NTNU – Trondheim
Norwegian University of
Science and Technology

Model-Based Safety Assessment with AltaRica 3.0

Towards the next generation of methods, concepts and tools for probabilistic safety assessment
(a computer scientist point of view)

Prof. Antoine Rauzy
Department of Mechanical and Production Engineering
NTNU, Trondheim, Norway
Antoine.Rauzy@ntnu.no

Head of chair Blériot-Fabre
Department of Industrial Engineering
CentraleSupélec, Paris, France

AltaRica 3.0

```
domain State {WORKING, HIDDEN_FAILURE, DETECTED_FAILURE}
domain Mode {OPERATION, INSPECTION}

block PeriodicallyInspectedComponent
    State state(init=WORKING);
    Mode mode(init=OPERATON};
    event failure(delay=exponential(lambda));
    event repair(delay=exponential(mu));
    event startInspection(delay=Dirac(tau));
    event completeInspection(delay=Dirac(pi));
    parameter Real lambda = 1.0e-3;
    parameter Real mu = 0.1;
    parameter Real tau = 720;
    parameter Real pi = 12;
    transition
        failure: state==WORKING -> state:=HIDDEN_FAILURE;
        repair: state==DETECTED_FAILED -> state:=WORKING;
        startInspection: mode==OPERATION -> mode:=INSPECTION;
        completeInspection: model==INSPECTION -> {
            mode:=OPERATION;
            state := if state==WORKING then WORKING else DETECTED_FAILED;
            }
end
```

Agenda

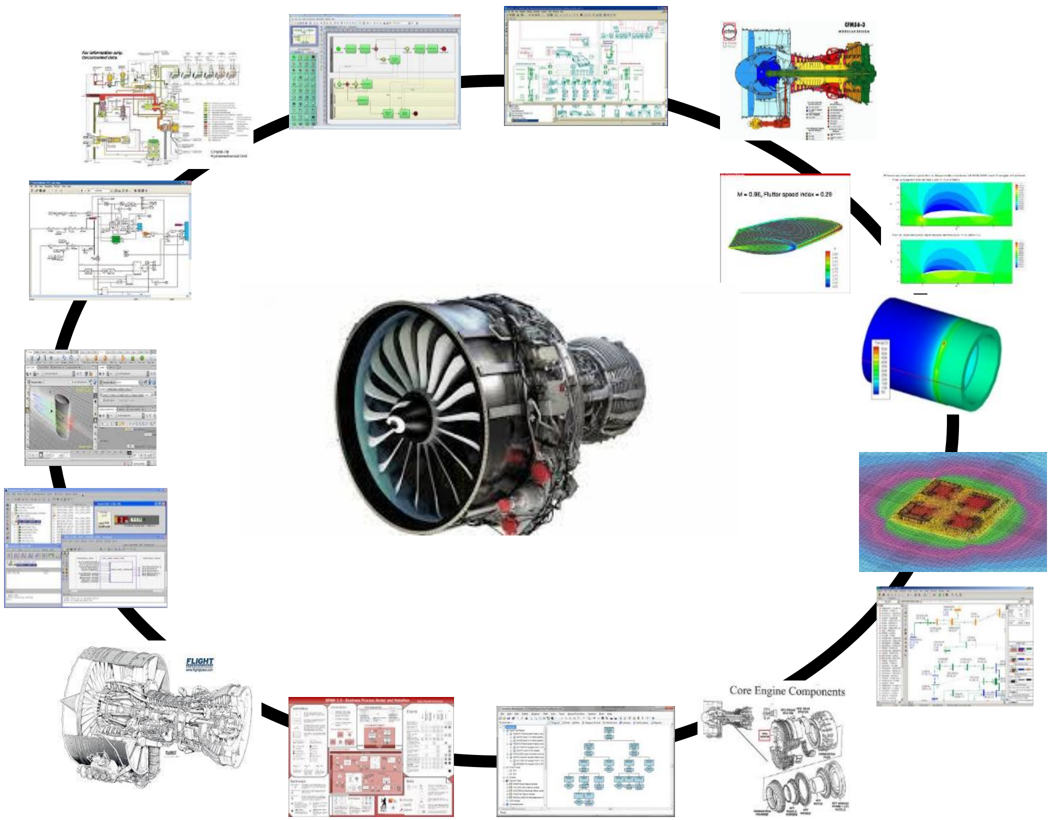
- Rational
- Theses
- Guarded Transitions Systems
- System Structure Modeling Language
- On going and future works

Agenda

- Rational
- Theses
- Guarded Transitions Systems
- System Structure Modeling Language
- On going and future works

Model-Based Systems Engineering

How many modeling tools, how many models to design and to operate an aircraft engine?



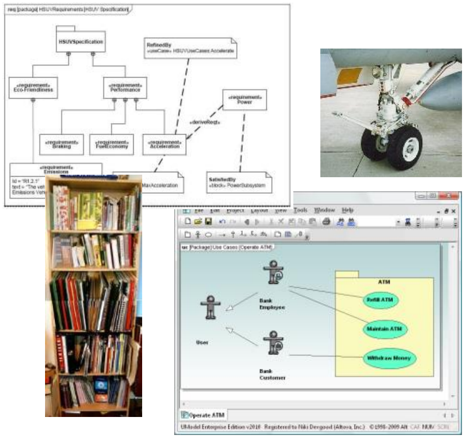
The emerging science of complex systems is the science of models

Today's Challenges of Probabilistic Safety Assessment

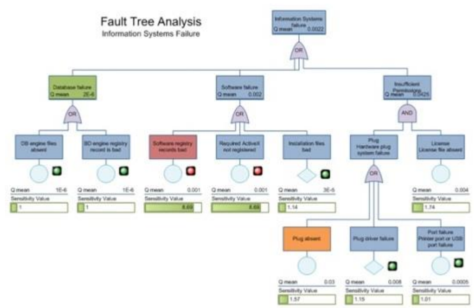
- How to deal with **mechatronics** and **cyber-physical systems** (control mechanisms, reconfigurations…)?
- How to **manage versions and configurations** of models through the life-cycle of systems?
- How to **better integrate** probabilistic risk/safety assessment models with models designed by other engineering disciplines, especially those designed by systems architects.

