INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

D2.1.2 – Properties modelling method

WP2.1 – Properties modelling language
WP2 – Properties modelling and Safety

MODRIO (11004)

Version 1.0

Date 01/05/2016

Authors

Thuy NGUYEN EDF

Executive summary

This document is deliverable MODRIO D2.1.2 *Properties Modelling Method*. It suggests a methodology for the modelling of potentially complex cyber-physical systems along their engineering process. The resulting models should also be useable for the operation of the system, and also for retrofits and upgrades.

The methodology proposed here has been applied and is illustrated in [Thuy Nguyen, "MODRIO D8.1.3 Part 1 *The Backup Power Supply (BPS) System*". and [Jardin A., Thuy Nguyen, "MODRIO D8.1.3 Part 2 *The Intermediate Cooling System (SRI)*".]. It relies on a very large part on the FOrmal Requirements Modelling Language (FORM-L) proposed in [Thuy Nguyen, "MODRIO D2.1.1 FOrmal Requirements Modelling Language (FORM-L)", EDF technical report, H-P1A-2014-00550-EN.]. It also relies on the modelling architecture proposed in [Bouskela D., "MODRIO Project - Modeling Architecture for the Verification of Requirements - MODRIO deliverable D2.1.1", EDF technical report, H-P1C-2014-15188-EN, 2015.].

In particular, it suggests that separate sets of models are developed for requirements specification, overall design and detailed design. Whereas the first two sets are composed of property models, the last one is generally includes behavioural models based on a precise knowledge of the characteristics and behaviour of the components constituting the system. In addition to these models, property and / or behavioural model of the environment of the system may be used in order to provide a context for, and interact with, the system models. Usually, all these models are developed at different stages of the systems engineering lifecycle, by different persons who have different viewpoints, different backgrounds and different modelling constraints.

Summary

Acronyms

BPS	Backup Power Supply
FORM-L	FOrmal Requirements Modelling Language
HIL	Hardware-In-the-Loop
MPS	Main Power Supply
SRI	Intermediate Cooling System
WP	Work Package

Glossary

Assumption	Property that is supposed to be satisfied: simulation scenarios assume / ensure that it is satisfied.
Conditional Probability	***`real`*** value (not a function) in the [0., 1.] range that states the probability of an event occurring at least once during a specified time period (a CTL). The time period specifies the condition
Condition-Based Property	A condition-based property states that a given ***`condition`*** expression is ***`true`*** at a specified time locator, possibly with a given duration constraint
Continuous Time Domain	Time domain where time is perceived continuously. There is only one continuous time domain
Continuous Time Locator	Expression that specifies one or more time intervals
Discrete Time Domain	Time domain where time is perceived not continuously but at specific, defined instants. Different discrete time domains may specify different sets of instants
Discrete Time Locator	Expression that specifies one or more time instants
Event	Named expression that characterises the occurrences of a fact that has no duration
Event-Based Property	An event-based property states that an event does or does not occur at a specified time locator, possibly with a count constraint
Finite State Automaton	Function of time the values of which belong to an enumerated set
Guard	Property that states the conditions that must be satisfied for a model to be valid
Probability	Function of ***`time`***, the result of which is a **real** value in the [0., 1.] range. Its value at ***`now`*** (the current instant) represents the probability that a given ***`event`*** has already occurred at least once
Property	See Condition-Based Property and Event-Based Property
Requirement	Property that must be satisfied: it is the objective of simulation to verify that it is not **violated**
Time Domain	Part of a model where objects have the same perception of time
Time Domain Interface	Part of a model where events and conditions in one time domain is made perceptible by objects belonging to another time domain
Time Instant	Position in time that has no duration
Time Interval	Period of time that has a duration (as opposed to an event, which is instantaneous and has no duration)
Time Locator	See Continuous Time Locator and Discrete Time Locator

1. Introduction

1.1. Context and objectives

This document is deliverable MODRIO D2.1.2 *Properties Modelling Methodology*. It suggests a methodology for the modelling of potentially complex cyber-physical systems along their engineering process. The resulting models should also be useable for the operation of the system, and also for retrofits and upgrades.

1.2. Links with other WPs

The methodology proposed here has been applied and is illustrated in [8] and [9]. It relies on a very large part on the FOrmal Requirements Modelling Language (FORM-L) proposed in [10]. It also relies on the modelling architecture proposed in [7].

