Models-Informed Engineering of Complex Socio-Cyber-Physical Systems and Large Scale Systems of Systems

A System Owner (Maître d'Ouvrage) Standpoint

ICSSEA 2016
Journée NEPTUNE

N. Thuy - EDF R&D

Overview

- **Motivation**
- **Quick introduction to FORM-L (FOrmal Requirements Modelling Language)**
- **Quick introduction to the associated method**

Do We Have the Right Behavioural Requirements?

- **"Weaknesses in requirements are one of the most significant contributors to systems and software failing to meet the intended goals."**
  OECD COMPSIS Project Report – November 2011

- **In the US, in 1991: a required nuclear reactor trip signal was delayed by 16 s**
  - Two simultaneous events: situation not addressed by the requirements specification

- **In France: multiple versions of safety-critical software in the first year of operation**
  - No serious issues were found, but initial requirements were not fully satisfactory

- **Also true in other safety-concerned industrial sectors, in some cases with catastrophic consequences**

To prevent spurious actuation while airborne, thrust reversers are required to be de-energised when an aircraft is not on the ground

Small airport, No local control tower, Snow, Poor visibility (Real accident)

Do We Have the Right Behavioural Requirements ?

- **“Weaknesses in requirements are one of the most significant contributors to systems and software failing to meet the intended goals.”**
  OECD COMPSIS Project Report – November 2011

- **In the US, in 1991: a required nuclear reactor trip signal was delayed by 16 s**
  - Two simultaneous events: situation not addressed by the requirements specification

- **In France: multiple versions of safety-critical software in the first year of operation**
  - No serious issues were found, but initial requirements were not fully satisfactory

- **Also true in other safety-concerned industrial sectors, in some cases with catastrophic consequences**

Wheels on the ground → Thrust reversers are energised and in operation

Small airport, No local control tower, Snow, Poor visibility (Real accident)

Do We Have the Right Behavioural Requirements ?

- **"Weaknesses in requirements are one of the most significant contributors to systems and software failing to meet the intended goals."**
  OECD COMPSIS Project Report – November 2011

- **In the US, in 1991: a required nuclear reactor trip signal was delayed by 16 s**
  - Two simultaneous events: situation not addressed by the requirements specification

- **In France: multiple versions of safety-critical software in the first year of operation**
  - No serious issues were found, but initial requirements were not fully satisfactory

- **Also true in other safety-concerned industrial sectors, in some cases with catastrophic consequences**

The pilots see a snow plough on the runway →
They deactivate the thrust reversers and take-off

Small airport, No local control tower, Snow, Poor visibility (Real accident)

Do We Have the Right Behavioural Requirements ?

- "**Weaknesses in requirements are one of the most significant contributors to systems and software failing to meet the intended goals.**"
  OECD COMPSIS Project Report – November 2011

- **In the US, in 1991: a required nuclear reactor trip signal was delayed by 16 s**
  - Two simultaneous events: situation not addressed by the requirements specification

- **In France: multiple versions of safety-critical software in the first year of operation**
  - No serious issues were found, but initial requirements were not fully satisfactory

- **Also true in other safety-concerned industrial sectors, in some cases with catastrophic consequences**

Wheels off the ground → Thrust reversers are de-energised: one is fully stowed, not the other

Small airport, No local control tower, Snow, Poor visibility (Real accident)

Do We Have the Right Behavioural Requirements ?

- **"Weaknesses in requirements are one of the most significant contributors to systems and software failing to meet the intended goals."**
  OECD COMPSIS Project Report – November 2011
- **In the US, in 1991: a required nuclear reactor trip signal was delayed by 16 s**
  - Two simultaneous events: situation not addressed by the requirements specification
- **In France: multiple versions of safety-critical software in the first year of operation**
  - No serious issues were found, but initial requirements were not fully satisfactory
- **Also true in other safety-concerned industrial sectors, in some cases with catastrophic consequences**

Aerodynamic pressure reopens it
→ Aircraft off-balance, pilots don't have time to react

Small airport, No local control tower, Snow, Poor visibility (Real accident)

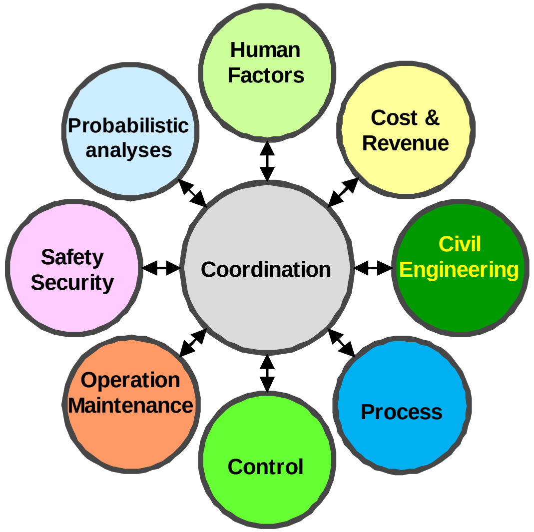
General Principles

- **Consider systems in their operational context**
- **Any system is a socio-cyber-physical system (SCPS): one ignores it at one's peril**
  - Human actions
  - Automated controls
  - Process, geographic proximity, connectivity, ...

