Early Integration of Dependability Studies in the Design of Cyber-Physical Systems

Thuy NGUYEN - EDF R&D
Marc BOUISSOU - EDF R&D
19 juin 2019

Summary

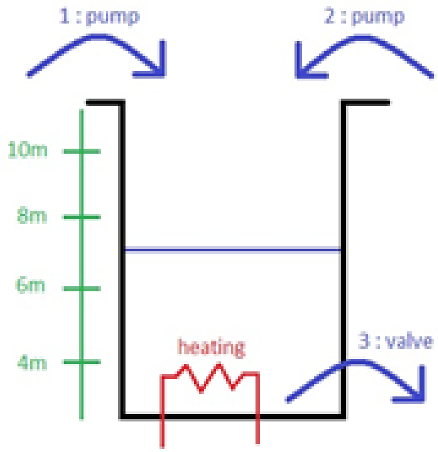
- Introduction
- **FORM-L and the MODRIO approach**
- **Figaro**
- **The Heated Tank Example**

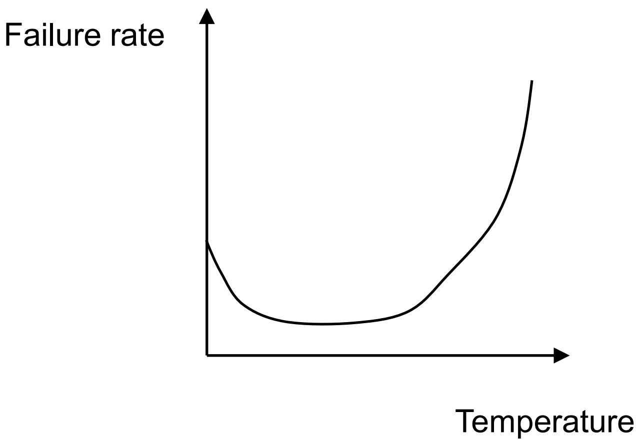
Motivation

- **Dependability studies are often performed long after design studies**
  - By a specialised team that did not participate in the design
- **Designers tend to focus on the normal functioning of the system**
- **They select components and add redundancies without being able to assess the dependability of the whole**
- **The dependability of cyber-physical systems (CPS) is often influenced by physical conditions**
  - In particular during exceptional situations which may be hard to fully understand without physical simulation ➔ Errors are often revealed only after a prototype or a detailed simulation model is available
- **If serious problems are discovered, redesign can result in very high costs and delays**

Motivation

- **Examples**
  - Extreme ambient conditions (temperature, humidity, pressure) may be determined by the operation of the system and at the same time influence the dependability of its components
  - Shocks, vibrations, clogging and wear may also influence the dependability of components