In particular, it suggests that separate sets of models are developed for requirements specification, overall design and detailed design. Whereas the first two sets are composed of property models, the last one is generally includes behavioural models based on a precise knowledge of the characteristics and behaviour of the components constituting the system. In addition to these models, property and / or behavioural model of the environment of the system may be used in order to provide a context for, and interact with, the system models. Usually, all these models are developed at different stages of the systems engineering lifecycle, by different persons who have different viewpoints, different backgrounds and different modelling constraints.

1.3. Notations

FORM-L statements are written in a **`bold Courier New`** font.

FORM-L keywords are in ***`blue italics`***.

The names of property models, of classes and types begin with an upper-case letter.

The names of objects and variables begin with a lower-case letter.

Event names begin with an 'e'.

The names of configurations are in upper-case.

The use of a predefined FORM-L library defining physical constants of various types is assumed. For example:

-	**`1*s`**	1 second
-	**`10*s`**	10 seconds
-	**`1*h`**	1 hour
-	**`100*ms`**	100 milliseconds
-	**`1000*kw`**	1000 kilowatts

2. System Requirements vs. System Specification

In the following, expression *system requirements* is used to designate the behavioural requirements that the system must satisfy. Some are placed on the system by entities of its environment. Others may be placed directly on the system, without any mediation from its environment. Expression *system specification* is used to designate a precise description of system behaviour, still considering it as a black box, as one possible answer to system requirements.

The distinction between the two is important. For example, a project owner issues a tender stating the system requirements. Different bidders then reply, each with their own system specification. The project owner needs to identify the bids (if any) that satisfy the requirements, a process that is not always straightforward.

3. Models and Objects

The notions of *model* and *object* serve two related but different purposes:

- A model is a basic unit for composing *configurations*, a configuration being a set of models that constitutes a whole that can be simulated, analysed or optimised. Thus, the organisation into models depends in large part on the engineering and behavioural modelling methodology applied by a project.
- An object represents an entity of the real world. Thus, the organisation into objects essentially depends on the system, functional and architectural analyses made by that same project.

Sometimes, but not necessarily, a model can be viewed as representing a single main object.

4. Reference Models, Requirements Specification Models

At a very early stage in the system engineering process, the first step in the behavioural modelling and simulation process is to develop and validate a *reference model*. This model will be used in subsequent steps to verify the correctness and adequacy of *solution models*. Thus, it must be a property model, whereas solution models may be either property models or behavioural models.

In the reference model, the system under study is represented as a *black box* object, i.e., its composition and design in terms of components and subsystems are unknown. What it does is identify the entities in the system environment that have an influence on the system. These entities are for example other systems interacting with the system under study, various human actors (e.g., operators or maintenance staff), or environmental conditions significant to the system (e.g., ambient conditions, seismic conditions, fire, flooding, etc.). The model represents the assumptions made on the behaviour and the interactions of these environment entities with the system under study. It also represents the requirements that they may place on the system. Lastly, it specifies any requirements placed directly on the system, without the mediation of any environment entity.

Thus, the reference model is a *system requirements model*. However, since a system may contain subsystems that can be considered as systems of their own with their own system requirements models, the notion of reference model highlights the fact that for a given engineering team working closely together, that model is the top-level model.

5. Contracts

A complex system or a system of systems may be viewed as a hierarchy of systems. Different systems may be assigned to different engineering teams. Each team then develops a reference model for each system under its responsibility. To ensure appropriate coordination and avoid misunderstanding, *contracts* may be established to formalise the obligations and expectations of each system with respect to the others. This notion is summarised in Figure 5-1.