Issues with “Classical” Safety Models

Systems specifications



Models



Fault Trees, Event Trees, Markov Chains, Stochastic Petri Nets…

Calculations

- Failure scenarios
- Failure probabilities

Classical modeling formalisms used for safety analyses lack of expressive power and/or of structure.

→ **Distance** between **systems specifications** and **models**;

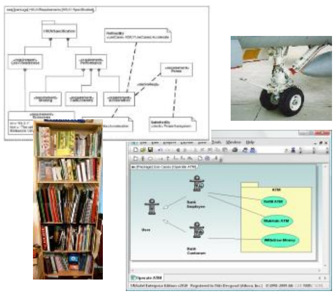
→ Models are **hard to design** and even **harder** to **share with stakeholders** and to **maintain** throughout the **life-cycle** of systems.

→ Often too **conservative** approximations

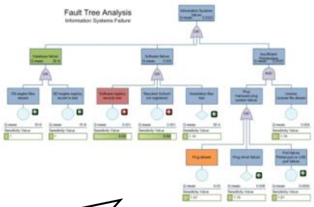
The Model-Based Safety Assessment promise

Reducing the gap between systems specifications and probabilistic safety assessments

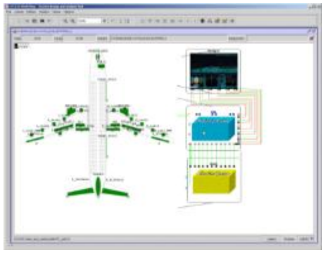
Systems specifications



Models



Calculations



Calculations

High level models
reflecting systems architecture

Agenda

- Rational
- Theses
- Guarded Transitions Systems
- System Structure Modeling Language
- On going and future works

“In philosophy and rhetoric, a thesis is a statement that can be summarized with a simple sentence, but that is supported by an organized set of hypotheses, arguments and conclusions. It is the position of an author, a school, a doctrine or a movement on a given subject.”

Wikipedia

Thesis 1

Models should not be confused with their graphical representations

Meaning and practical consequences:

- A model is a **mathematical object**.
- A **graphical representation** is a view on the model, very useful for **communication**, but…
- **Complex models cannot be fully represented graphically**.

- Moreover, which **several alternative graphical representations** can be proposed for the same model.

In a word, we have to think first to mathematical objects, then to their possible graphical representations

Thesis 2

A probabilistic safety assessment model results always of a tradeoff between the accuracy of the description of the system under study and the computational cost of calculations of risk/safety indicators

Meaning and practical consequences:

- Calculations of probabilistic indicators are **provably computationally hard** (#P-hard).
- Assessment algorithms perform (unwarranted) **approximations**.
- The more complex the model, the coarser the approximations.

- Adding **more expressive power** is interesting only if it can be done at **low computational cost**.
- Moreover, the more complex the model, the harder its **validation**.

Thesis 3

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"

AltaRica 3.0

Behaviors	+	Structures	=	Models
GTS	**+**	**S2ML**	**=**	**AltaRica 3.0**

GTS: Guarded Transitions Systems

Generalization of state/transitions formalisms such as (multiphase) Markov chains and stochastic Petri nets

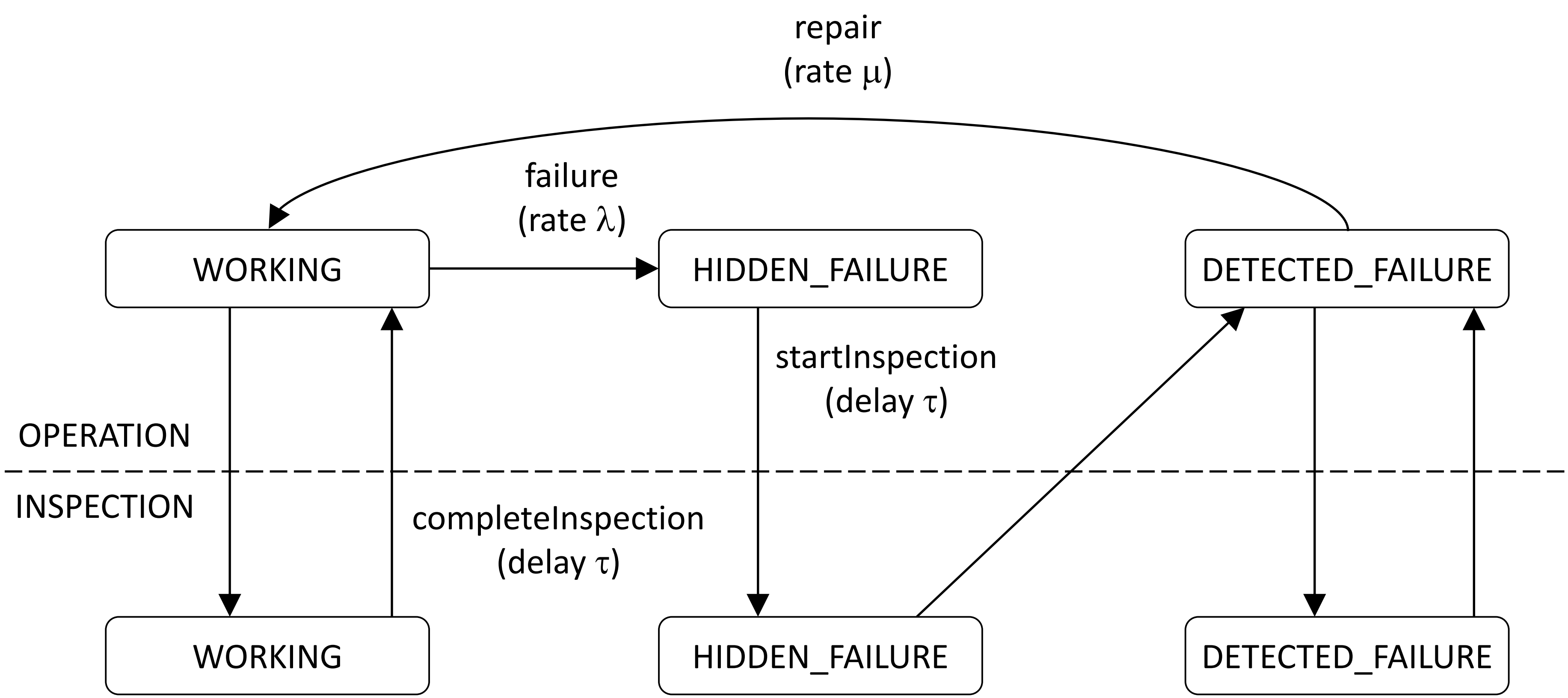
S2ML: System Structure Modeling Language