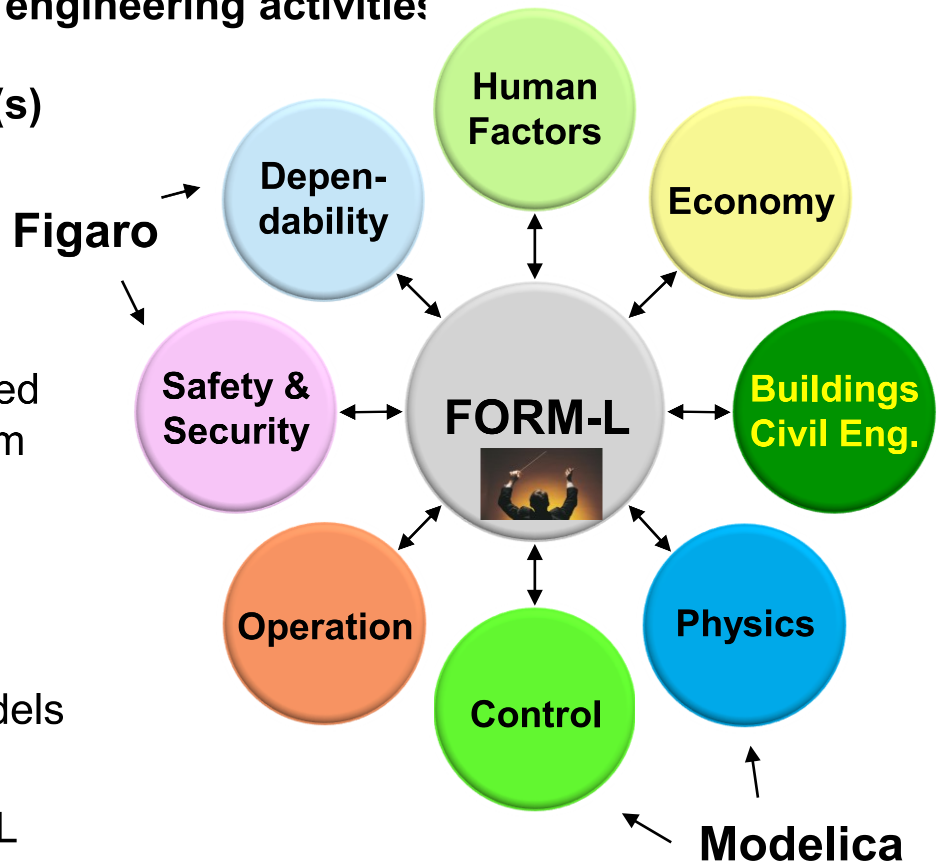


Approach & Principles

- **A systems engineering approach that closely integrates dependability with the other design studies**
  - At all stages of requirements elicitation and solutions design and V&V
- **Formal specification, using the Form-L language, of requirements and preliminary solutions for the system**
  - Including dependability and probabilistic aspects
- **Automatic derivation of code to be inserted in**
  - Physical simulation models, to check that deterministic requirements are not violated
  - Dependability models, to check compliance with probabilistic requirements
- **Use of simulation and analysis at all engineering stages to verify that solutions comply with requirements**
  - Dependability studies can guide the design of solutions even at early stages of engineering

Thrifty Modelling

- **The engineering of a complex CPS requires the collaboration and coordination of many disciplines and teams**
- **Each discipline performs many different engineering activities**
- **Each activity may require its own model(s)**
- **There is a risk of**
  - Inconsistency: models used for different activities could contradict one another
  - Waste of effort: different models could need the same information regarding the system (e.g., its architecture), but each in its own format
- **Thrifty modelling**
  - Allows the use of specific disciplinary models
  - But avoids unnecessary duplication with the use of coordination models in FORM-L



Summary

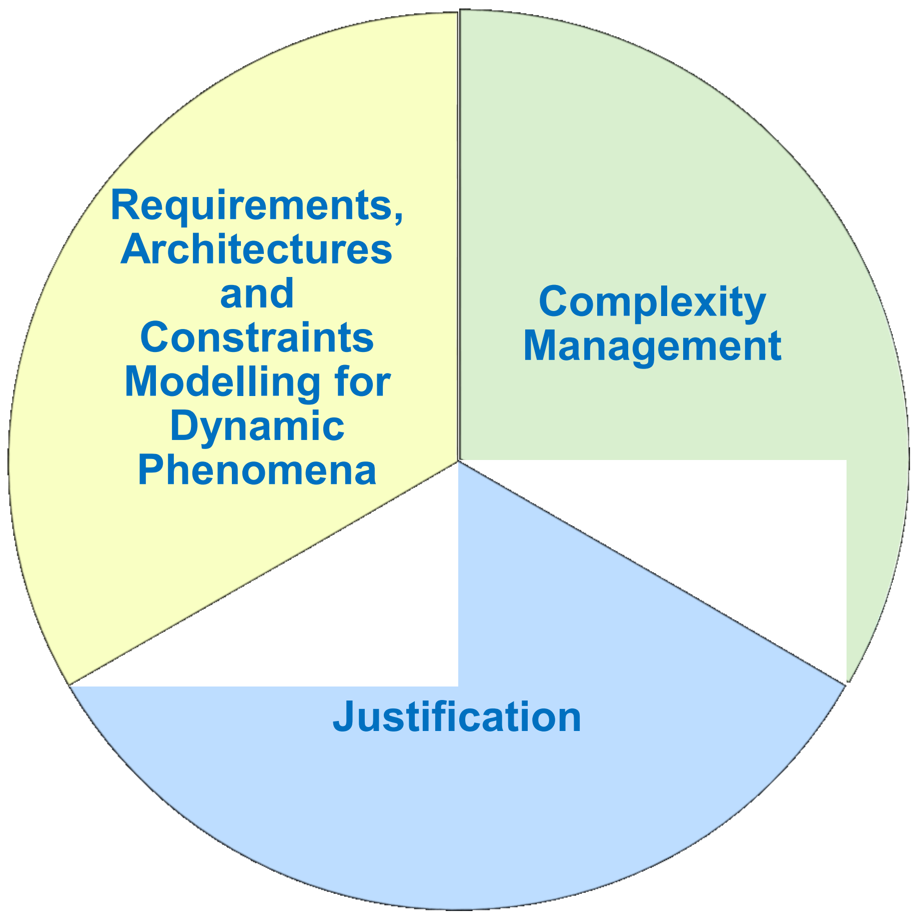
- **Introduction**
- **FORM-L and the MODRIO approach**
- **FIGARO**
- **The Heated Tank Example**