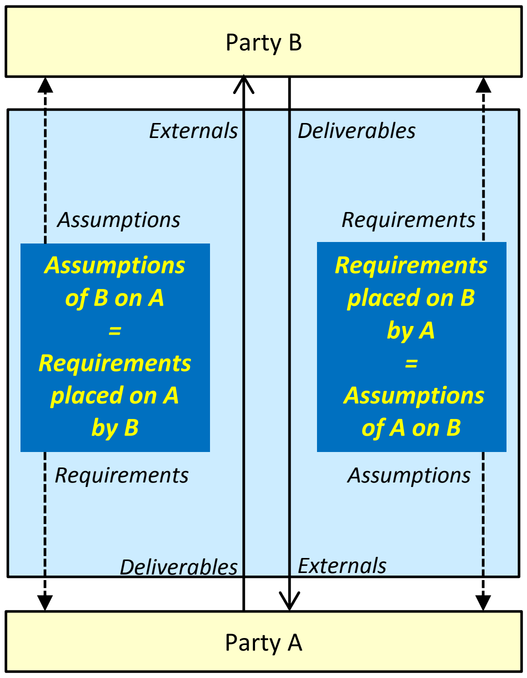
Party B
Externals
Deliverables
Assumptions
Requirements
Assumptions of B on A = Requirements placed on A by B
Requirements placed on B by A = Assumptions of A on B
Requirements
Assumptions
Deliverables
Externals
Party A

Figure 5-1
Contract between parties A and B. Items declared external by one party is provided as a deliverable by the other party. Assumptions made by one party on the other are requirements for that other party.

A contract is established between two or more *parties*. After identifying the parties, the first objective of a contract is to specify the *deliverables* (in terms of variables, events and objects) that each party is responsible for and must provide to the others. A party that needs to access a deliverable from another party just declares it as *external*.

Also, one party may expect certain behaviour from another party, and expresses it in the form of assumptions. These are considered as requirements by the concerned party. This is often the case in a client-supplier relationship: the client assumes a certain behaviour from the supplier (in order to satisfy its own requirements), but that behaviour is a requirement for the supplier. This is also the case when progressing from one design phase to the next: once a tentative solution (expressed in terms of assumptions) is thoroughly verified and validated at an early design phase, then it can be viewed as a set of requirements for the subsequent design phases.

6. Surrogate Models

The environment entities of a given system may be viewed through a contract, even those that are not systems (e.g., human actors and environmental conditions). In the first phases of the system engineering process, an environment entity can be represented by a simple *surrogate model*: Such a model just needs to satisfy the entity's part of the contract. Indeed, it is worthwhile that the system under study makes no additional, implicit assumptions on its environment than what is formally specified in the contracts. Thus, a surrogate model could and should be property model allowing all behaviours not forbidden by the contract.

In subsequent design phases, if more detailed information regarding a given environment entity is available and needs to be taken into consideration, more precise models (including behavioural models) may be used.

7. Scenario Models

A simulation run may be based on a randomly generated test case that is just compliant with the assumptions[1]. However, 'obvious' general physical constraints (e.g., the continuity of analogue signals) are often not explicitly represented in assumptions. Also, one often needs to verify behaviour in particular cases that are unlikely to be produced rapidly by generating test cases purely randomly.

A *scenario model* is a property model that specifies additional assumptions in order to guide the test case generator so that it produces test cases of interest. Various configurations are possible. For example:

- A configuration may have one specific scenario model per entity of the system environment, acting as a surrogate model for that entity. In this case, there is no coordination between the entities, unless the configuration includes contracts between them.
- A configuration may have one scenario model for multiple entities of the system environment, acting as a surrogate model for each. This facilitates coordination between them.
- Different configurations may have different scenario models for the same entity to test different situations.

8. Validation of the Reference Model

The validation of a reference model aims at ensuring that it correctly represents what the authors want to represent. The process should include reviews, inspections and analyses, in particular to verify that:

- The other teams in charge of entities of the system environment agree with their respective contracts with the system, and indeed include these contracts in their own modelling.
- All relevant statements in design basis documents concerning the system have been addressed and adequately represented in the reference model.
- The reference model complies with any modelling rules that might have been specified by the systems engineering organisation.
- The reference model is consistent and does not contain any contradictions (in which case there would be no possible solution).

The process could also include simulation, with automatically or manually generated test cases complying with the assumptions specified either directly in the reference model on in its contracts. One can then verify that for each case, the model behaves as intended, and in particular reaches expected decisions regarding the specified properties and requirements.

9. Test Coverage

In general, many test cases are necessary in order to gain adequate confidence in a model. To increase that confidence, one often specifies test coverage criteria 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.). Such criteria could be specified as FORM-L requirements addressing not the system under study, but the information recorded during simulation. As new test cases are simulated and recorded information accumulates, one can provide new inputs to the test case generator so that it generates cases that increase the coverage of the required criteria. This issue of test coverage has not been fully studied in the MODRIO project, and will need to be addressed in a subsequent project.

