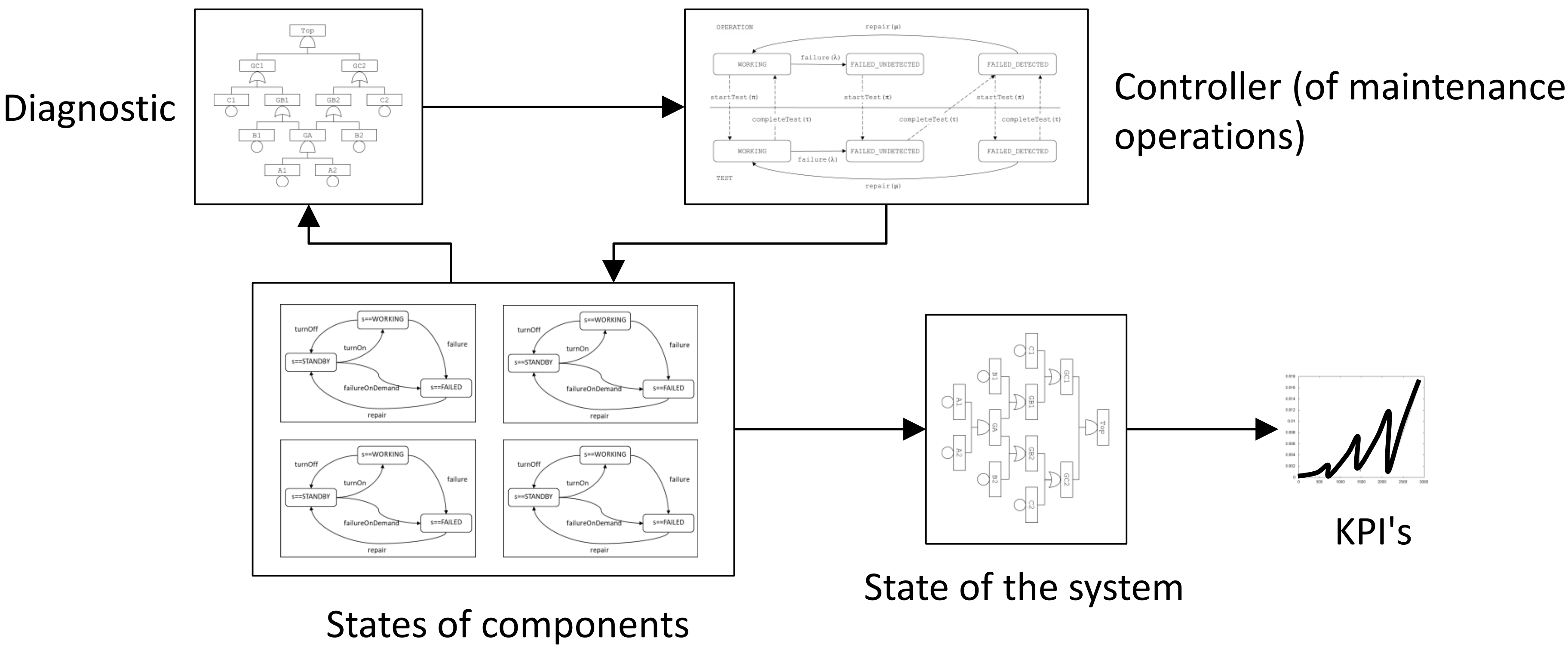
```
domain componentState = { STANDBY, WORKING, FAILED}

block SpareComponent
    componentState s (init = WORKING);
    event turnOn, turnOff, failureOnDemand,
          failure, repair;
transition
    turnOn: s == STANDBY -> s := WORKING;
    failureOnDemand: s == STANDBY -> s := FAILED;
    turnOff: s == WORKING -> s := STANDBY;
    failure:  s == WORKING -> s := FAILED;
    repair:  s == FAILED -> s := STANDBY;
end
```



Reuse of Modeling Patterns

In top-down, prototype-oriented modeling approach, modeling efficiency relies on the design of generic and domain specific libraries of **on-the-shelf, reusable modeling patterns**.

