Models Composition with FORM-L

N. Thuy - EDF R&D
February 3rd, 2016
Socio-Cyber-Physical Systems (SCPS)

- Cyber-physical systems integrate physical processes, computation and networking

- **Socio-cyber-physical systems** also have an important human aspect

- Large, complex SCPSs often have important societal implications
  - Need to ensure safety, dependability, performance, adaptability, …

- **Modelling can be of great help for optimisation and simulation-based verification**
  - Conventional behavioural modelling
    - Detailed description → for downstream systems engineering activities
  - **FORM-L (FOrmal Requirements Modelling Language)**
    - Generalisation: assumptions, requirements, overall solutions
    - Applicable to complex SCPSs and systems of systems
    - For all activities, including requirements specification & prospective studies

---

**ENVELOPES**
- Individual trajectory
- Constructions, upgrades, prospective studies
- Recent past, immediate future, far future
FORM-L in One Slide

WHY and HOW are expressed by models organisation

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

```FORM-L
class Pump
  external Boolean failure;
  external String location;
  event eFails = when failure becomes true;
end Pump;

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

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

requirement req1 =
  probability prop1.violated < 0.001;
end coolingSystem;
```

indicates pump failure
indicates in which room the pump is
external means that the value will be provided by another model (describing a solution)

indicates that the system is operating
All the pumps of the system

desirable property

WHERE
forAll p in pumps suchThat p.location = "A"

WHEN
duringAny 1*year and during operation

WHAT
check (count p.eFails) ≤ 2;

the true requirement

HOW WELL
probability prop1.violated < 0.001;
The Challenges of Complex Systems Modelling

- Modelling thriftiness: minimising the need for different modelling languages when addressing different engineering activities
- Ascertaining the correctness of models and simulation results
- Generation and assessment of a large number of cases
- Perenniality of models and simulation infrastructure
- **Models composability**: assembling models representing
  - Viewpoints of different disciplines
  - Different parts of the system (or system environment)
  - Different phases and activities of system development and operation
- In FORM-L, three main notions: **contracts**, **bindings** and **configurations**
Reference Models

- The first step is to develop a reference model for the system under study
  - The basis for the verification of solution models

- System as a black box

- Identification of environment entities having an influence on the system
  - Other systems, human actors, physical environment

- Identification and characterisation of flows
  - Fluids, events, information

- Identification of situations
  - System states, states of environment entities, operational goals, transitions

- Assumptions made from the system standpoint on environment entities
  - These may depend on situations

- Requirements placed on the system
  - Some may be placed by environment entities
  - May also depend on situations
Example: Intermediate Cooling System (SRI)

Active / Not active
Heat generated when active
Max internal temperature
Min RW temperature
Max flow

Client 1 (Heat Source)

Client 2 (Heat Source)

Client 3 (Heat Source)

Refreshed Water (RW)

Hot Water (HW)

Temperature in range 0.-27°C

Temperature difference less than 5°C

Operator

Orders

Information

Cold Water (CW)

Warmed Water (WW)

Source of Cold Water (SEN)

Source of Demineralised Water (SED)

DW Leaks

Demineralised Water (DW)

DW Request

Active / Not active
Heat generated when active
Max internal temperature
Min RW temperature
Max flow

Client 1

Client 2

Client 3

(SEN)

(SED)
Formal Contracts

- **A FORM-L contract** helps ensure that there is no misunderstanding between models and engineering teams. It specifies
  - The **models** that are parties to the contract
  - The **deliverables** (fluids and information) to be provided by each party
    - The other parties can access them via **external** declarations
  - The obligations (**requirements**) of each party
    - An obligation for one party is a right (**assumption**) for the others

- **A standard contract** is a template that can be applied to different parties

- **A contract can be extended** with additional clauses
  - When solutions are known in more detail

- **A contract is between “consenting” models**, whereas a **binding** is between models that do not know one another
  - Contracts are more likely to be useful in the **top-down** phases of systems engineering
  - Bindings are more likely to be useful in the **bottom-up** phases or to put together models independently developed by different disciplines
Example: Contract-Based Configuration for the SRI Reference Model

- pModel op
- contract hsi
- pModel sri
- contract deminW
- contract coldW
- pModel sed
- pModel sen
- supplier contract cooling1 client
- supplier contract cooling2 client
- supplier contract cooling3 client
- supplier contract coolig client
- pModel client1
- pModel client2
- pModel client3
Surrogate Models

- A reference model views an environment entity preferably through a contract.

- In the first phases of engineering, a simple surrogate model may be used:
  - It just needs to satisfy the contract.
  - One should not make implicit assumptions → a surrogate model should have all behaviours allowed by the contract.
    - General physical constraints (e.g., the continuity of analog signals) may need to be added.

- Cases can be generated automatically based on the surrogate model.

- This reduces the need for big, resource-hungry, difficult to simulate models.