Sets of structuring mechanisms stemmed from object-oriented programming

Agenda

- Rational
- Theses
- Guarded Transitions Systems
- System Structure Modeling Language
- On going and future works

State, Events and Transitions



Model for a periodically inspected component

Stochastic Discrete Event Systems

```
domain State {WORKING, HIDDEN_FAILURE, DETECTED_FAILURE}
domain Mode {OPERATION, INSPECTION}

block PeriodicallyInspectedComponent
    State state(init=WORKING);
    Mode mode(init=OPERATON};
    event failure(delay=exponential(lambda));
    event repair(delay=exponential(mu));
    event startInspection(delay=Dirac(tau));
    event completeInspection(delay=Dirac(pi));
    parameter Real lambda = 1.0e-3;
    parameter Real mu = 0.1;
    parameter Real tau = 720;
    parameter Real pi = 12;
    transition
        failure: state==WORKING -> state:=HIDDEN_FAILURE;
        repair: state==DETECTED_FAILED -> state:=WORKING;
        startInspection: mode==OPERATION -> mode:=INSPECTION;
        completeInspection: model==INSPECTION -> {
            mode:=OPERATION;
            state := if state==WORKING then WORKING else DETECTED_FAILED;
            }
end
```

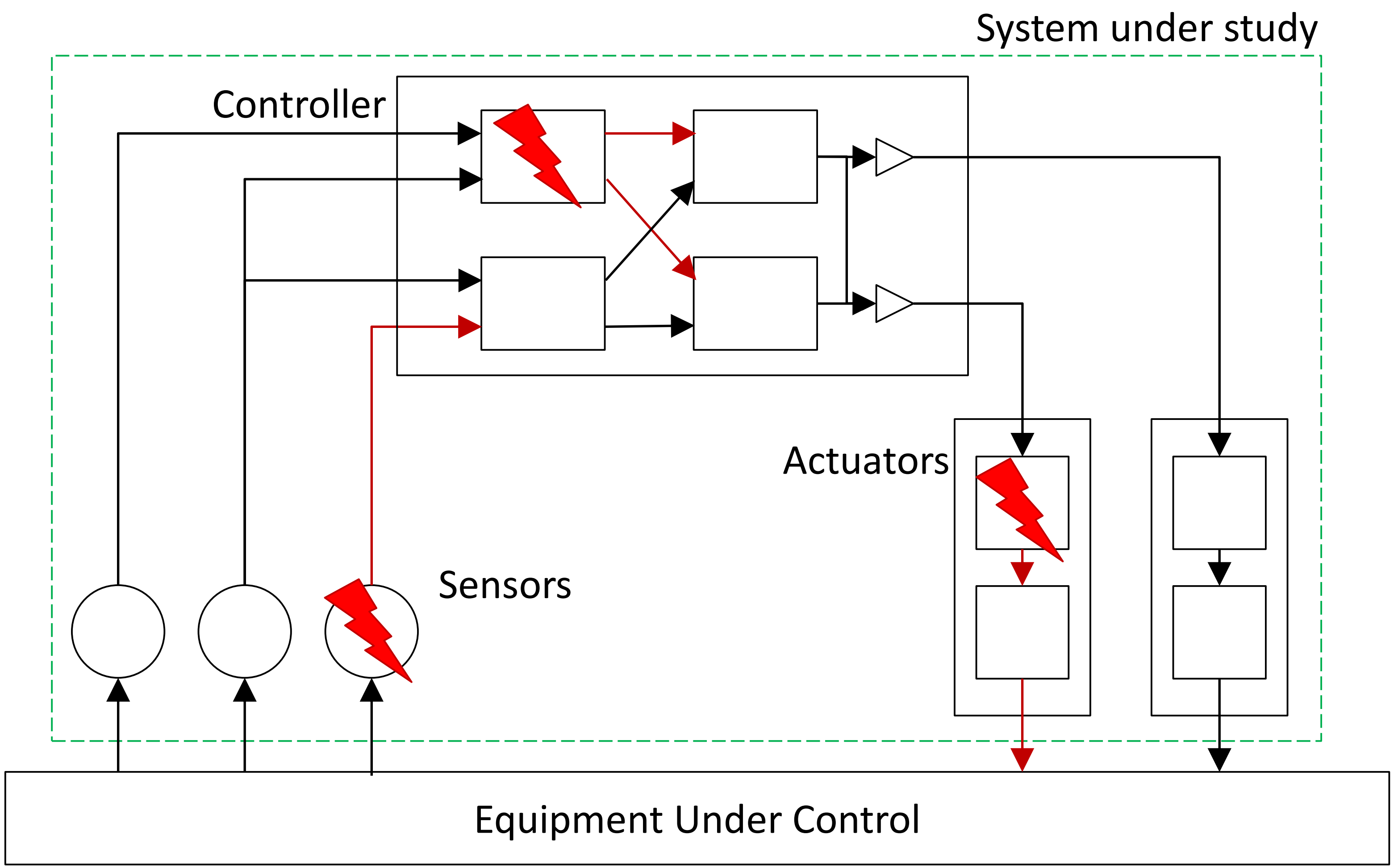
State variables

Events associated with stochastic or deterministic delays

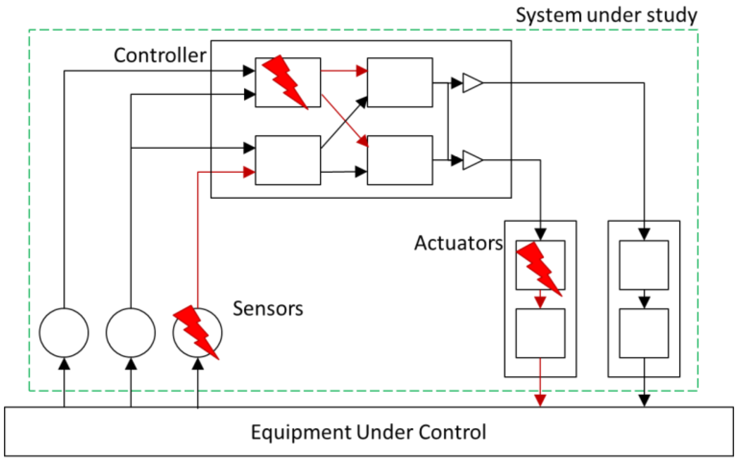
Parameters

Transitions

Flow propagation



Flow propagation

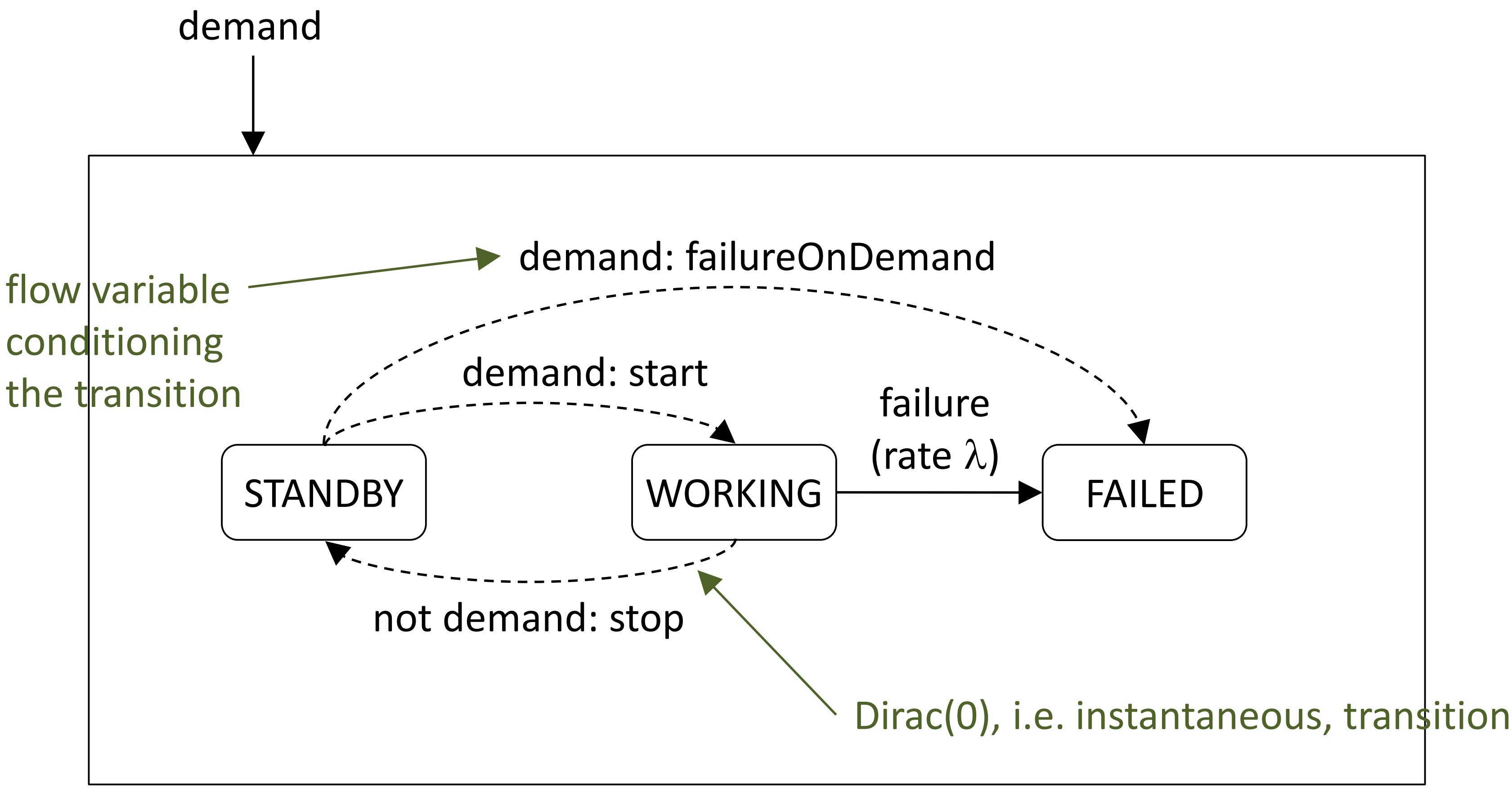



```
block System
    …
    block Sensor3
        …
        Boolean input, output (reset=false),
        …
        assertion
            output := state==WORKING and input;
            …
    end
    …
    block Controller
        block AcquisitionModule1
            …
            Boolean input1, input2 (reset=false);
            …
        end
        …
    end
    assertion
        …
        Controller.AcquisitionModule1.input1 := Sensor3.output;
        …
end
```

Flow variables

Assertions
(transfer functions)

Reusable Modeling Components



Model for a cold spare component

Synchronization

It is possible to fire several events simultaneously. This is called a **synchronization**.