A typical overall models organisation for validating a reference model is shown in Figure 9-1.

[1] For example using StimuLus, a software tool developed and marketed by ArgoSim. It can generate random signals complying to constraints (here, assumptions).

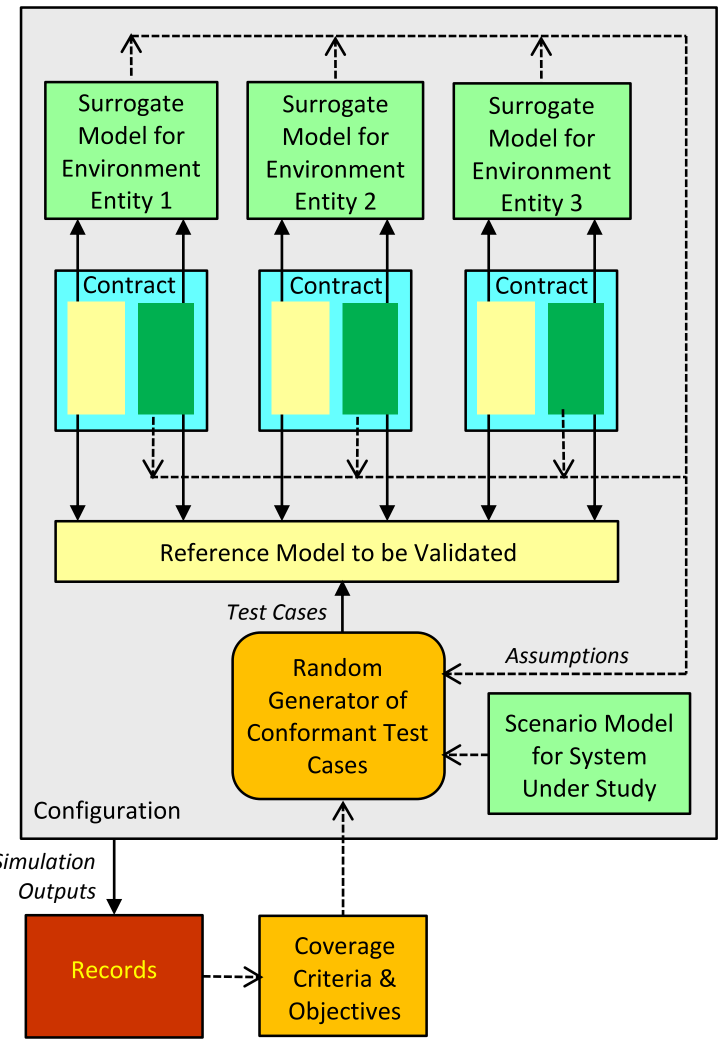
Surrogate Model for Environment Entity 1
Surrogate Model for Environment Entity 2
Surrogate Model for Environment Entity 3
Contract
Contract
Contract
Reference Model to be Validated
Test Cases
Random Generator of Conformant Test Cases
Assumptions
Scenario Model for System Under Study
Configuration
Simulation Outputs
Records
Coverage Criteria & Objectives

Figure 9-1
Example of configuration for the validation of a reference model. Boxes in light blue represent contracts. Boxes in light or dark green express assumptions that are provided to the test case generator. Boxes in yellow express requirements. The one specifying the coverage criteria and objectives also provides inputs to the test case generator in order to achieve the coverage objectives.

10. Verification of Solution Models

Once the reference model is validated, then one can develop and model one or more solutions. As stated in Section 2 *System Requirements vs. System Specification*, the first level solution is a system specification. To be able to use a framework similar to the one of Figure 9-1 (see Figure 10-1), the requirements of the system specification are modelled as assumptions. The test case generator will then generate behaviours consistent with the system specification, which can then be verified against the system requirements of the reference model.

