Model-Based Reliability Engineering: The AltaRica Language

Pr. Antoine Rauzy
LIX – École Polytechnique
FRANCE
Contents

• Motivations & Project Overview
• Formal Model
• Tools & Industrial Applications
• Conclusions & Perspectives
• Motivations & Project Overview
• Formal Model
• Tools & Industrial Applications
• Conclusions & Perspectives
What is modeled?

Engineered Systems: Aircrafts, Nuclear Power Plants, Cars....
Information systems
Social, biological systems

Why is it modeled?

To study whether the system presents acceptable risks for itself, its users and its environment.
Safety and Dependability Studies

Bidimensional Nature of Risk

Safety/Dependability Studies

Inputs:

• System specifications, operational/functional/physical architectures,
• Safety standards, rules, regulations
• Operation feedback

Outputs:

• (Severe) Failure Scenarios together with their probabilities
Models of Systems

- There are exponentially many failure scenarios
- They are implicitly described by models
- Their probability is calculated

System

- Internal State: q(t)
- Inputs: i(t)
- Events (aleas)
- Outputs: o(t)
- Fields: f(t)
Limitations of Classical Approaches

Fault Trees, Markov Processes, Petri Nets... have good properties:

- Well defined semantics
- Easy to handle
- Textual and graphical
- Good tradeoffs expressivity/efficiency

... But are too distant from the systems under study

=> Models are hard to design and to maintain
AltaRica Workflow

Systems

Guarded Transition Systems

GUI ~ Process & Instrumentation Diagrams

AltaRica

node repairableSystem

event startRepair, endRepair;

sub C:component, R:repairer;

sync

startRepair = C.startRepair and
R.startJob,
endRepair = C.endRepair and R.endJob;

edon

(Low Level) Models

Qualitative and Probabilistic Results
Contents

• Motivations & Project Overview
• Formal Model
• Tools & Industrial Applications
• Conclusions & Perspectives
A Valve

node valve
  state closed: bool;
  flow input: float; output: float;
  event open, close;
trans
  closed |- open -> closed:=false;
  not closed |- close -> closed:=true;
init
  closed := true;
assert
  output = if closed then 0 else input;
edon
Two Valves in Series

node twoValves
flow input: float; output: float;
sub A: valve, B: valve;
assert
A.input = input,
B.input = A.output,
output = B.output;
edon
Guarded Transition Systems [Rauzy 08]

A = <S,I,O,E,T,δ,α>
- S: state variables
- I: input variables
- O: output variables
- E: events
- T: transitions
- δ: transfer function
- α: initial state

assertion: O = δ(S,I)

transition: g(S,I)       S:=f(S,I)

No change in state variables
Between two events

instantaneous update:
action of the transition; recalculation of O = δ(S,I)
Types of Events

- Timed events: non null (stochastic) delay
  Probability distributions with parameters (exponential, Weibull, ...)
- Immediate events
- Instantaneous events: take no time and may have a probability to be fired
Composition of Guarded Transition Systems

Operations

Product Connection Synchronization of events

Synchronization

\[ \begin{align*}
G1 \mid e1 & \rightarrow S1:=f1 \\
G2 \mid e2 & \rightarrow S2:=f2 \\
G3 \mid e3 & \rightarrow S2:=f3 \\
e & = e1 \text{ and } (e2 \text{ or } e3)
\end{align*} \]

\[ \begin{align*}
G1 \text{ and } (G2 \text{ or } G3) \mid e & \rightarrow \\
S1 & := \text{if } G1 \text{ then } f1 \text{ else } S1 \\
S2 & := \text{if } G2 \text{ then } f2 \text{ else } S2 \\
S3 & := \text{if } G3 \text{ then } f3 \text{ else } S3
\end{align*} \]

fire the fireable local transitions
Guarded Transition Systems

... generalize fault trees, Markov graphs, Petri nets

\[(P1>0) \text{ and } (P2=0) \quad | \quad T \rightarrow P1:=P1-1, \ P3:=P3+2;\]

... generalize block-diagrams

... make it possible to define hierarchies, packages, ...

remote interactions
Contents

• Motivations & Project Overview
• Formal Model
• Tools & Industrial Applications
• Conclusions & Perspectives
Assessment Tools

- Failure Mode and Effect Analysis
- Model Validation (type checking...)
- Stepwise simulation
- Compilation to Fault Tree
- Model-Checking
- Sequence Generation
- Stochastic Simulation

Node repair system

Event startRepair, endRepair;
Sub C:component, R:repairer;
Sync
  startRepair = C.startRepair and R.startJob,
  endRepair = C.endRepair and R.endJob;
Edon

P&ID like GUI

AltaRica
AltaRica Workbenches

Cecilia OCAS

Safety Designer

Some Users

Aerospace & Defense

Transportation

DASSAULT

Simula

CASSIDIAN

DASSAULT SYSTEMES

SNCF

AIRBUS

THALES

SAGEM

cnes

RENAULT

ALSTOM
Models Designed So Far

3 categories of models

• Functional models
  – Mainly academic (Bordeaux) -> model checking

• Simple and big safety models (~ bloc diagrams), e.g.
  – Dassault F7X, ...
  – Compilation into fault trees
  – Treatment chain validated by certification authorities (FAA, EASA)

• Complex but (relatively) small models, e.g.
  – Total,
  – Production availability, High Integrity Protection Systems (IEC 61508)
  – Markov analyses, Monte-Carlo simulation
System Safety

Objective: Assess Safety of Ariane 5 Launcher Main Loop
Production Availability

Objective: estimate the average lost of production due to (possibly spurious) failures over a period of time (typically 1 to 5 years)
Contents

• Motivations & Project Overview
• Formal Model
• Tools & Industrial Applications
• Conclusions & Perspectives
Lessons Learned (10 years)

• High level modeling provides:
  – A better understanding of systems under study,
  – A better communication amongst stakeholders,
  – Models that are much easier to maintain and to reuse.

• Modeling to describe is not the same as modeling to calculate:
  – SysML vs Matlab/Simulink vs Scade vs AltaRica...
  – A model always results from a tradeoff between accuracy and (computational) complexity.

• Model-Based Engineering requires engineering models
  – A model is always a family of models (versions, configurations,...)
  – Modeling is always a two directions activity: bottom-up (reuse) and top down (design)
Model-Based System Engineering & Simulation

Dynamic Systems/Simulation: Differential Equations
e.g. Matlab-Simulink, Modelica

Safe Code Generation: Data-Flow models
e.g. State Charts, Scade

Reliability Engineering: Event-based models, non-determinism
e.g. AltaRica

Adopting the same (object-oriented) structure would facilitate
- Learning, documentation...
- Development, maintenance, versioning, storage of models...
- Transition from architecture formalisms (e.g. SysML)
- Introduction of higher order concepts (e.g. reactive ML [Pouzet])

First attempt S2ML [Perrot, Rauzy 2009]