- **Formal modelling of assumptions, requirements and design**
  - Focus on dynamic phenomena
  - Remove ambiguity of natural or semi-formal languages
  - Avoid overspecification
  - Address innovation, safety and dependability
  - Provide tool support
    - Simulation, static analysis, optimisation in design and operation, automated test case generation, automated results verification, failure analyses, training, operation support,...
  - Thrifty modelling: reuse models along system lifecycle whenever possible
  - Models and modelling patterns as repositories of design knowledge and lessons learned

Modelling of Large, Complex SCPSs

- **Master complexity**
  - Modular modelling
  - Models composition (top-down and bottom-up)
- **Coordinate numerous teams & disciplines**
  - Teams working on different parts of the system
  - Disciplines that often don't understand one another
  - Coordination "just as needed"
    - Neither too much (paralysis) nor too little (chaos)
- **Cover the complete system lifecycle**
  - Including retrofits and upgrades in 40 years from now



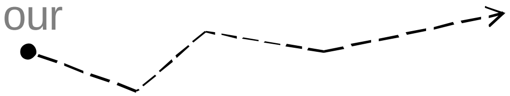
operation & maintenance — *projects (new builds, retrofits, upgrades)* — *prospective & scoping studies*

recent past: maintenance | *immediate future:* optimisation | *mid-term future:* validation & commissioning, detailed design, architectural design, system specification, system requirements | *far future:* what types of systems will be needed?

EDF

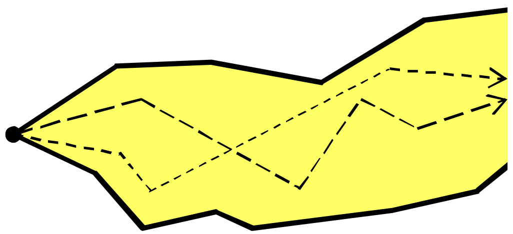
Formal Modelling of Dynamic Phenomena

- **Deterministic models**
  - Given initial and boundary conditions, only one possible behaviour
  - Examples: Modelica models, functional block diagrams , ...
  - Detailed and accurate ➔ for downstream engineering activities



Deterministic Model

- **Constraints-based models**
  - Envelopes of acceptable behaviours: avoid overspecification
  - Also envelopes of uncertainties
  - To specify requirements, assumptions and preliminary designs
    ➔ for engineering activities along the complete lifecycle



Constraints Model

- **Not only for physics and controls**
  - Human actions and procedures (operation & maintenance)
  - Events, such as components failures, malicious or natural agressions
  - Economic aspects
  - Tasks scheduling
  - ...

Which Constraints-Based Modelling Language?

- **Many languages for cyber systems ...**
  - OCL (Object Constraint Language) of the OMG, associated with SysML
  - MARTE (Modelling and Analysis of Real-Time and Embedded Systems)
  - AADL (Architecture and Analysis Design Language)
  - PSL (Property Specification Language)
  - ARTiMon (A Real-Time Monitor)
  - ... and many others
- **... but none found really addressing the needs of socio-cyber physical systems**
  - Continuous time, noise and uncertainties, variability of human behaviour
  - Random failures, fault-tolerance and probabilistic requirements
  - Functional propagations, but also failure propagation by agression and invasion
    - Example: crash of the Concorde
- **Development of FORM-L in the framework of project**
  - FOrmal Requirements Modelling Language
  - In association with the Modelica language and the Modelica Association