FORM-L (FOrmal Requirements Modelling Language)

- **Developed in the framework of project**

INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

MOdel DRIven physical systems Operation

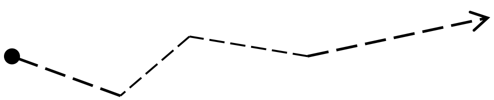


Various Forms of Modelling for Dynamic Phenomena

- **Non-formal** **or** **semi-formal** **modelling**
  - Natural language or drawings ➔ often ambiguous
    - Examples: SADT or SysML
  - Limited analysis and simulation capabilities ➔ a problem for complex systems
  - Individual models tend to address and be useful for only for a limited part of the lifecycle

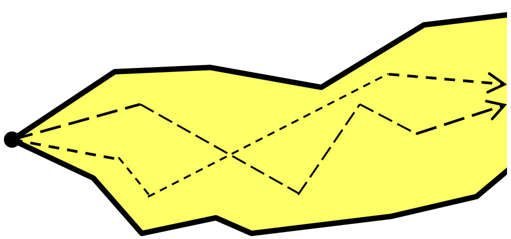
well defined syntax & semantics

- **Deterministic formal** **modelling**
  - Given initial and boundary conditions, only one possible behaviour
    - Examples: Modelica or finite elements models for physics, functional block diagrams for I&C, ...
  - Detailed and accurate ➔ only for downstream engineering activities
  - In general, specific to a discipline



Deterministic Model

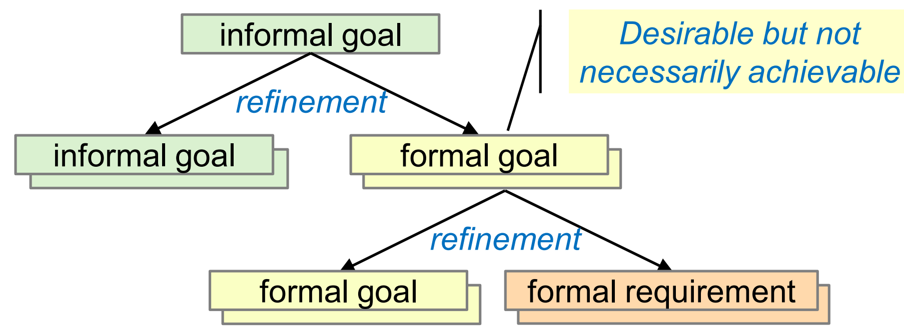
- **Constraints-based formal** **modelling (for CPS: FORM-L)**
  - Envelopes of expected behaviours: avoid over-specification, enables simplification and abstraction
  - To model requirements, assumptions and preliminary solutions ➔ for engineering activities along the complete lifecycle
  - Can also represent uncertainties and human variability