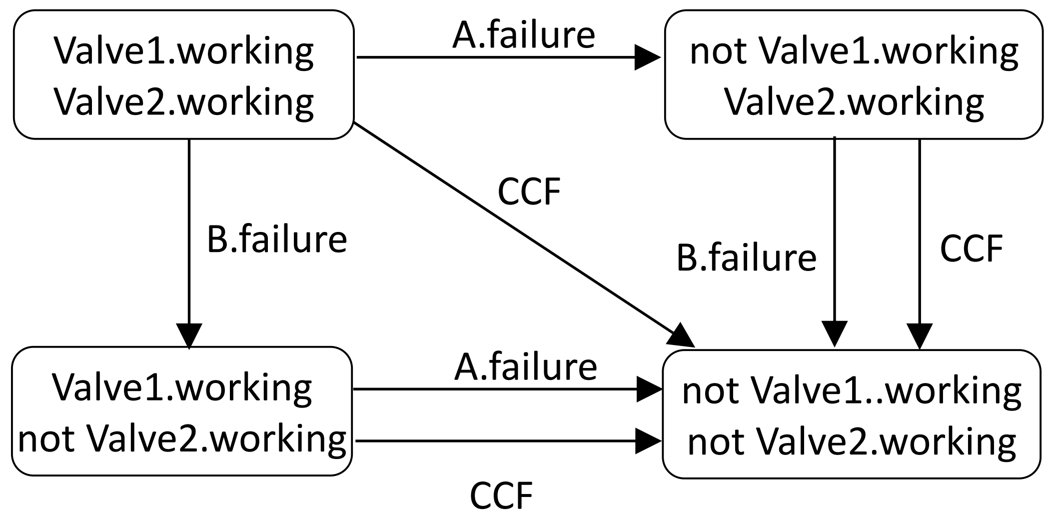
```
block System
   block Valve1
      Boolean working (init = true);
      event failure;
   transition
      failure: working -> working := false;
      repair: not working -> working := true;
   end

   block Valve2
      ...
   end

   event CCF;
   transition
         CCF: ?Valve1.failure & ?Valve2.failure;
end
```

synchronization

Model with a common cause failure



Formal Definition

A **Guarded Transition Systems** is a quintuple $\langle V,E,T,A,\iota\rangle$, where:

- V is a set of **variables**. V is the disjoint union of the set S of **state variables** and the set F of **flow variables**: $V=S\uplus F$.
- E is a set of **events**.
- T is a set of **transitions**, i.e. of triples $\langle e,G,P\rangle$, where e is an event of E, G is a Boolean expression built on variables of V and P is an instruction built on variables of V. For the sake of the clarity, we shall write a transition $\langle e,G,P\rangle$ as $e: G \rightarrow P$.
- A is an **assertion**, i.e. an **instruction** built on variables of V.
- ι is an assignment of variables of V, so-called initial or **default assignment**.

The set of **instructions** is the smallest set such that.

- "skip" is an instruction.
- If v is a variable and E is an expression, then "v := E" is an instruction.
- If C is a (Boolean) expression, I is an instruction, then "if C then I" is an instruction.
- If I_1 and I_2 are instructions, then so is "I_1 ; I_2".

Formal (Denotational and Operational) Semantics

S0: $\dfrac{}{\langle skip, \sigma, \tau \rangle \longrightarrow \tau}$

S1: $\dfrac{\tau(v) = ?, \;\; \sigma(E) \in dom(v)}{\langle v \coloneqq E, \sigma, \tau \rangle \longrightarrow \tau[\sigma(E)/v]}$

S2: $\dfrac{\tau(v) = \; \sigma(E), \;\; \sigma(E) \in dom(v)}{\langle v \coloneqq E, \sigma, \tau \rangle \longrightarrow \tau}$

S3: $\dfrac{\sigma(E) = ERROR \;\; or \;\; \sigma(E) \notin dom(v) \;\; or \;\; \tau(v) \neq ?, \;\; \sigma(E) \neq \tau(v)}{\langle v \coloneqq E, \sigma, \tau \rangle \longrightarrow ERROR}$

S4: $\dfrac{\sigma(C) = TRUE}{\langle if \;\, C \; then \; I, \sigma, \tau \rangle \longrightarrow \langle I, \sigma, \tau \rangle}$

S5: $\dfrac{\sigma(C) = FALSE}{\langle if \;\, C \; then \; I, \sigma, \tau \rangle \longrightarrow \tau}$

S6: $\dfrac{\sigma(C) = ERROR}{\langle if \;\, C \; then \; I, \sigma, \tau \rangle \longrightarrow ERROR}$

S7: $\dfrac{\langle I_1, \sigma, \tau \rangle \longrightarrow \tau'}{\langle I_1; I_2, \sigma, \tau \rangle \longrightarrow \langle I_2, \sigma, \tau' \rangle}$

S8: $\dfrac{\langle I_2, \sigma, \tau \rangle \longrightarrow \tau'}{\langle I_1; I_2, \sigma \rangle \longrightarrow \langle I_1, \sigma, \tau' \rangle}$

S9: $\dfrac{\langle I_1, \sigma, \tau \rangle \longrightarrow \langle I_1', \sigma, \tau' \rangle}{\langle I_1; I_2, \sigma, \tau \rangle \longrightarrow \langle I_1'; I_2, \sigma, \tau' \rangle}$

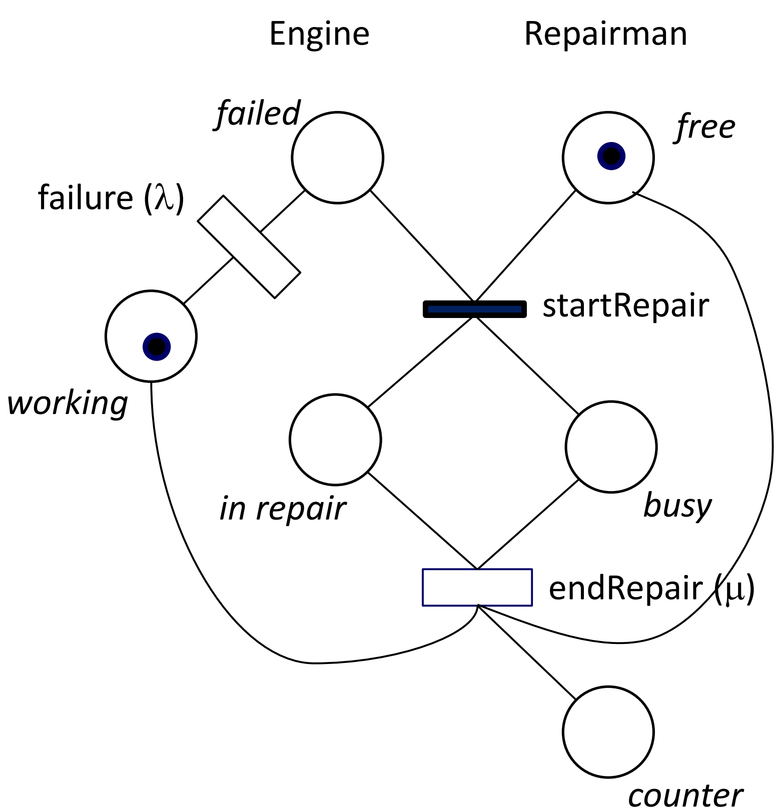
S10: $\dfrac{\langle I_2, \sigma, \tau \rangle \longrightarrow \langle I_2', \sigma, \tau' \rangle}{\langle I_1; I_2, \sigma \rangle \longrightarrow \langle I_1; I_2', \sigma, \tau' \rangle}$