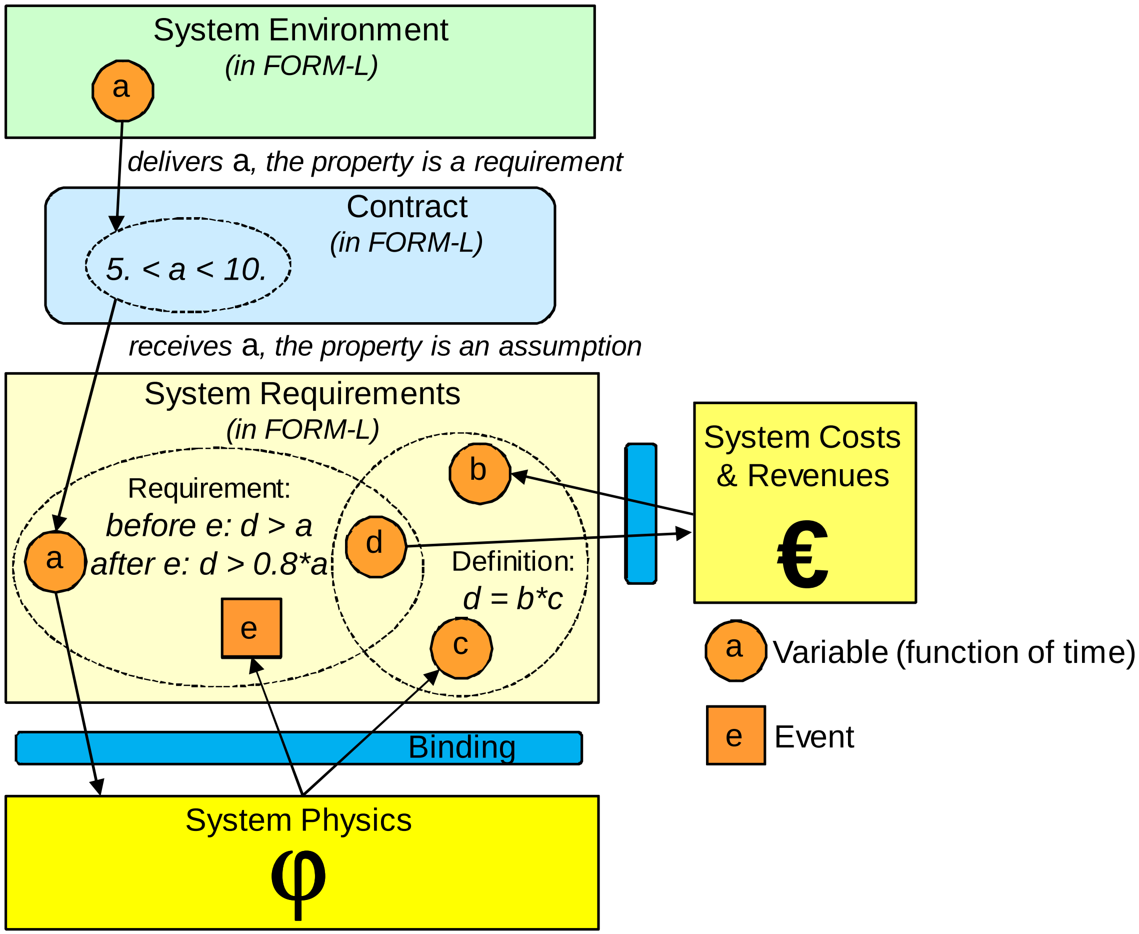
ITEA 2 INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

MODRIO *MOdel DRIven physical systems Operation*

Overview

- Motivation
- Quick introduction to FORM-L (FOrmal Requirements Modelling Language)
- Quick introduction to the associated method

Overview


System Environment
(in FORM-L)
a
delivers a, the property is a requirement
Contract
(in FORM-L)
5. < a < 10.
receives a, the property is an assumption
System Requirements
(in FORM-L)
Requirement:
before e: d > a
after e: d > 0.8*a
a
b
d
e
c
Definition:
d = b*c
System Costs
& Revenues
€
a
Variable (function of time)
e
Event
Binding
System Physics
φ

Example 1: Probabilistic Requirement

When the system is in operation, the probability that a pump in room A fails more than 2 times a year shall be less than 0.1 %

```
class Pump
   external Boolean failure;
   event eFails = when failure becomes true;
   external String room;
end Pump;

object coolingSystem
   external Boolean operation;
   external Pump { } pumps;


   property prop1 =
      forAll p in pumps suchThat p.room = "A"
      durringAny 1*year of operation
      check (count p.eFails) ≤ 2;

   requirement req1 =
      check probability prop1.violated < 0.001;

   end coolingSystem;
```

Pump failure state indicator → *external Boolean* failure;

Instants of failure occurrence → *event* eFails = *when* failure *becomes true*;

Room in which the pump is located → *external String* room;

external means the value is to be provided by a solution or scenario model

Indicates that the system is operating → *external Boolean* operation;

All the pumps of the system → *external* Pump { } pumps;

Desirable property → *property* prop1 =

WHERE → *forAll* p *in* pumps *suchThat* p.room = "A"

WHEN → *duringAny* 1*year *of* operation

WHAT → *check* (*count* p.eFails) ≤ 2;

The true requirement → *requirement* req1 =

HOW WELL → *check probability* prop1.*violated* < 0.001;

WHY and HOW are expressed by models organisation

Example 1: Probabilistic Requirement

When the system is in operation, the probability that a pump in room A fails more than 2 times a year shall be less than 0.1 %

	class Pump
Pump failure state indicator →	*external Boolean* failure;
Instants of failure occurrence →	*event* eFails = *when* failure *becomes true*;
Room in which the pump is located →	*external String* room;
	end Pump;
external means the value is to be provided by a solution or scenario model	
	object coolingSystem
Indicates that the system is operating →	*external Boolean* operation;
All the pumps of the system →	*external* Pump { } pumps;
Desirable property →	*property* prop1 =
WHERE →	*forAll* p *in* pumps *suchThat* p.room = "A"
WHEN →	*duringAny* 1*year *of* operation
WHAT →	*check* (*count* p.eFails) ≤ 2;
The true requirement →	*requirement* req1 =
HOW WELL →	*check probability* prop1.*violated* < 0.001;
WHY and HOW are expressed by models organisation	
	end coolingSystem;