Constraints Model

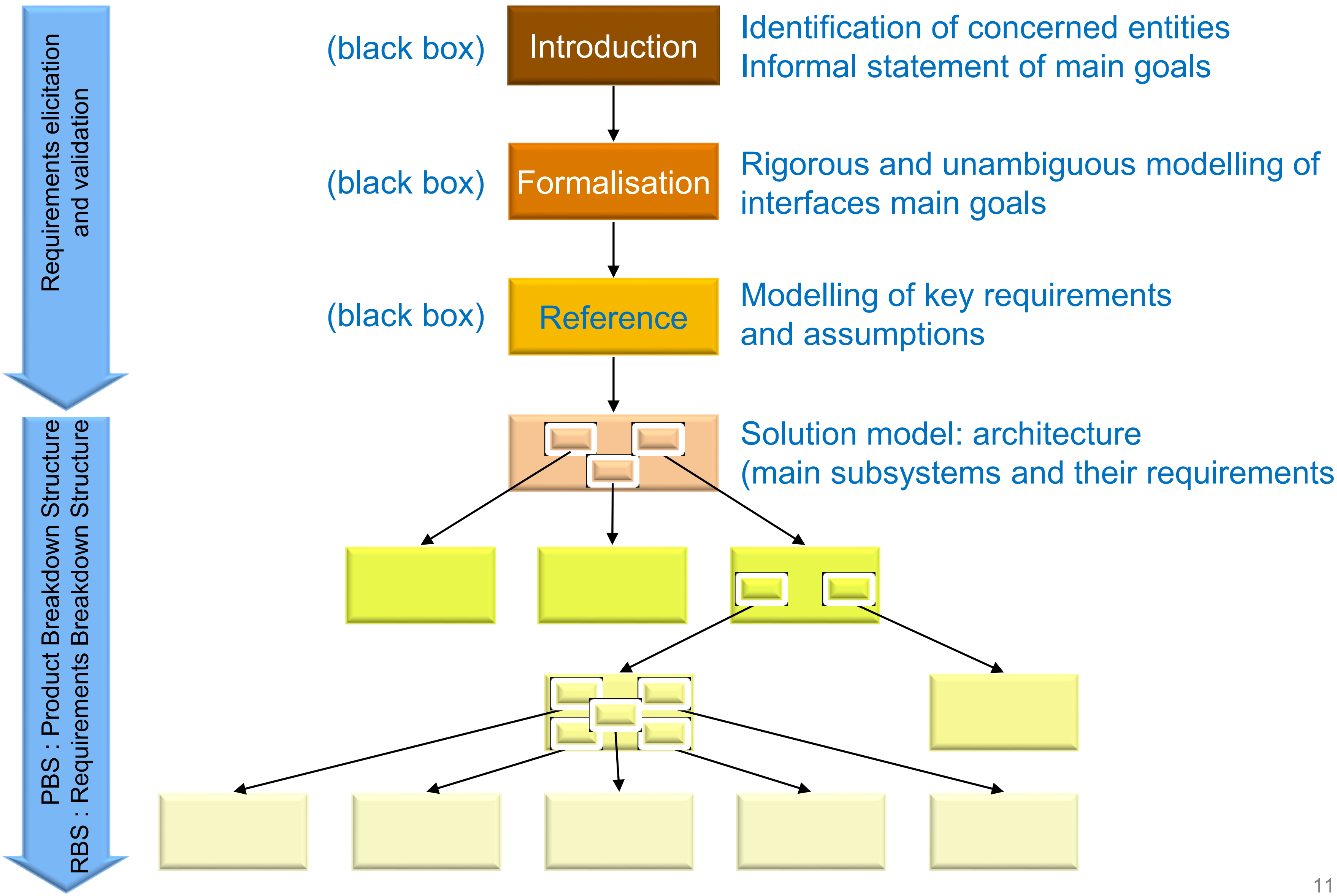
Key Concepts

- **Variables**
  - Functions of time
  - Booleans, integers, reals, quantities, finite state automata and statecharts, strings
- **Events**
- **Goals**
  - Informal statement of what one aims at achieving
- **Properties**
  - Formal statement of a constraint (WHAT), a time locator (WHEN), and a spatial locator (WHERE)
  - Deterministic vs probabilistic constraints
  - Simple properties, objectives (formal statement of what one aims at achieving), requirements, assumptions
- **Refinement**
  - Step-by-step transformation of an informal goal into one or more informal or formal subgoals, and formal requirements



- **Justification**
  - One or more goals, objectives, requirements and assumptions that justify a refinement

Step-by-Step Refinement & Verification of Solutions



Summary

- **Introduction**
- **FORM-L and the MODRIO approach**
- **FIGARO**
- **The Heated Tank Example**

Figaro modeling language objectives

- **Provide an appropriate formalism for generic descriptions of components**
- **Be more general than all usual reliability models**
- **Find the best trade-off between modelling power (or generality) and possibilities for the processing of models**
- **Be as legible as possible**
- **Be easily associated with graphic representations**
- **Have a formally defined semantics – support consistency proofs**

The Figaro language, developed in 1990, has been validated by hundreds of studies of complex systems

KB3 workbench principles

Knowledge Base (in Figaro language)