Pattern for condition-based maintenance (AltaRica)

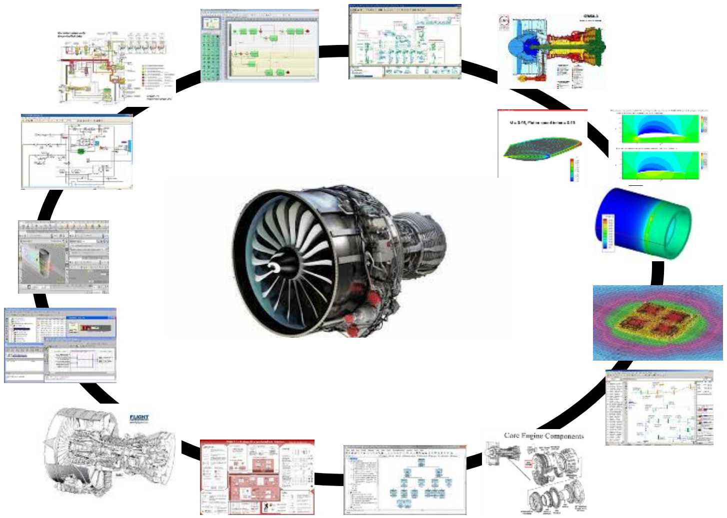


Agenda

- Introduction
- Behavioral Models
- Experiments *in Silico* and their Complexity
- Reuse Modeling Components and Patterns
- Model Synchronization
- Conclusion

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**.



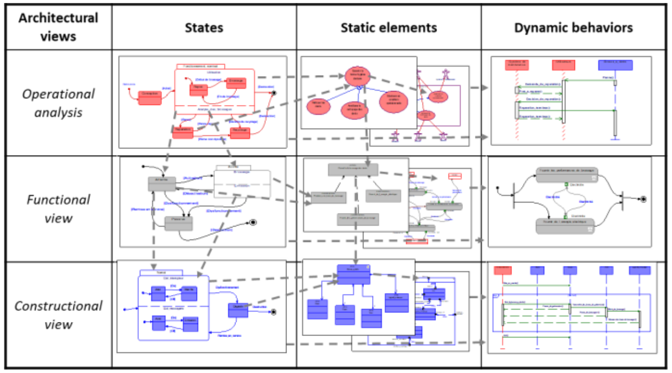
complexity → simplexity

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

Pragmatic versus Formal Models

System Architecture

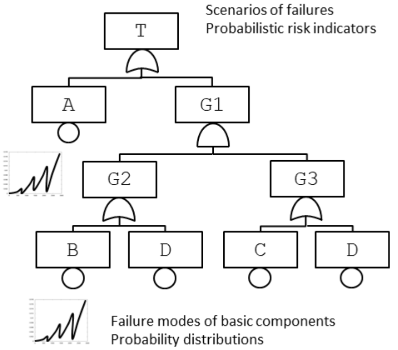
Models to communicate
amongst stakeholders



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.