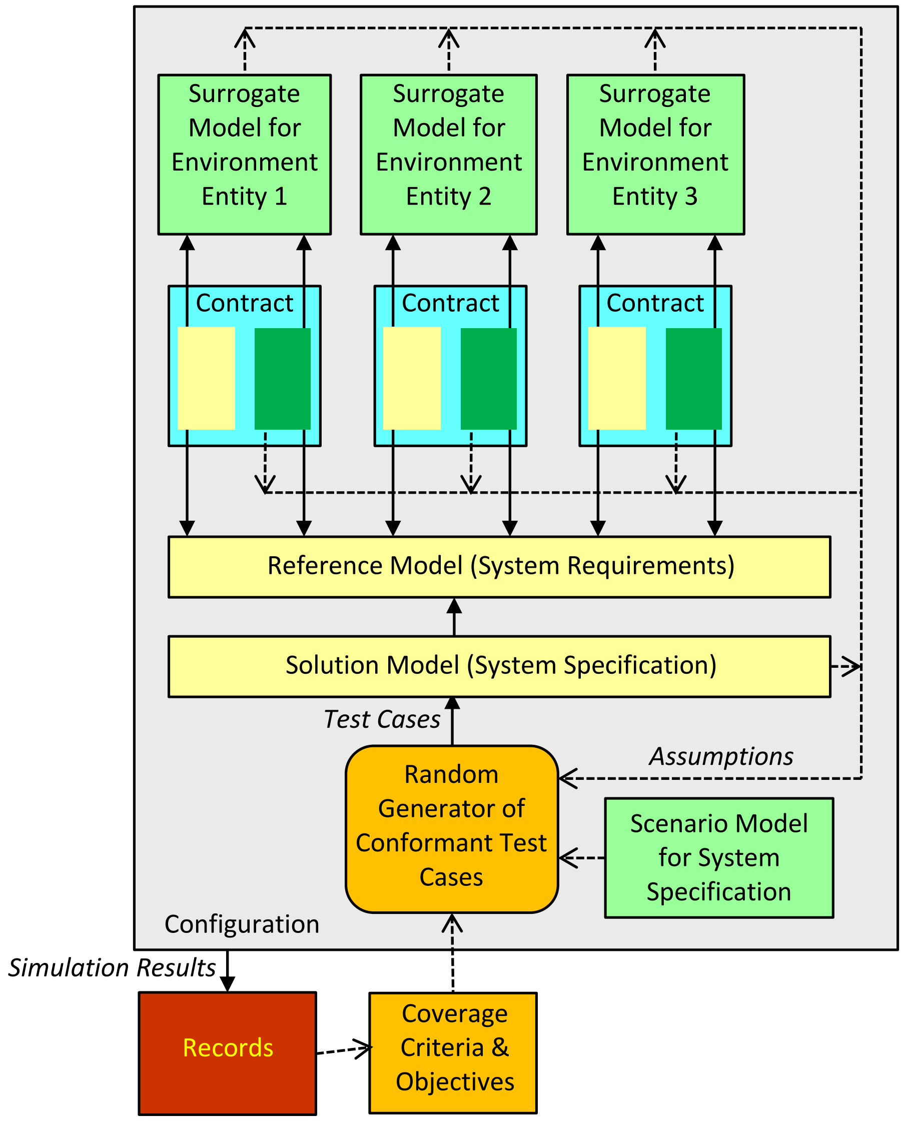
Surrogate Model for Environment Entity 1
Surrogate Model for Environment Entity 2
Surrogate Model for Environment Entity 3
Contract
Contract
Contract
Reference Model (System Requirements)
Solution Model (System Specification)
Test Cases
Assumptions
Random Generator of Conformant Test Cases
Scenario Model for System Specification
Configuration
Simulation Results
Records
Coverage Criteria & Objectives

Figure 10-1
Example of configuration for the verification of a solution model stating system specification.

At a later stage of the engineering process, an overall design is developed, identifying the system main components and specifying their interactions and requirements for each. An associated property model can then be used to verify that that overall design indeed complies with the system specification and satisfies the system requirements (see Figure 10-2).