Generic description of components

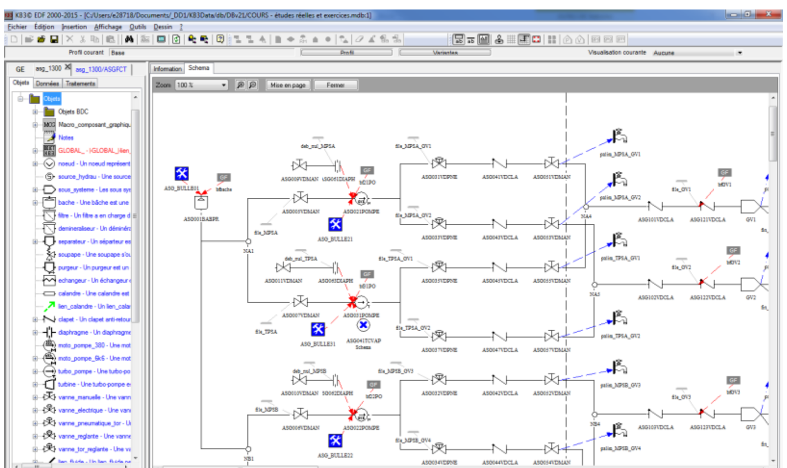
Figaro 0
(textual model)

If static model

KB3 fault tree generator

Standard fault-tree processors: GRIF/Tree, Risk-Spectrum…
- ✓Minimal cutsets
- ✓Reliability,
- ✓Availability

Graphic input of system models in KB3



If dynamic model

Sequences Generator: FIGSEQ
- ✓Most probable sequences
- ✓Reliability, MTTR
- ✓Asymptotic availability

Monte-Carlo simulator: YAMS
- ✓Most probable sequences
- ✓Reliability, availability
- ✓Mean values of numeric variables…

Benefits of MBSA with KB3

- **Studies consistency**
- **Assumptions traceability and legibility**
- **Studies quality**
- **Productivity (40% to 80% time saving)**
- **Accessibility to non-specialists**

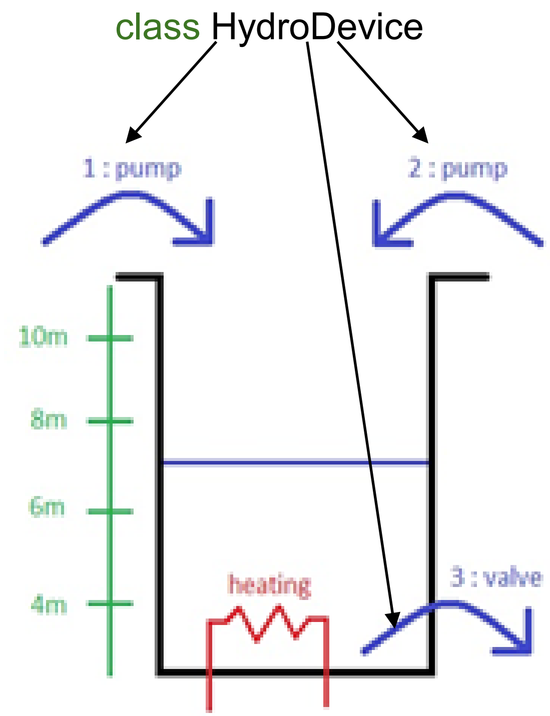
- **+**
- **In nuclear PSAs, the KB3 models ensure**
  - a consistent coding of events, intermediate gates and top gates
  - an easier exploitation of the models
  - easy maintenance of models, over tens of years

Summary

- **Introduction**
- **FORM-L and the MODRIO approach**
- **FIGARO**
- **The Heated Tank Example**

Introduction to the Heated Tank

- **The system purpose is to hold a reserve of warm water, with a certain degree of fault tolerance**
- **The heating device delivers a constant power**
- **A control system monitors the fluid level in the tank and sends orders to the pumps and the valve to keep the level in the range [6m, 8m]**
  - One can consider it is represented by failure modes leading to pumps and valve stuck_on and stuck_off states
  - For the valve, "on" means open and "off" means closed
- **The system is not repairable**



- **Names in the models are those of the paper taken here as a reference:**

Zhang H., Dufour F., Dutuit Y. and Gonzalez, K. (2009). Piecewise deterministic Markov processes and dynamic reliability. Proceedings of the Institution of Mechanical Engineers, Part O: *Journal of Risk and Reliability*, volume 222(4), pages 545–551.

