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
domain State \{WORKING, HIDDEN\_FAILURE, DETECTED\_FAILURE\}
domain Mode \{OPERATION, INSPECTION\}

block PeriodicallyInspectedComponent

State state(\texttt{init}=WORKING);
Mode mode(\texttt{init}=OPERATION);
\textbf{event} failure(\texttt{delay}=exponential(lambda));
\textbf{event} repair(\texttt{delay}=exponential(mu));
\textbf{event} startInspection(\texttt{delay}=Dirac(tau));
\textbf{event} completeInspection(\texttt{delay}=Dirac(pi));
\textbf{parameter} Real lambda = 1.0e-3;
\textbf{parameter} Real mu = 0.1;
\textbf{parameter} Real tau = 720;
\textbf{parameter} Real pi = 12;
\textbf{transition}

failure: state==WORKING -> state:=HIDDEN\_FAILURE;
repair: state==DETECTED\_FAILED -> state:=WORKING;
startInspection: mode==OPERATION -> mode:=INSPECTION;
completeInspection: model==INSPECTION -> {
    mode:=OPERATION;
    state := \textbf{if} state==WORKING \textbf{then} WORKING \textbf{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

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

Fault Trees, Event Trees, Markov Chains, Stochastic Petri Nets...
The Model-Based Safety Assessment promise

Reducing the gap between systems specifications and probabilistic safety assessments
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
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
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.
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 \{\text{WORKING, HIDDEN\_FAILURE, DETECTED\_FAILURE}\}

domain Mode \{\text{OPERATION, INSPECTION}\}

block PeriodicallyInspectedComponent
  State state(init=\text{WORKING});
  Mode mode(init=\text{OPERATION});
  event failure(delay=exponential(\lambda));
  event repair(delay=exponential(\mu));
  event startInspection(delay=\text{Dirac}(\tau));
  event completeInspection(delay=\text{Dirac}(\pi));
  parameter Real \lambda = 1.0e-3;
  parameter Real \mu = 0.1;
  parameter Real \tau = 720;
  parameter Real \pi = 12;
  transition
    failure: state==\text{WORKING} \Rightarrow state:=\text{HIDDEN\_FAILURE};
    repair: state==\text{DETECTED\_FAILED} \Rightarrow state:=\text{WORKING};
    startInspection: mode==\text{OPERATION} \Rightarrow mode:=\text{INSPECTION};
    completeInspection: model==\text{INSPECTION} \Rightarrow {
      mode:=\text{OPERATION};
      state := if state==\text{WORKING} then \text{WORKING} else \text{DETECTED\_FAILED};
    }
end
Flow propagation

System under study

Controller

Sensors

Actuators

Equipment Under Control
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
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
end
```

**Model with a common cause failure**

```
Valve1.working
Valve2.working
A.failure
not Valve1.working
Valve2.working
B.failure
CCF
not Valve1..working
not Valve2.working
A.failure
CCF
not Valve1..working
not Valve2.working
B.failure
CCF
not Valve1..working
not Valve2.working
```

**synchronization**
Formal Definition

A **Guarded Transition Systems** is a quintuple \( \langle V, E, T, A, i \rangle \), 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**: \( V = S \cup F \).
- **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 \( V \).
- **i** 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: \( \langle \text{skip}, \sigma, \tau \rangle \rightarrow \tau \)

S1: \( \tau(v) = ?, \sigma(E) \in \text{dom}(v) \)
\( \langle v := E, \sigma, \tau \rangle \rightarrow \tau[\sigma(E)/v] \)

S2: \( \tau(v) = \sigma(E), \sigma(E) \in \text{dom}(v) \)
\( \langle v := E, \sigma, \tau \rangle \rightarrow \tau \)

S3: \( \sigma(E) = \text{ERROR} \) or \( \sigma(E) \notin \text{dom}(v) \) or \( \tau(v) \neq ?, \sigma(E) \neq \tau(v) \)
\( \langle v := E, \sigma, \tau \rangle \rightarrow \text{ERROR} \)

S4: \( \sigma(C) = \text{TRUE} \)
\( \langle \text{if } C \text{ then } I, \sigma, \tau \rangle \rightarrow \langle I, \sigma, \tau \rangle \)

S5: \( \sigma(C) = \text{FALSE} \)
\( \langle \text{if } C \text{ then } I, \sigma, \tau \rangle \rightarrow \tau \)

S6: \( \sigma(C) = \text{ERROR} \)
\( \langle \text{if } C \text{ then } I, \sigma, \tau \rangle \rightarrow \text{ERROR} \)

S7: \( \langle I_1, \sigma, \tau \rangle \rightarrow \tau' \)
\( \langle I_1; I_2, \sigma, \tau \rangle \rightarrow \langle I_2, \sigma, \tau' \rangle \)

S8: \( \langle I_2, \sigma, \tau \rangle \rightarrow \tau' \)
\( \langle I_1; I_2, \sigma \rangle \rightarrow \langle I_1, \sigma, \tau' \rangle \)

S9: \( \langle I_1, \sigma, \tau \rangle \rightarrow \langle I_1', \sigma, \tau' \rangle \)
\( \langle I_1; I_2, \sigma, \tau \rangle \rightarrow \langle I_1'; I_2, \sigma, \tau' \rangle \)

S10: \( \langle I_2, \sigma, \tau \rangle \rightarrow \langle I_2', \sigma, \tau' \rangle \)
\( \langle I_1; I_2, \sigma \rangle \rightarrow \langle I_1; I_2', \sigma, \tau' \rangle \)

S11: \( \langle I_1, \sigma, \tau \rangle \rightarrow \text{ERROR} \)
\( \langle I_1; I_2, \sigma, \tau \rangle \rightarrow \text{ERROR} \)

S12: \( \langle I_2, \sigma, \tau \rangle \rightarrow \text{ERROR} \)
\( \langle I_1; I_2, \sigma, \tau \rangle \rightarrow \text{ERROR} \)
Comparison with Existing Modeling Formalisms

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