Example 1: Probabilistic Requirement

When the system is in operation, the probability that a pump in room A fails more than 2 times a year shall be less than 0.1 %

```
class Pump
  external Boolean failure;
  event eFails = when failure becomes true;
  external String room;
end Pump;

object coolingSystem
  external Boolean operation;
  external Pump { } pumps;


  property prop1 =
    forAll p in pumps suchThat p.room = "A"
    duringAny 1*year of operation
    check (count p.eFails) ≤ 2;

  requirement req1 =
    check probability prop1.violated < 0.001;

  end coolingSystem;
```

Pump failure state indicator → external Boolean failure;

Instants of failure occurrence → event eFails = when failure becomes true;

Room in which the pump is located → external String room;

external means the value is to be provided by a solution or scenario model

Indicates that the system is operating → external Boolean operation;

All the pumps of the system → external Pump { } pumps;

Desirable property → property prop1 =

WHERE → forAll p in pumps suchThat p.room = "A"

WHEN → duringAny 1*year of operation

WHAT → check (count p.eFails) ≤ 2;

The true requirement → requirement req1 =

HOW WELL → check probability prop1.violated < 0.001;

WHY and HOW are expressed by models organisation

EDF

Example 1': Another Interpretation

Ambiguous! → When the system is in operation, the probability that a pump in room A fails more than 2 times a year shall be less than 0.1 %

All system pumps in room A → Pump { } pumpsInA = {p in pumps suchThat p.room = "A"};

WHAT → check count OR {p.eFails forAll p in pumpsInA} ≤ 2;

```
class Pump
   external Boolean failure;
   event eFails = when failure becomes true;
   external String room;
end Pump;

object coolingSystem
   external Boolean operation;
   external Pump { } pumps;
   Pump { } pumpsInA =
        {p in pumps suchThat p.room = "A"};

   property prop2 =
      duringAny 1*year of operation
      check count OR {p.eFails forAll p in pumpsInA} ≤ 2;

   requirement req2 =
      check probability prop2.violated < 0.001;

   end coolingSystem;
```

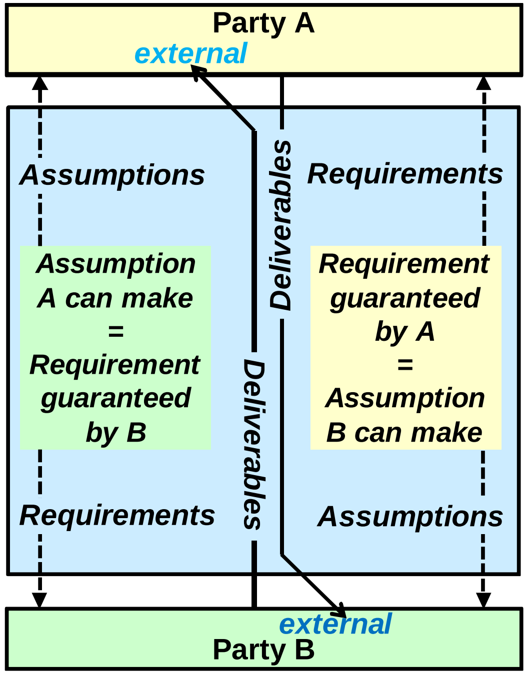
Example 2: Fault Tolerance

When there is no more than one failed component in the system, there shall be no spurious actuation

	partial class Component
Indicates the failure of the component	*Boolean* failed;
	end Component;
All the components of the system	*external* Component { } components;
	Boolean tolerance =
Number of failed components	*count* {e *in* components *suchThat* e.failed} ≤ 1;
Indicates whether the system should be active	*external Boolean* needed;
Indicates whether the system is really active	*external Boolean* active;
Desirable property: no spurious actuation	*property* prop3 = *when not* needed *ensure not* active;
The true requirement	*requirement* req3 =
	during tolerance *ensure not* prop3.*violated*;

Contracts

- **Formal specification of the mutual obligations among two or more FORM-L objects**
  - Party: one of the objects concerned
  - Deliverable: information one party provides to the others
  - Guarantee: requirement for one party, assumption for the others
- **Standard contract: template that can be instantiated with different sets of parties**
  - E.g., a supplier and its differents clients
- **Contract extensions: additional clauses to a contract**
  - Resulting from the detailing of solutions
- **A contract ties together "consenting" objects**
  - Mainly for top-down approaches
- **Contracts are powerful means for coordination and abstraction**



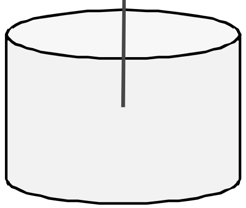
Bindings

- **Enable information transfer between models not knowing one another**
  - Without having to modify any of them
  - Some may be non-FORM-L models, or even engineering databases
  - For reuse and bottom-up approaches
- **Example : Determination of a functional state for a FORM-L requirements model**
  - From physical variables computed by a Modelica model
  - And from static characteristics specified in an architectural model in FORM-L
  - Using a library function