FORM-L Model - Class HydroDevice

```
class HydroDevice
  Duration^-1 lambda_hat is specific;
  Duration^-1 lambda is a(theta)*lambda_hat;
  private Real a(Temperature t)
    Temperature^-1 b1 is 3.0295/_K;
    Temperature^-1 b2 is 0.7578/_K;
    Temperature^-1 bc is 0.05756/_K;
    Temperature^-1 bd is 0.2301/_K;
    define value is (b1*exp(bc*(t-20*_K)) + b2*exp(-bd*(t-20*_K)))/(b1+b2);
  end a;

  automaton state (on, off, stuck_on, stuck_off)
    when t0 ensure value in {on, off};
    during value = on define (next becomes stuck_on).rate = lambda;
    during value = on define (next becomes stuck_off).rate = lambda;
    during value = off define (next becomes stuck_on).rate = lambda;
    during value = off define (next becomes stuck_off).rate = lambda;
    after value becomes stuck_on define next is value;
    after value becomes stuck_off define next is value;
  end state;

  Real v_c is if state in {on, stuck_on) then 1 else 0;
end HydroDevice;
```

specific to each instance

failure rate depends on temperature

failures

no repair

1 iff water is flowing through

FORM-L Model - Pumps & Valve

```
HydroDevice pump_1
  lambda_hat is 2.2831e-3/h;
  when t0 define state is on;    // Initial state
end pump_1;

HydroDevice pump_2
  lambda_hat is 2.8571e-3/h;
  when t0 define state is off;   // Initial state
end pump_2;

HydroDevice valve_1
  lambda_hat is 1.5625e-3/h;
  when t0 define state is on;    // Initial state
end valve_1;
```

```
HydroDevice valve_2
  lambda_hat is 1.5625e-3/h;
  when t0 define state is off;
end valve_2;
```

Could be added to the model to describe an architecture variant. The FORM-L model is designed to accept any number of pumps and valves (see next slides)

FORM-L Model - Tank 1/2

```
object tank
  HydroDevice {} input is {pump_1, pump_2};
  HydroDevice {} output is {valve_1};

  Length max_level is 8*m;
  Length min_level is 6*m;
  Length overFlowLevel is 10*m;
  Length dryOutLevel is 4*m;

  event eDryout   is (level <= dryOutLevel)    becomes true;
  event eOverflow is (level >= overFlowLevel) becomes true;
  event eBoiling  is (theta >= 100*_C)         becomes true;

  Real p1 is probability (eDryout);
  Real p2 is probability (eOverflow);
  Real p3 is probability (eBoiling);
```

components in interaction with the tank (these sets may be modified to represent different architectures)

set points for water level

undesirable events

their probabilities

FORM-L Model - Tank 2/2

water level, defined by initial value and derivative

```
Length level
  when t0 define value is 7*m;
  define derivative is
    gg*(sum(for all p in input: p.v_c)-sum(for all p in output: p.v_c));
end level;
```

water temperature, defined by initial value and derivative

```
Temperature theta
  when t0 define value is 30.9261*_C;
  define derivative is
    (gg*sum(for all p in input: p.v_c)*(theta_in-value) + 23.88915*_C*m/h)/level;
end theta;
```

pumps control

```
for all p in input begin
  during level <= min_level and p.state = off define p.state.next is on;
  otherwise during level >= max_level and p.state = on define p.state.next is off;
  otherwise ensure no (p.state.next leaves (on, off) towards (on, off));
end;
```

valves control

```
for all p in output begin
  during level <= min_level and p.state = on define p.state.next is off;
  otherwise during level >= max_level and p.state = off define p.state.next is on;
  otherwise ensure no (p.state.next leaves (on, off) towards (on, off));
end;
end tank;
```

Translation into Figaro

- **This relatively simple model can be automatically translated into Figaro**
  - Figaro is dedicated to discrete systems, so the Figaro model will not be as precise as a truly hybrid model
  - Work is ongoing to develop a translation tool (Form-L → Figaro)

Excerpts of the Figaro model (1/2)