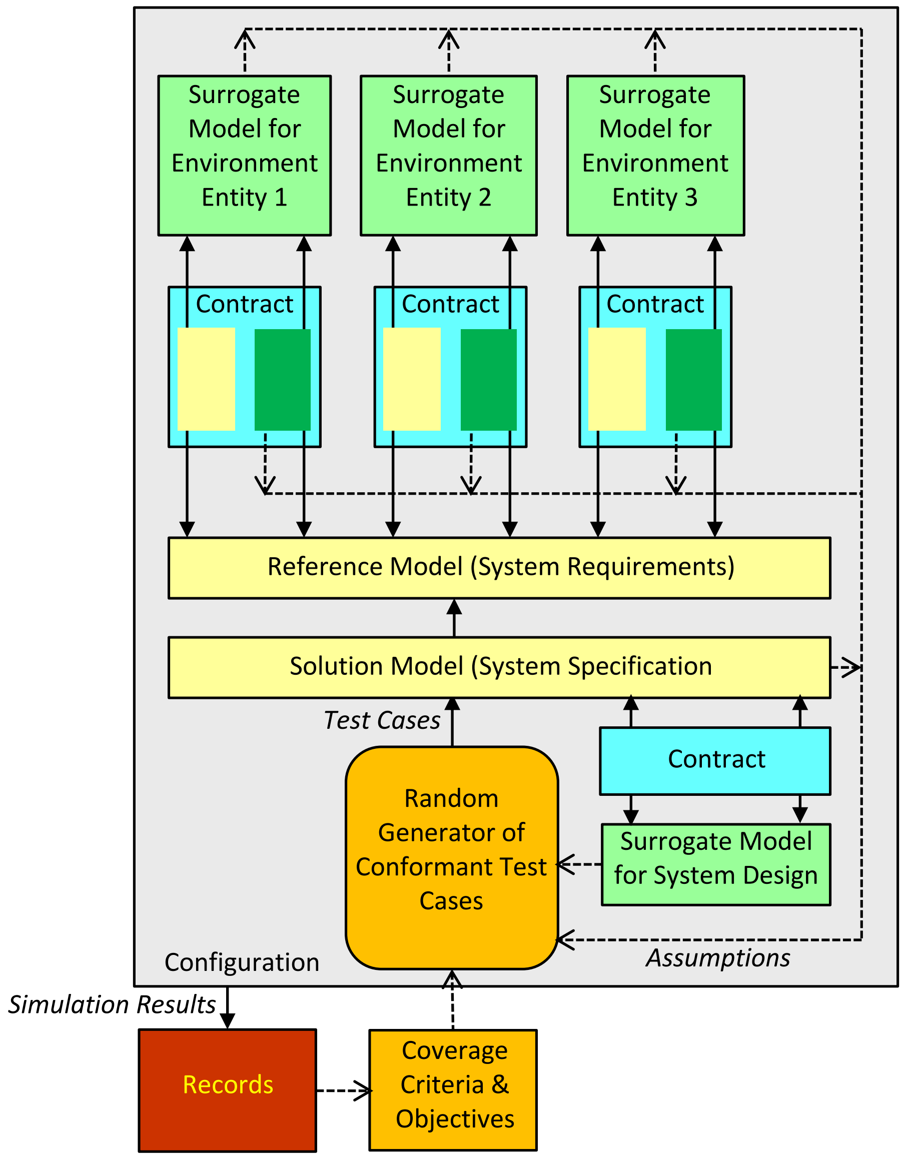
Surrogate Model for Environment Entity 1
Surrogate Model for Environment Entity 2
Surrogate Model for Environment Entity 3
Contract
Contract
Contract
Reference Model (System Requirements)
Solution Model (System Specification
Test Cases
Contract
Random Generator of Conformant Test Cases
Surrogate Model for System Design
Assumptions
Configuration
Simulation Results
Records
Coverage Criteria & Objectives

Figure 10-2
Example of configuration for the verification of a solution model stating system overall design.

At yet a later stage, where detailed design choices are made regarding the components and detailed information is available regarding actual response times, accuracies, failure modes, etc., a behavioural modelis developed and included the simulation.

11. Modelling Organisation

The organisation shown in [7] is typical for requirements modelling, where one needs to represent not only the system under study (hereafter called the **system**), but also the entities that constitute its **environment**, and the **contracts** that the system may have with some of these entities.

At that stage of the systems engineering process, the system and the entities of its environment are considered as black boxes. The modelling focuses mainly on the information exchanged between these boxes, and on the corresponding requirements and assumptions.

Requirements and assumptions may depend on the **state** of the system, on the **states** of the different entities of the environment, and on the **operational goals** that are placed on the system at a given instant. A combination of system state, environmental states and operational goals is called a **situation**.

12. Modelling Patterns

12.1. Requirements vs. Desirable Properties

When specifying requirements in a formal manner, it is often necessary to make a distinction between desirable properties and genuine requirements. In particular, one needs to consider that real systems are made of components subject to failure and that desirable properties cannot be satisfied under all failure conditions.