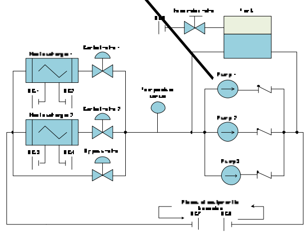
cavitates = HCavitate (npsh, pressure, flow)

```
property model req
  class Pump
    external Boolean cavitates;
    end Pump;
  external Pump{} pumps;
  external Boolean emergencyOp;
  requirement r1 =
    forAll p in pumps
    during not emergencyOp
    ensure not p.cavitates;
end req;
```

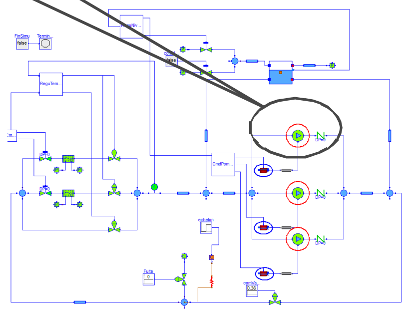
Requirements Model (FORM-L)



Functions Library (FORM-L)



Architecture Model (FORM-L)

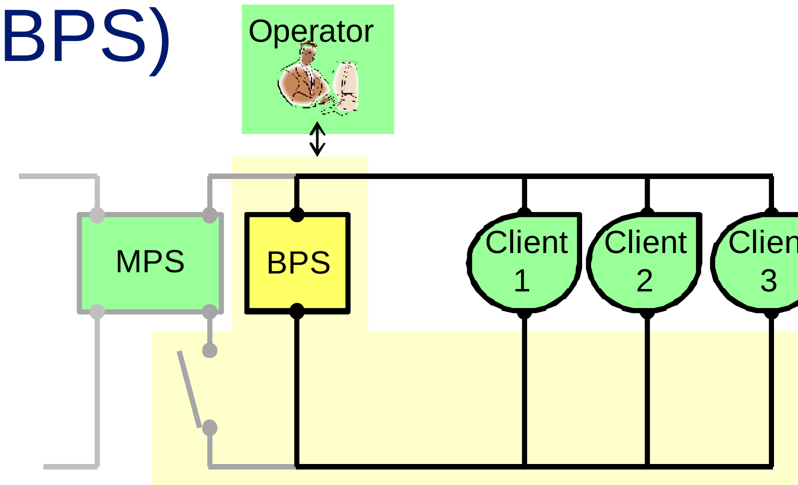


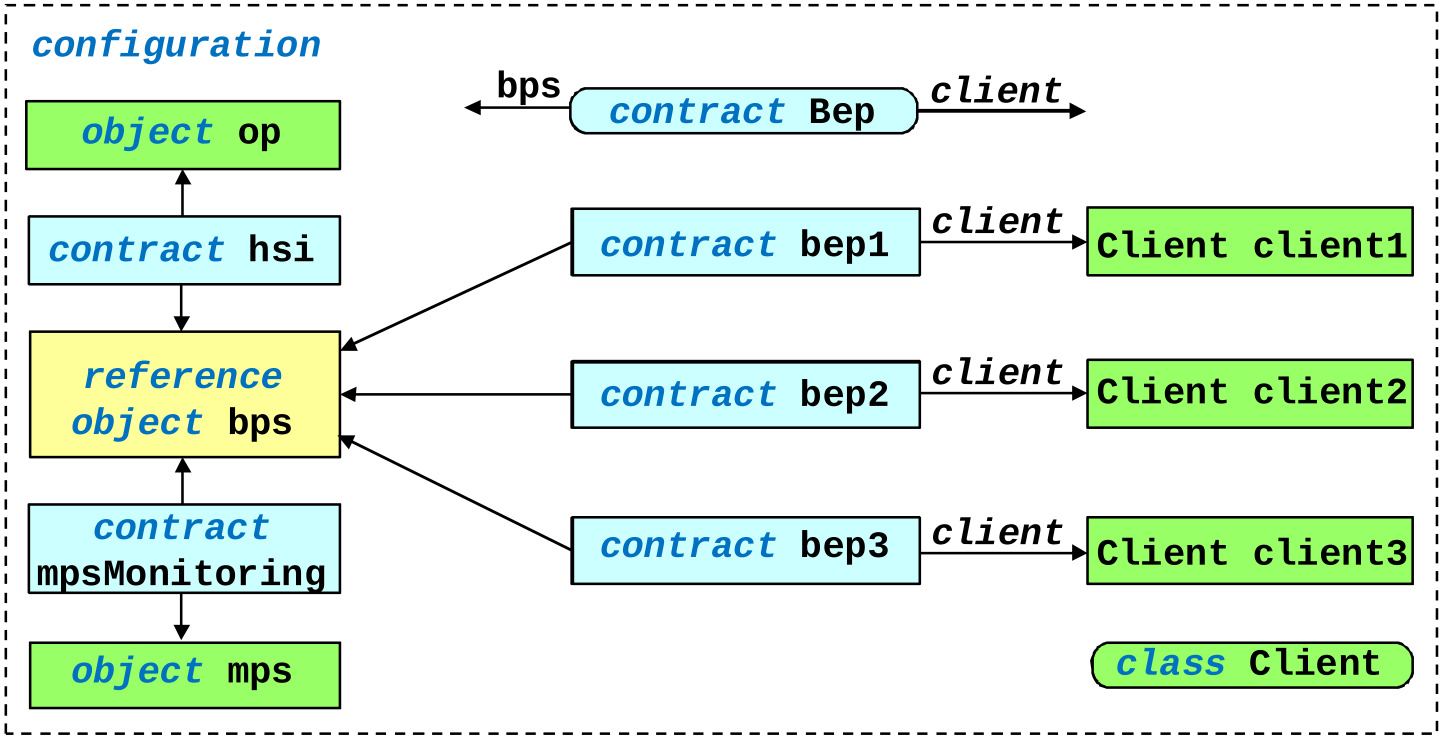
Behavioural Model (Modelica)