Definition of a clock to
ensure a *maximum*
time difference
between two events

```
STEPS_ORDER
 default_step ;   (* includes updates of continuous variables *)
 control ;        (* actions depending on continuous variables *)
 memorisation ;  (* derivatives at this instant are memorised *)

CLASS Clock ;
  CONSTANT time_step DOMAIN REAL DEFAULT 0.5 ;
  ATTRIBUTE tick DOMAIN BOOLEAN DEFAULT FALSE ;
                last_event_date DOMAIN REAL DEFAULT 0;

  OCCURRENCE
    IF NOT tick
    MAY_OCCUR TRANSITION t
    DIST T_C(time_step)
    INDUCING tick ;

  INTERACTION
  (* The tick duration is zero *)
  STEP memorisation
  IF tick THEN tick <-- FALSE;
  STEP memorisation
  THEN  last_event_date <-- CURRENT_DATE ;

SYSTEM_OBJECT C IS_A Clock;
```

In the class Tank

sets of components in interaction with the tank

```
CLASS Tank;

 INTERFACE
      input KIND HydroDevice ;
      output KIND HydroDevice ;

 ATTRIBUTE
    level              (*  Height of the water in the tank *)
          DOMAIN REAL DEFAULT 7;
   der_level DOMAIN REAL DEFAULT 0;
 EFFECT level_updated ;

 INTERACTION
   STEP default_step  (* update of level - Euler method *)
   IF NOT level_updated
   THEN level <-- level + der_level *
        (CURRENT_DATE - last_event_date(C)) ,
        level_updated ;

   STEP memorisation
   THEN der_level <-- gg*
    (SUM FOR_ALL x AN input OF_TERMS v_c(x) -
     SUM FOR_ALL y AN output OF_TERMS v_c(y));
```

update of the derivative of the level

Excerpts of the Figaro model (2/2)

Definition of the control
by hysteresis
(in the Tank)

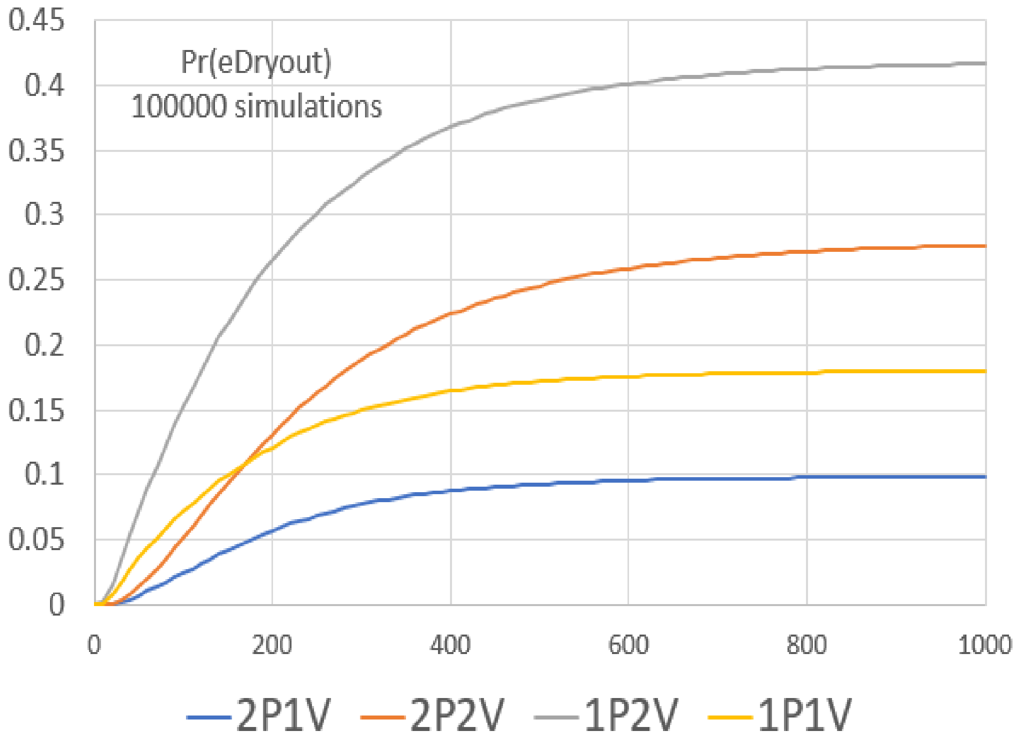
```
INTERACTION
    GIVEN p AN input  STEP control
    IF  level <= min_level AND  state(p) = 'off'
    THEN state(p) <-- 'on';

    GIVEN p AN input STEP control
    IF  level >= max_level AND  state(p) = 'on'
    THEN state(p) <-- 'off';

    GIVEN p AN output STEP control
    IF  level <= min_level AND  state(p) = 'on'
    THEN state(p) <-- 'off';

    GIVEN p AN output STEP control
    IF  level >= max_level AND  state(p) = 'off'
    THEN state(p) <-- 'on';
```