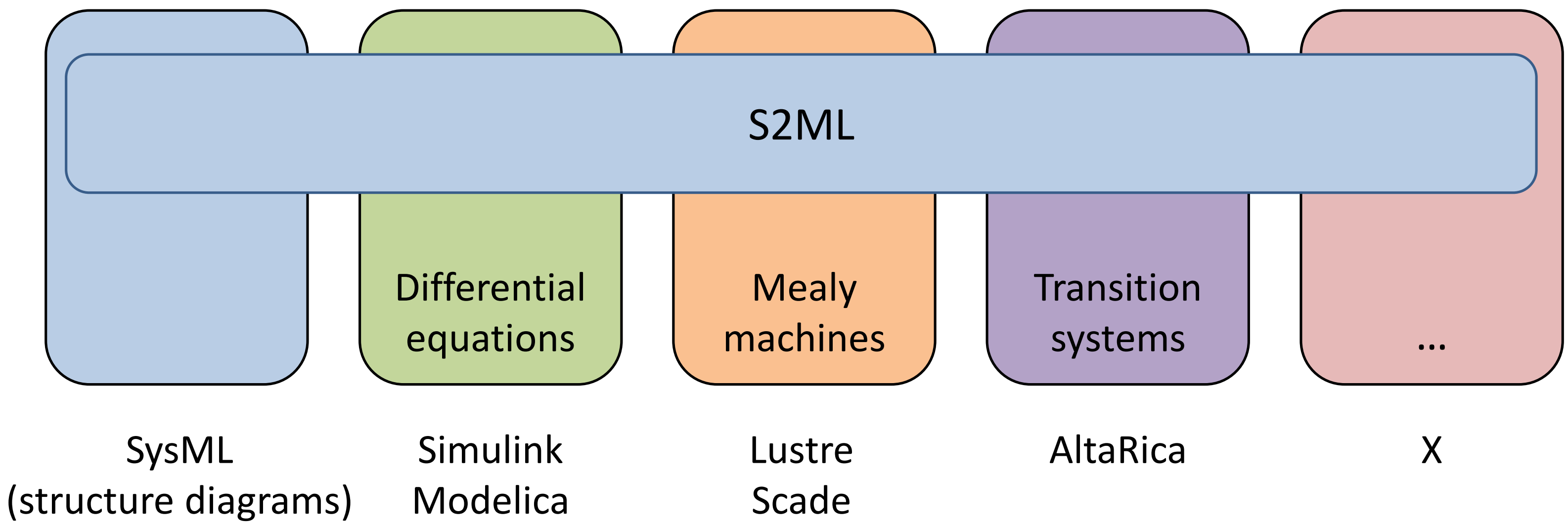
Epistemic gap

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**.

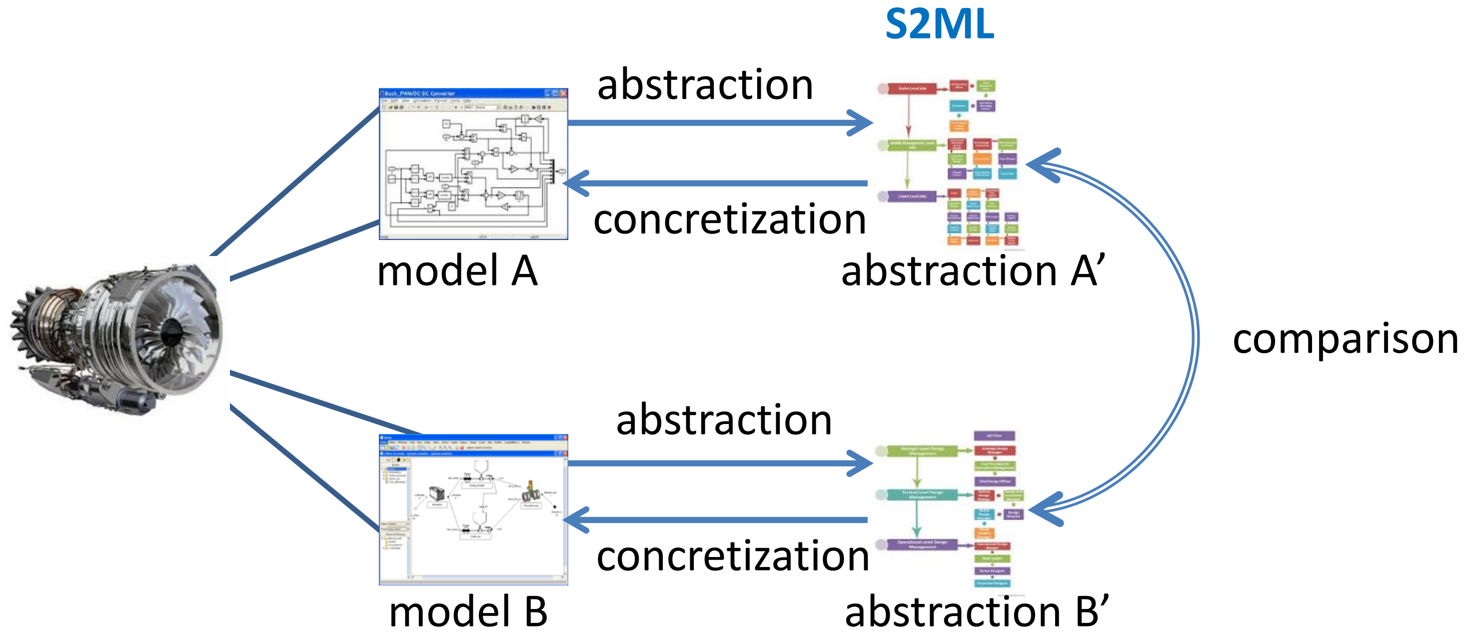
The question is how to ensure that they are "speaking" about the **same system**, i.e. to **align** them.

As the **behavioral part** of models is **purpose-dependent**, the main way to compare models is to compare their **structure**.



Model Synchronization

Abstraction + Comparison = Synchronization



How to agree on disagreements?

Agenda

- Introduction
- Behavioral Models
- Experiments *in Silico* and their Complexity
- Reuse Modeling Components and Patterns
- Model Synchronization
- Conclusion

Wrap-Up & Conclusion

- "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.
- **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**.
- 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 **system thinking**, and
  - With skills and competences in **specific application domains**.