Overview

- **Motivation**
- **Quick introduction to FORM-L (FOrmal Requirements Modelling Language)**
- **Quick introduction to the associated method**

Example of Modelling Configuration: The Backup Power Supply (BPS)





Reference Model

- **Specifies the top-level behavioural requirements that will be the basis for the verification of solutions and their models**

A. Identify the environnement entities that interact with the system

- Other systems, human actors, the physical environment

B. Identify situations

- System states, environment entities states, operational goals, transitions
- Need to also address abnormal situations

C. Identify flows

- Fluids, information, events
- May depend on situations (e.g., agression and invasion in some abnormal situations)

D. Model the assumptions made by the system regarding its environment

- May also depend on situations

E. Model the requirements placed on the system by its environment

- Some requirements may be placed on the system directly
- May also depend on situations

Surrogate & Scenario Models

- **A reference model views its environment preferably through contracts**
- **In the first engineering phases, simple surrogate models just satisfying their respective contract may be used**
  - Preferable, since one should not make implicit assumptions
  - Effort focused on the system under study, reduction of modelling complexity, reduction of computing power for simulation
- **Test cases can be generated automatically**
  - With tools such as StimuLus (developed and marketed by ArgoSim)
  - Test cases generated randomly, but consistent with assumptions and definitions
- **To obtain cases of particular interest, scenario models may be used to guide the test case generator**
  - Additional assumptions
- **Later on, more accurate models may be used to study possible emergent behaviours**