Definition of random failures in the class
Hydrodevice

```
ATTRIBUTE
  state DOMAIN 'on' 'off' 'stuck_on' 'stuck_off'
        DEFAULT 'on';

OCCURRENCE
    IF state='on' MAY_OCCUR FAULT fail_stuck_on
          DIST EXP (lambda)
          INDUCING state <-- 'stuck_on';

    IF state='on' MAY_OCCUR FAULT fail_stuck_off
          DIST EXP (lambda)
          INDUCING state <-- 'stuck_off';

    IF state='off' MAY_OCCUR FAULT fail_stuck_on
          DIST EXP (lambda)
          INDUCING state <-- 'stuck_on';

    IF state='off' MAY_OCCUR FAULT fail_stuck_off
          DIST EXP (lambda)
          INDUCING state <-- 'stuck_off';
```

The failure rate lambda is updated at each event (clock or other)

Use of the Figaro model to compare 4 architectures



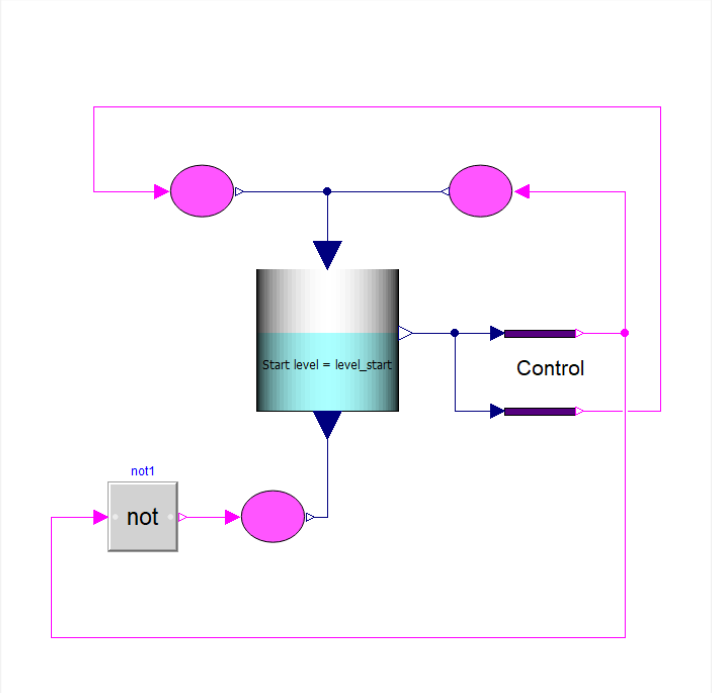
Simulation time : 6mn on a standard laptop
for 10^5 simulations

Probability of the 3 undesirable events at 500 hours

Variant	2P1V	2P2V	1P2V	1P1V
eDryout	**0.093**	0.246	0.389	0.172
eBoiling	0.164	**0.103**	0.219	0.340
eOverflow	0.462	0.260	**0.100**	0.212
Total	0.719	0.609	0.708	0.725

How about deterministic requirements?

- **A similar approach can be applied with Modelica**
  - Deterministic requirements can be automatically translated into Modelica
    - Using the ReqSysPro Modelica library
  - These requirements can then be checked automatically during Modelica simulations



Modelica model of the Heated tank

Example of deterministic requirement: starting from the nominal state, whatever the failures, there can not be a shortage of water in less than x hours

ce serait bien de mettre l'exigence en Form-L et en Reqsyspro

Conclusion

- **On the basis of a simple example, we have illustrated the principles of the approach we propose for performing dependability studies very early in the engineering of cyber-physical systems, in close interaction with other engineering disciplines such as architecture design and physical processes studies**

- **We have also shown how a reference model in FORM-L can be automatically translated into FIGARO for dependability studies**

- **Translation from Form-L to Modelica has not been addressed in this paper, but other work is ongoing to integrate the requirements expressed in a FORM-L model into a Modelica solution model so that the satisfaction or violation of requirements can be automatically checked during simulation**

- **This approach saves time in models development and ensures consistency between models used by different disciplines**

Thank you for your attention

Any questions?