S11: $\dfrac{\langle I_1, \sigma, \tau \rangle \longrightarrow ERROR}{\langle I_1; I_2, \sigma, \tau \rangle \longrightarrow ERROR}$

S12: $\dfrac{\langle I_2, \sigma, \tau \rangle \longrightarrow ERROR}{\langle I_1; I_2, \sigma, \tau \rangle \longrightarrow ERROR}$

Comparison with Existing Modeling Formalisms

Guarded transitions systems **generalize at no computational cost** existing modeling formalisms such as Markov chains, Stochastic Petri Nets…




```
domain EngineState = { WORKING, FAILED, IN_REPAIR }
domain RepairManState = { FREE, BUSY }

block MyNet
  EngineState engine (init = WORKING);
  RepairManState repairMan (init = FREE);
  Integer counter (init = 0);
  event failure (delay = exponential(lambda));
  event startRepair (delay = 0);
  event enRepair (delay = exponential(lmu));
  parameter Real lambda = 1.0e-3;
  parameter Real mu = 1.0e-1;
transition
  failure: engine==WORKING ->  engine := FAILED;
  startRepair: engine==FAILED and  repairMan==FREE -> {
     engine := IN_REPAIR; repairMan := BUSY; }
  endRepair: engine==IN_REPAIR and  repairMan==BUSY -> {
       engine := WORKING; repairMan := FREE;
       counter:= counter+1;  }
end
```

Agenda

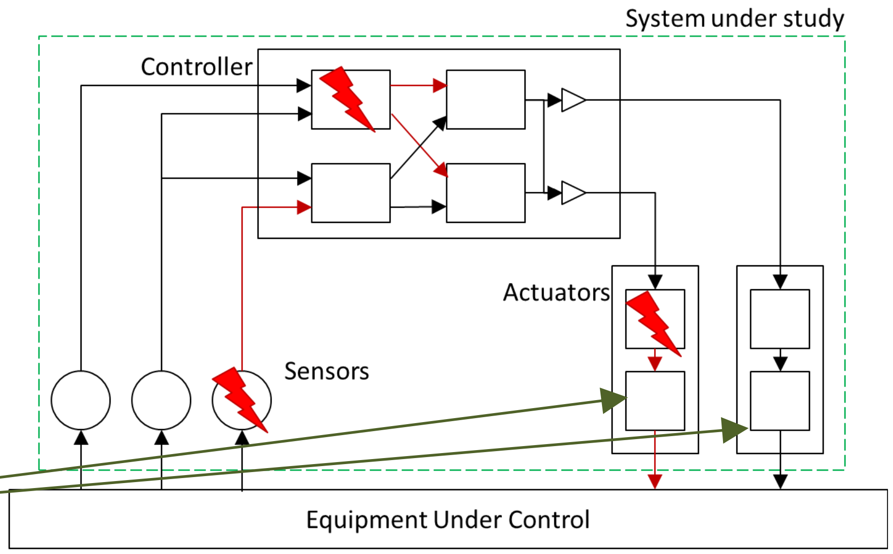
- Rational
- Theses
- Guarded Transitions Systems
- System Structure Modeling Language
- On going and future works

Object-Oriented Modeling

```
class PeriodicallyInspectedComponent
    State state(init=WORKING);
    Mode mode(init=OPERATION);
    event failure(exponential(lambda));
    …
end

class Valve
    inherits SpareComponent;
    …
end

block SafetyInstrumentedSystem
    Valve SDV1, SDV2;
    …
end
```



Fundamental objects and relations

S2ML gathers and organizes **fundamental concepts** of modeling languages.

Objects

Ports	variables: state, demand, events: failure…
Containers	block SDV1, class Pump…

Operational relations

Connection	failure: state==WORKING -> state:= FAILED;

Hierarchical relations

Composition	pump SDV1 **is-part-of** of system SIS
Aggregation	system SIS **uses** power-supply PW

Reuse relations

Instantiation	SDV1 **is-a-copy-of** on-the-shelf component Pump
Inheritance	Pump **is-a** PeriodicallyInspectedComponent
Cloning	train2 **is-a-copy-of** train1

Polymorphism

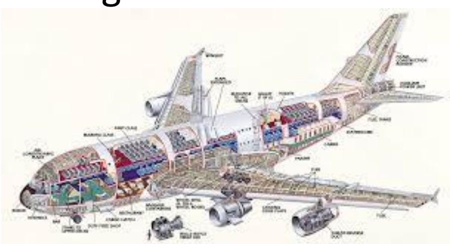
Agenda

- Rational
- Theses
- Guarded Transitions Systems
- System Structure Modeling Language
- On going and future works

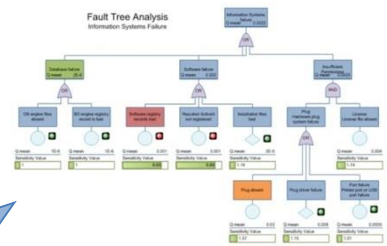
Model-Based Safety Assessment

Promise: AltaRica 3.0 help to **fill the gap** between **Systems** and **Models** and to **integrate probabilistic risk/safety assessment** with **systems architecture**.