```plaintext
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 endRepair (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
```

[Diagram of a guarded transitions system with states: Engine (working, failed, in repair) and RepairMan (free, busy). Events include failure, startRepair, endRepair, and an exponential parameter for the failure rate (lambda) and repair rate (mu).]

---

Norwegian University of Science and Technology
Agenda

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

```plaintext
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

<table>
<thead>
<tr>
<th>Ports</th>
<th>variables: state, demand, events: failure...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Containers</td>
<td>block SDV1, class Pump...</td>
</tr>
</tbody>
</table>

### Operational relations

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

### Hierarchical relations

<table>
<thead>
<tr>
<th>Composition</th>
<th>pump SDV1 <strong>is-part-of</strong> of system SIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregation</td>
<td>system SIS <strong>uses</strong> power-supply PW</td>
</tr>
</tbody>
</table>

### Reuse relations

<table>
<thead>
<tr>
<th>Instantiation</th>
<th>SDV1 <strong>is-a-copy-of</strong> on-the-shelf component Pump</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inheritance</td>
<td>Pump <strong>is-a</strong> PeriodicallyInspectedComponent</td>
</tr>
<tr>
<td>Cloning</td>
<td>train2 <strong>is-a-copy-of</strong> train1</td>
</tr>
</tbody>
</table>
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
Fault Trees, Event Trees, Markov Chains, Stochastic Petri Nets...

Calculations
• Failure Scenarii
• Reliability Indicators

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

<table>
<thead>
<tr>
<th>Behaviors</th>
<th>+</th>
<th>Structures</th>
<th>=</th>
<th>Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guarded Transitions Systems</td>
<td>+</td>
<td>S2ML</td>
<td>=</td>
<td>AltaRica 3.0</td>
</tr>
<tr>
<td>Boolean equations</td>
<td>+</td>
<td>S2ML</td>
<td>=</td>
<td>Fault Trees (++)</td>
</tr>
<tr>
<td>Markov chains</td>
<td>+</td>
<td>S2ML</td>
<td>=</td>
<td>...</td>
</tr>
<tr>
<td>Petri nets</td>
<td>+</td>
<td>S2ML</td>
<td>=</td>
<td>GRIF (++)</td>
</tr>
<tr>
<td>Ordinary Differential Equations</td>
<td>+</td>
<td>S2ML</td>
<td>=</td>
<td>Simulink (++) Modelica (++)</td>
</tr>
<tr>
<td>Mealy machines</td>
<td>+</td>
<td>S2ML</td>
<td>=</td>
<td>Lustre (++)</td>
</tr>
<tr>
<td>Process algebras</td>
<td>+</td>
<td>S2ML</td>
<td>=</td>
<td>Scola</td>
</tr>
<tr>
<td>Bayesian networks</td>
<td>+</td>
<td>S2ML</td>
<td>=</td>
<td>...</td>
</tr>
<tr>
<td>Requirements</td>
<td>+</td>
<td>S2ML</td>
<td>=</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>+</td>
<td>...</td>
<td>=</td>
<td>...</td>
</tr>
</tbody>
</table>
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

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