- When more detailed information is available on the entity and needs to be taken into consideration, more precise models may be used:
  - Including behavioural models and implementations.
Validation of the Reference Model

- Ensure that the reference model correctly represents what the authors have in mind

- **Reviews, inspections and analyses**
  - Agreement of other teams on their contracts with the system
  - Coverage and correct representation of relevant statements from the design basis documents
  - Compliance with the modelling rules specified by the engineering organisation
  - Consistency and freedom from contradictions
    - No possible solution in case of contradiction

- **Simulation, with automatically or manually generated test cases**
  - Verification that the model behaves as intended and reaches expected conclusions regarding requirements
Scenario Models

- Need to verify behaviour in cases of particular interest unlikely to be produced rapidly by random generation

- A scenario model is a property model that specifies additional assumptions to guide the test case generator

- Various configurations are possible
  - One specific scenario model per environment entity, acting as a surrogate model
    - No coordination between the entities, unless there are contracts between them
  - One scenario model for multiple environment entities, acting as a surrogate model for each.
    - This facilitates coordination between entities

- Different configurations may have different scenario models for the same entity
Test Coverage

- Many test cases are necessary to gain adequate confidence in a model

- Specification of **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 information recorded and accumulated during simulation runs

- As simulation information accumulates, inputs can be provided to the test case generator to produce cases that increase coverage

- Issue not fully studied in MODRIO
  - Will need to be addressed more completely in a subsequent project
Configuration for Validating SRI Reference Model

- SRI Reference Model to be Validated
- Random Generator of Conformant Test Cases
- Scenario Model for SRI
- Scenario Model for SEN
- Scenario Model for SED
- Scenario Model for Client 1
- Scenario Model for Client 2
- Scenario Model for Client 3
- Surrogate Model for Operator
- Surrogate Model for Client
- Surrogate Model for SEN
- Surrogate Model for SED
- Surrogate Model for Client 1
- Surrogate Model for Client 2
- Surrogate Model for Client 3

Simulation Results
Records
Coverage Criteria & Objectives

Contract
Scenario Model for SEN
Scenario Model for SED
Scenario Model for Client 1
Scenario Model for Client 2
Scenario Model for Client 3
Surrogate Model for Operator
Surrogate Model for Client
Surrogate Model for SEN
Surrogate Model for SED
Surrogate Model for Client 1
Surrogate Model for Client 2
Surrogate Model for Client 3

Test Cases

Configuration
Various Models for the Same Entity
Solution Models – System Specification

- **System specification**: precise system description, still considered as a black box, as one solution to the requirements of the reference model
  - A project owner issues a tender stating the system requirements
  - Different bidders reply, each with their own system specification

- Need to determine whether a specification complies with the requirements
  - A process that is 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 generator produces complying behaviours to be checked against requirements
  - The solution model views the system specification as a set of requirements
Solution Models – System Architecture

- Once the system specification is verified, a system architecture can be developed
  - The assumptions made to verify the specification against the requirements are considered as requirements for a system architecture
  - A contract may be established between the system specification and the architecture

- An architecture
  - Identifies the components that constitute 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

- It is verified in simulation with components behaviours consistent with assumptions
  - This can be done with surrogate models, without having to wait for detailed design solutions and models

- Some components may be considered as systems of their own, and the same process is applied iteratively
Solution Models – System Architecture

System Reference Model

- assumptions

contract

- System specification

requirements

System Solution Model
Solution Models – System Architecture
Solution Models – System Architecture

- System Reference Model
  - Assumptions
  - Contract
  - System specification
  - Requirements

- System Solution Model
  - Assumptions
  - Contract
  - Requirements
    - Comp1 Reference Model
    - Comp2 Reference Model
    - Comp3 Reference Model
Solution Models – System Architecture
Looking for an Optimal Solution

- Pressure from competition, budget constraints, planning constraints, multiple suppliers solutions, ...

- Need to **innovate** and thus to evaluate **multiple solutions**
  - Preferably **as early as possible** in the engineering and design process
  - Manually developed alternatives
  - Possible application of genetic approaches

- **Various assessment criteria**
  - Compliance with behavioural requirements
  - Cost and profit during operation (including maintenance)
  - Justification for safety and security
  - Socio-human factors
  - ...

[EDF Logo]
Reusing Existing Solutions

- A design process usually does not follow a fully top-down approach
  - It often tries to find a path to reuse existing, well-proven solution elements
  - These could be property or behavioural models, or even implementations

- To avoid modifying an existing model, one may use bindings as bridges with project-specific models or with other existing models
Behavioural Models - Implementations

- At some point, the design of a component may be represented not by an architecture model, but by a **behavioural model**
  - Model-in-the-Loop verification
  - A behavioural model needs sometimes to be expressed in physical terms that are not those of property models, which tend to be more functional
  - Bindings may be useful even when a behavioural model is specifically developed for a project

- At a further point, a component may be represented by an **implementation**
  - Software-in-the-Loop, Hardware-in-the-Loop
  - Closed loops

- In both cases
  - Automatic generation of test cases and verification of test results
Conclusion

- A configuration specifies a meaningful set of models that together can be used for simulation or optimisation

- The same real world entity can be represented by various models
  - Representing different stages of the engineering process
    - From requirements to detailed design
  - Representing the views of different engineering disciplines
  - Representing abstractions used in the analysis of other entities
    - Surrogate models, scenario models

- A configuration can contain multiple models for the same entity

- Contracts can be used to formally specify what a model expects or ensures with respect to other models
  - Contracts between peer entities
  - Contracts between engineering disciplines
  - Contracts between requirements and solutions

- Bindings are used to transfer information between models that have been developed independently
Thank you for your attention

Any questions?