System Specifications
Experience feedback
Regulations and Standards



Models

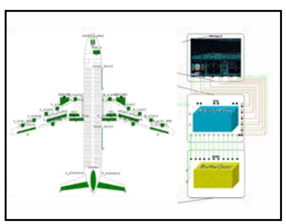


Calculations

- Failure Scenarii
- Reliability Indicators

Fault Trees, Event Trees, Markov Chains, Stochastic Petri Nets…

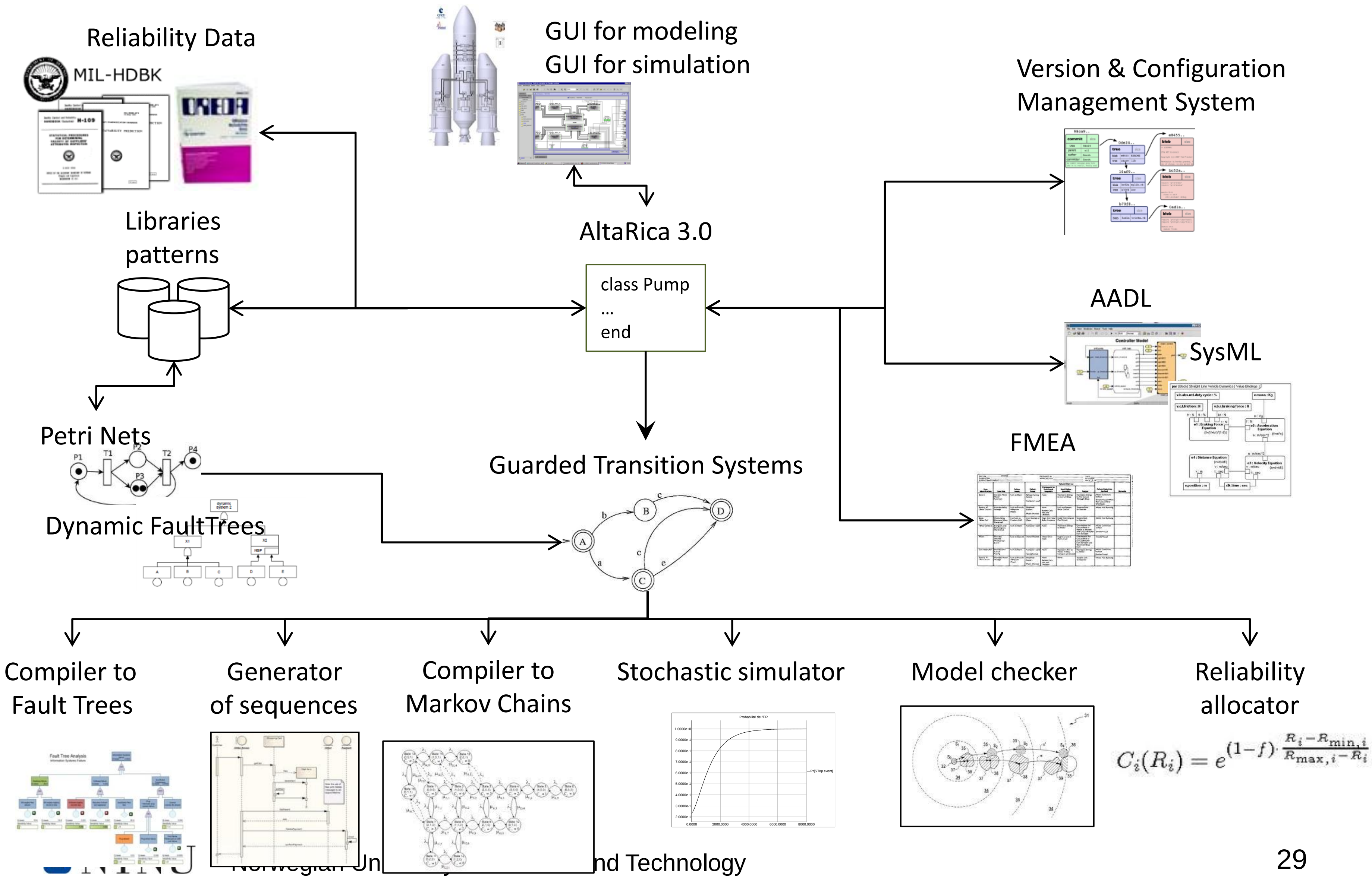
AltaRica



```
class Pump
 Boolean working (init = TRUE);
 event failure (delay = exponential(lambda));
 transitions
   failure: working -> working := FALSE;
end
```

- Make safety models closer to system specifications
- Design one model, calculate several safety goals

