Modelling & Simulation of Complex Socio-Cyber-Physical Systems and Large Scale Systems of Systems

Along their Lifetime, a System Owner Standpoint

CSDM 2016
December 13-14, 2016

N. Thuy - EDF R&D
General Principles

- Most systems are **socio-cyber-physical systems (SCPS)**: you ignore it at your peril
  - Human actions
  - Automated controls
  - Process, geographic proximity, connectivity, ...

- Need to consider them in their **operational context**

- **Formal modelling** of assumptions, requirements and design
  - Focus on dynamic phenomena
  - Remove ambiguity of natural or semi-formal languages
  - Avoid overspecification
  - Reconcile innovation, and safety & dependability
  - Provide tool support
    - Simulation, automated test case generation, automated results verification, static analysis, optimisation in design and operation, 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 (divide et impera: divide and conquer)
  - 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

---

![Diagram of system lifecycle and coordination](image-url)
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

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

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

Deterninistic Model

Constraints Model

EDF
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 aggression and invasion
    - Example: crash of the Concorde

- Development of FORM-L in the framework of project
  - FOrmal Requirements Modelling Language
  - Can also address systems not yet designed
Modular Modelling

**Definition:**
\[ d = b \times c \]

**Requirement:**
before \( e \): \( d > a \)

after \( e \): \( d > 0.8 \times a \)

---

**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)**
- \( a \)
- Requirement:
  - before \( e \): \( d > a \)
  - after \( e \): \( d > 0.8 \times a \)
- \( b \)
- \( d \)
- Definition:
  - \( d = b \times c \)
- \( e \)

**System Costs & Revenues**
- \( \€ \)
  - \( a \) Variable (function of time)
  - \( e \) Event

**System Physics**
- \( \phi \)
Contracts

- 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 different 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

\[
cavitates = HCavitate(npsh, \text{pressure}, \text{flow})
\]

```
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;
```

Etemaker et al. 2009
System Requirements
Reference Model for the System Under Study

Specifies the top-level behavioural requirements that will be the basis for the verification of solutions

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., aggression 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
System Requirements
Surrogate & Scenario Models for the Environment

- A reference model views the entities of its environment preferably through contracts.

- In early phases, for these entities, use surrogate models, which just satisfy the entities' contracts:
  - Avoids making 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:
  - Randomly, but in consistency with assumptions and definitions.
  - With tools such as StimuLus (developed and marketed by ArgoSim).

- To guide the generator towards cases of particular interest, use scenario models:
  - Additional assumptions.

- Later on, more accurate models may be used to study possible emergent behaviours.
Example of Modelling Configuration: The Backup Power Supply (BPS)
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**
  - Agreement 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
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
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
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
Innovation, Optimisation, Safety, …

- 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
  - …

- **FMECA**: Failure Modes Effects & Criticality Analysis
  - Modelling of components / subsystems faults & failure modes
  - Simulation with faults injection and verification that safety requirements are not violated

- **STPA**: System-Theoretic Process Analysis
Conclusion

- **Summary**
  - Enhanced engineering processes tested in EDF and Dassault Aviation case studies
  - 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 textual and graphical 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
Thank you for your attention

Any questions?