EDF

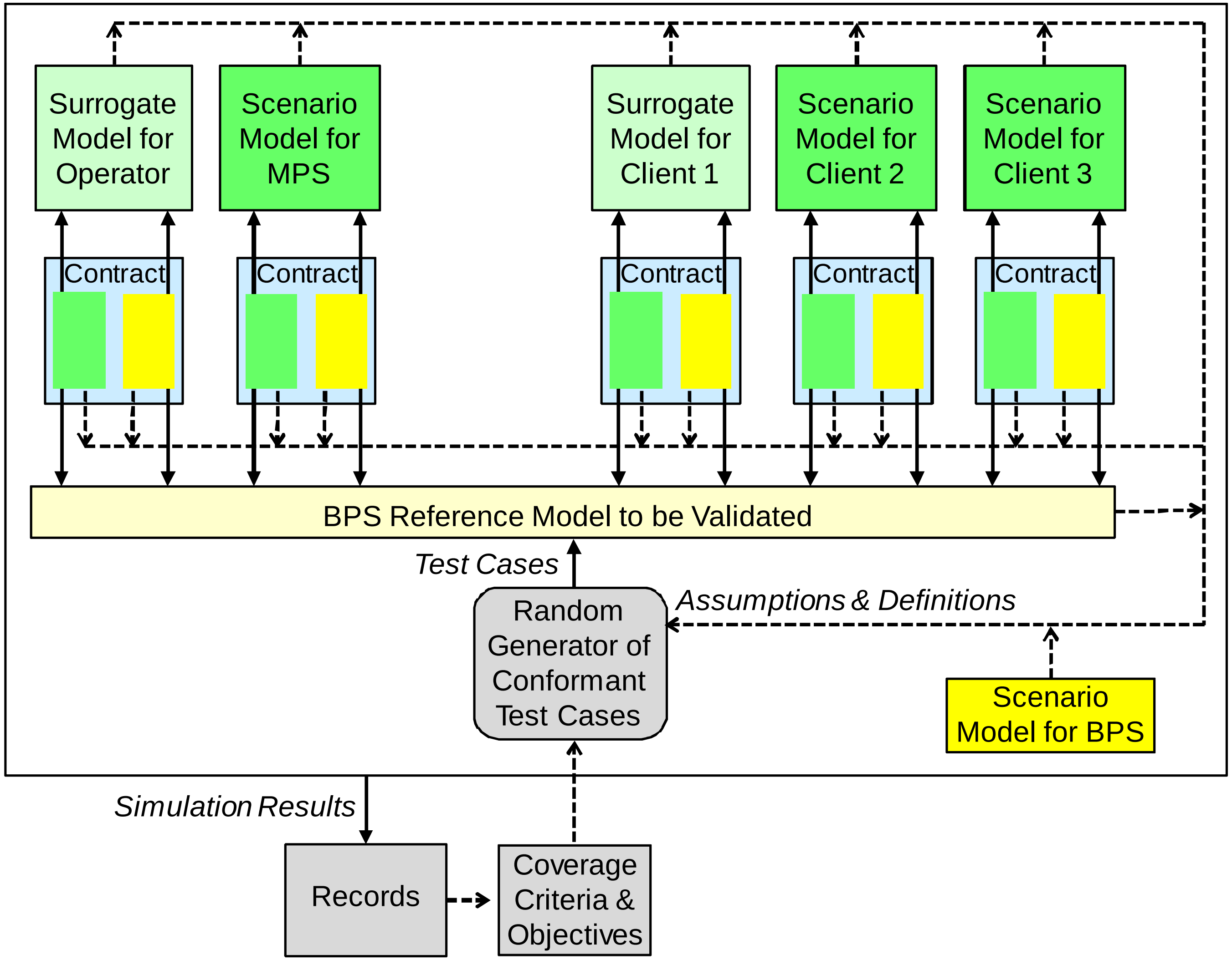
Validation of the Reference Model

- **Ensure that the reference model correctly represents what the authors have in mind and the real needs**
- **Reviews, inspections and analyses**
  - Agrement of other teams on their contracts with the system
  - Coverage and correct representation of relevant statements of input documents
  - Compliance with modelling rules
  - Consistency and freedom from contradictions
    - No solution in case of contradiction
- **Simulation, with automatically or manually generated test cases**
  - Verification that the model behaves as intended and reaches expected conclusions on requirements
- **Often, conflicting stakeholders expectations**
  - Simulation may help stakeholders understand the specified requirements and their effects
  - It may also help them decide whether the proposed compromises are acceptable

Test Coverage

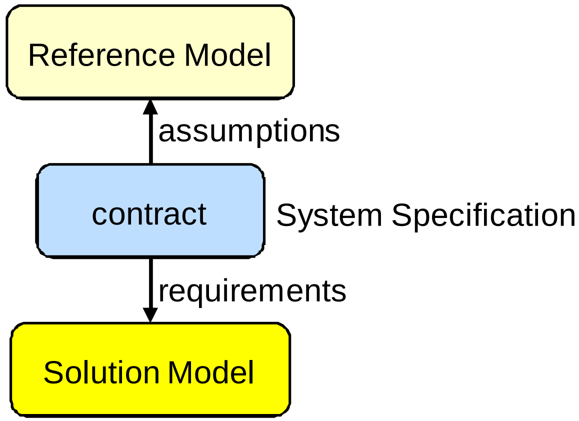
- **Many test cases are necessary to gain adequate confidence in a model**
- **Test coverage criteria and objectives to be collectively satisfied by the test cases**
  - Many criteria are possible, e.g., visiting each state of each automaton, taking each state transition of an automaton, etc.
- **Criteria and objectives could be specified as FORM-L requirements**
  - Not with respect to the system under study, but to the recorded simulation results
- **As simulation records accumulate, the test case generator can be guided to improve coverage**

Validation of the BPS Reference Model



Solution Models – System Specification

- **System specification: system description as one solution to the requirements of the reference model**
  - A project owner issues a tender specifying the system requirements
  - Different bidders reply, each with their own system specification
  - System still viewed as a black box, or as a dark grey box
- **Need to determine whether a system specification complies with the requirements**
  - Often far from straightforward
- **Contract between the reference model and a solution model**
  - The reference model views the system specification as a set of assumptions
    - The test case generator produces compliant behaviours to be checked against system requirements
  - The solution model views the system specification as a set of requirements
    - To be satisfied by further, more detailed solution models