The AltaRica 3.0 project



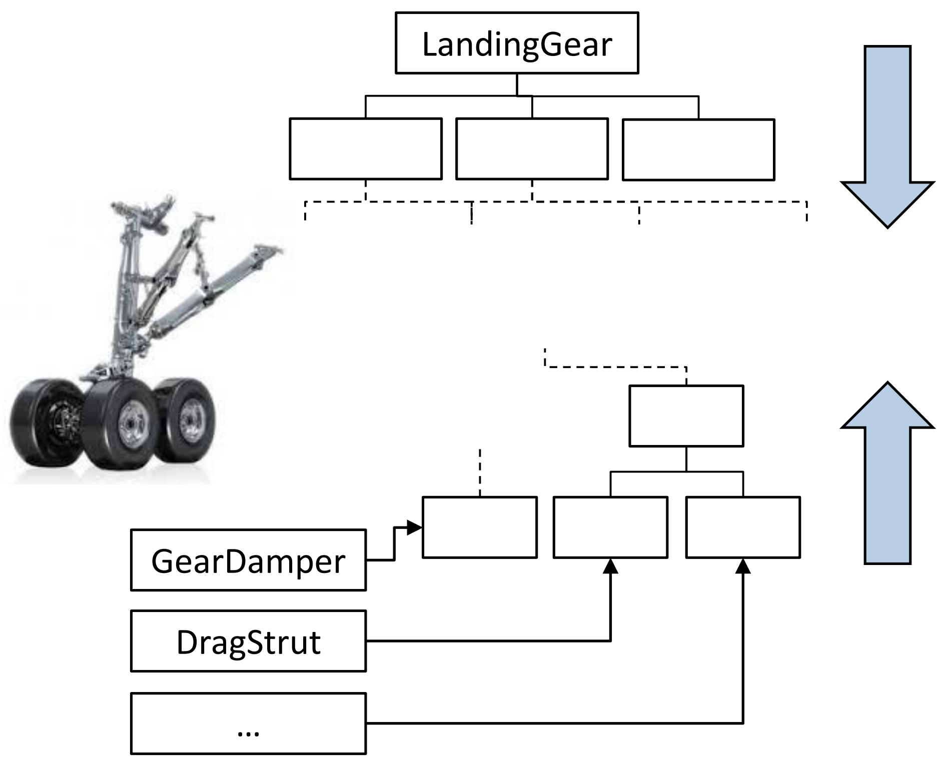
S2ML+X paradigm

Behaviors	+	Structures	=	Models
Guarded Transitions Systems	+	S2ML	=	AltaRica 3.0
Boolean equations	+	S2ML	=	Fault Trees (++)
Markov chains	+	S2ML	=	…
Petri nets	+	S2ML	=	GRIF (++)
Ordinary Differential Equations	+	S2ML	=	Simulink (++) Modelica (++)
Mealy machines	+	S2ML	=	Lustre (++)
Process algebras	+	S2ML	=	Scola
Bayesian networks	+	S2ML	=	…
Requirements	+	S2ML	=	…
…	+	…	=	…

Thesis 4 (Pattern-Based Systems Engineering)

Reuse is the key issue for the efficiency of the modeling process

Meaning and practical consequences:



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

system architecture

safety

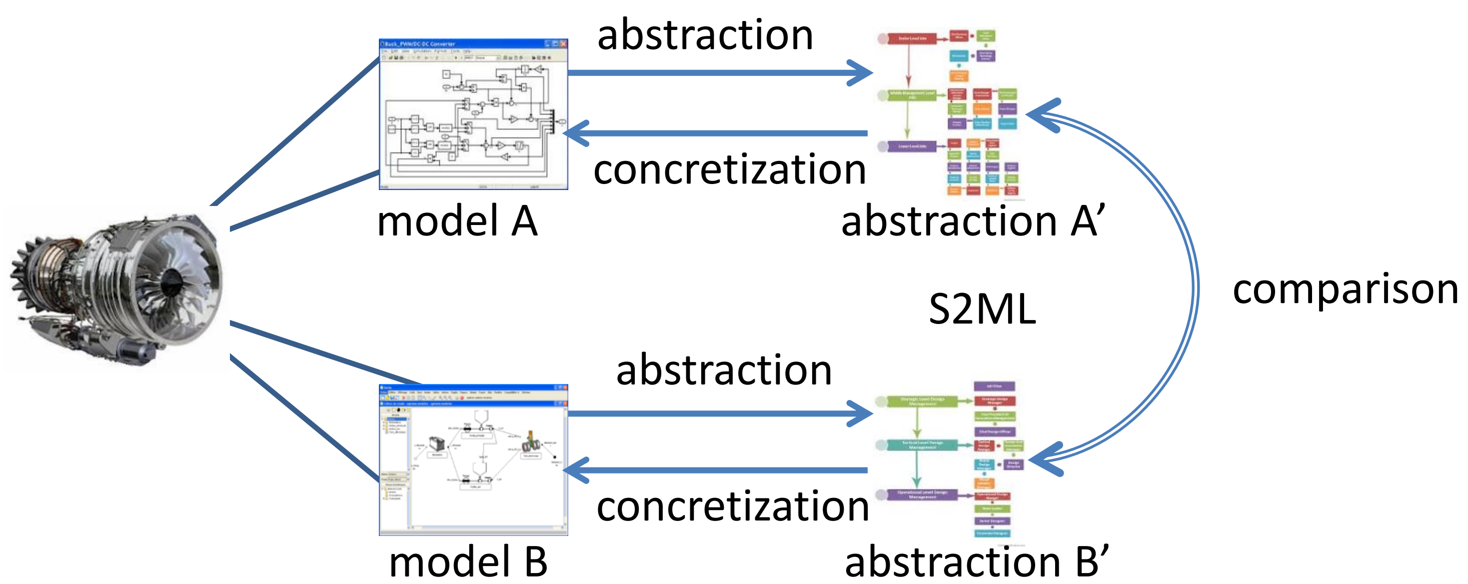
- Bottom-up model design
- Component level
- Reuse of modeling components
- Object-Orientation

Multiphysics simulation

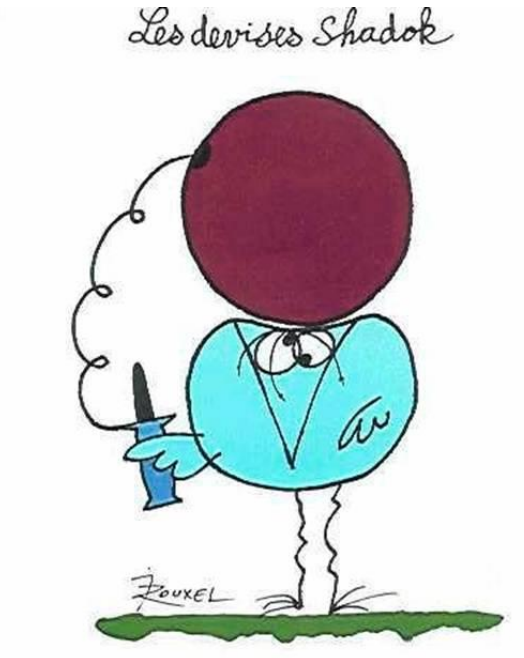
Thesis 5 (Model Synchronization)

Abstraction + Comparison = Synchronization*

Meaning and practical consequences:



(*) Cousot's abstract interpretation is thus the conceptual framework of model synchronization.



By trying and trying again, you always end up in succeeding. Consequently, the more you fail, the better your chances of success