In this case study, the following approach has been taken:

- A desirable condition or event is expressed in the form of a property:

```
  property px = ...;
```

- The real requirement is specified in the form of a requirement that specifies under which failure conditions the corresponding desirable property must be satisfied[1]:

```
  requirement rx = during tolerance check not px.violated;
```

- Additional requirements may be added to put a limit on the probability that the desirable property is not satisfied. It sometimes useful to make a distinction between the probability of failure on demand (PFD) and the frequency of failure during continuous operation:

```
  requirement pFDx =    // Limit on probability of failure on demand
    when demand check probability px.violated < ...;

  property pf = duringAny period check not px.violated;
  requirement pFx =     // Limit on failure rate during continuous op.
    during operation check probability pf.violated < ...;
```

12.2. 'Thick Boundaries'

For real systems, requirements regarding transitions (e.g., modification of discrete variables, signalling of events) based on analogue inputs cannot be based on sharp, infinitely thin boundaries. In particular, one needs to consider that sensors and analogue-to-digital conversions have limited, finite accuracy and may drift. Even without these effects, sharp boundaries could lead to undesirable 'chattering'. Thus, transitions based on analogue inputs generally need to specify 'thick boundaries' (in value and time) and hysteresis (see Figure 12-1).

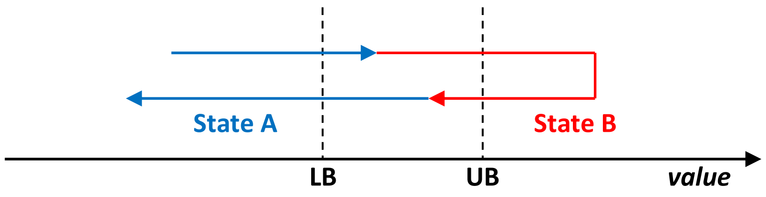

Figure 12-1
Thick boundary - Between the lower bound (LB) and the upper bound (UB), both states are legitimate

12.3. 'Grace Periods'

In the same spirit, one should consider that real systems cannot react instantaneously to their inputs. Thus, a formal requirement specification must always give the system a 'grace time' within which it is allowed to react, even if the informal, natural language requirement mentions none.

[1] `not px.violated` means that `px` may be either `notTested` or `satisfied`., but not `violated`.

12.4. End of Simulation Scenario

The modelling is valid only until the end of the reset sequence that terminates the first use of the *BPS*. After that, the simulation should be stopped. On the reverse, simulation should not be stopped before that, otherwise incorrect conclusions could be drawn regarding the satisfaction of requirements.

13. Conclusion

These methodological guidelines have been developed based on the experience from case studies described in [8] and [9]. They will likely to be extended as more experience is gained in other case studies, with the use of new, more powerful tools, and when it is applied by more users.

14. References

[1] Fritzson P., “Principles of Object-Oriented Modelling and Simulation with Modelica”, IEEE Press, 2003.

[2] Harel D., “Statecharts: A Visual Formalism For Complex Systems”, in Science of Computer Programming, vol. 8, pp. 231-274, 1987.

[3] Schamai W., “Model-Based Verification of Dynamic System Behavior Against Requirements”, Dissertation No. 1547, Linköping University, 2013.

[4] Jardin A., “EuroSysLib sWP7.1 - Properties Modelling”, EDF technical report, H-P1C-2011-00913-EN, 2011.

[5] Thomas E., “EuroSysLib WP7.1 – Dysfunctional: Use Cases and User Requirements”, Dassault Aviation technical report, DGT116083, 2008.

[6] Thomas E., Chastanet L., “EuroSysLib WP7.1 – Dysfunctional: Use Cases and User Requirements”, Dassault Aviation technical report, DGT116083B, 2009.

[7] Bouskela D., “MODRIO Project - Modeling Architecture for the Verification of Requirements - MODRIO deliverable D2.1.1”, EDF technical report, H-P1C-2014-15188-EN, 2015.

[8] Thuy Nguyen, “MODRIO D8.1.3 Part 1 *The Backup Power Supply (BPS) System*”.

[9] Jardin A., Thuy Nguyen, “MODRIO D8.1.3 Part 2 *The Intermediate Cooling System (SRI)*”.

[10] Thuy Nguyen, “MODRIO D2.1.1 FOrmal Requirements Modelling Language (FORM-L)”, EDF technical report, H-P1A-2014-00550-EN.