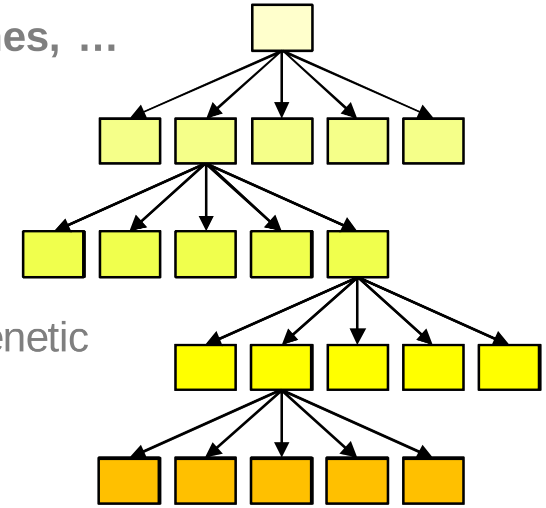


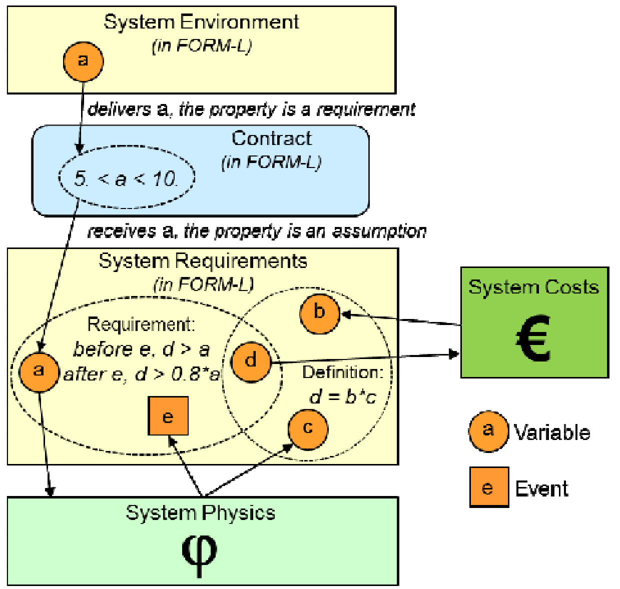
Solution Models – System Architecture

- **Once the system specification is verified, a system architecture can be developed**
  - Identifies the main components of the system
  - Places assumptions on components behaviours and interactions
    - Allocation of the requirements of the system specification to components
  - Contracts may be established between the architecture and its components, so that the assumptions made by the architecture are requirements for the components
  - Contracts may also be established between components
  - Enables early probabilistic analyses (safety, dependability)
- **Verification by simulation, with components behaviours consistent with assumptions**
  - Contract between the system specification model and the architecture model
  - Surrogate component models: no need to wait for detailed design solutions
- **Some components may be considered as systems of their own, and the same process is applied iteratively**

Optimisation

- **Competition, changing context, financial constraints, deadlines, …**
- **Need to innovate and thus to explore multiples solutions**
  - Preferably early in the engineering process
  - Manually developed solutions
  - Possible application of more systematic approaches such as genetic approaches
- **Diverse evaluation criteria**
  - Satisfaction of requirements
  - Cost of construction and profitability of operation (including maintenance)
  - Safety and security justification
  - …





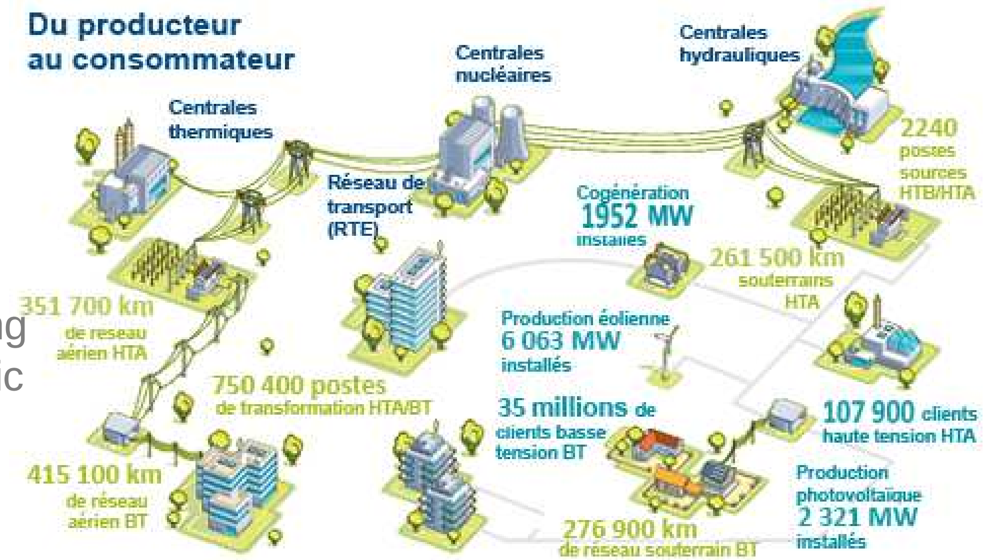
Solution Models – Deterministic Models and Implementation

- **At some point in the design process, a component may be represented by a deterministic model**
  - Model-in-the-Loop verification
- **At a further point, a component may be represented by an implementation**
  - Software-in-the-Loop, Hardware-in-the-Loop
- **In both cases, automatic generation of test cases and verification of test results**

EDF

Conclusion

- **Summary**
  - Enhanced engineering processes tested in EDF and Dassault Aviation case studies
  - Not presented here: multi-mode modelling, system state estimation, time domains, ...
  - Partial FORM-L implementation with Modelica and StimuLus libraries, but translation still manual
- **On-going work**
  - Development of a FORM-L compiler
  - Development of Graphical FORM-L (FORM-GL) based on graphical and textual boilerplates
  - Completion of support libraries, including one for FIGARO (failure and probabilistic analyses)
  - Bridges with SysML?



- **EDF exploitation perspectives**
  - Internal project for modelling and simulation of a national or continental power grid
  - Horizon 2020 proposal **HOLMES** (HOListic Modelling of cybEr-physical Systems)

Thank you for your attention

Any questions?