Simulation validation using the compatibility between Simulation Model and Experimental Frame

Damien Foures, Vincent Albert, Alexandre Nketsa

Simulation validation using the compatibility between Simulation Model and Experimental Frame

Damien FOURES[1,2], Vincent ALBERT[1,2], Alexandre NKETSA[1,2]
[1]CNRS, LAAS, 7 avenue du colonel Roche, BP 54200, 31031 TOULOUSE, FRANCE
[2]Univ de Toulouse, UPS, LAAS, 31400 TOULOUSE, FRANCE
dfoures@laas.fr, valbert@laas.fr, alex@laas.fr

Keywords: Discrete event simulation, experimental frame, measure, validation, V&V

Abstract

This work illustrates the possibilities associated with the concept of experimental frame in the domain of simulation. The experimental frame noted "EF" is used to define the environment in which a model will evolve. EF and model are extracted from specifications of the system understudy. Formal language has been utilized for designing EF & model such that model checking techniques can be employed. We have chosen "I/O automata", applying the frame to the model implies incompatibilities with parts of the model. In this contribution we detect these incompatibilities to measure it. Our method will allow a set of metrics evaluate the validity of simulations according to goals of simulation user-defined.

1. INTRODUCTION

Since 60's, Systems Engineering (SE) has evolved. There is general agreement to define the SE as a interdisciplinary scientific approach, whose goal is to formalize and apprehend the design of complex systems successfully. Complex systems have invaded our environment, closest are in our pocket, farthest are at some 18 billion kilometres from Earth (space probe Voyager 1).

A system is said to be complex, when it includes a large number of heterogeneous entities in interaction and includes calculations requiring either time or complex operators. This complexity poses hurdles to any possibility of knowing the system behavior in its entirety by an observer. These complexity problems have led to use two approaches for systems validation, to answer the question: "It is the good system?".

The first method consists of developing the system before submission to a set of tests. Then it is manually applied a set of stimuli to the system while observing his behavior (time-costly procedure). This method allows exploring only a small part of the system.

The second one, more complex could not appear until the arrival of computing. It consists in building a model of the system and simulate it to perform validation activities as early as possible in the development cycle of the system. This second method reduces development costs and time to market by detecting problems earlier in the design. For example, the latest Peugeot 208 (a french car)has required twice less physical tests than the 207, giving more prominence to the simulation [Clapaud 2012]. Due to a growing interest in simulation and the increase of computer computational power, modelling and simulation (M&S) is now a fully fledged discipline, with its own problems and its own research areas.

In order to obtain the confidence of engineer and development teams in the simulation results, it is essential to validate these results. Our research is primarily concerned with the following issues:

- Assess the validity of a simulation model based on the level of abstraction and objectives of simulation.
- Attach information to clarify the validity range of the simulation model and possible means to remain in the defined area.
- Define metrics to quantify the efficiency of simulation considering models involved and the expected results.

We use the concept of experimental frame originally introduced by [Zeigler 1976] to answer these issues in the field of simulation.

This paper will address these problematic and briefly describe the proposed solutions to address them. It will focus on the presentation of the simulation, the concept of experimental frame and related formal methods. We will then present our approach and methodology through SiML, a SysML profile for simulation. Section 5, puts forward the metrics that can be derived by our approach. We will conclude this paper by research perspectives.

2. SIMULATION

This section acts as preliminary to present simulation and key concepts of *experimental frame for modelling and simulation* as they were introduced by B.P.Zeigler in [Zeigler 1976]. Simulation is used throughout the life cycle, from customer requirements capture to system validation as shown in Figure 1.

Simulation is divided into four stages according to project progress. The transition from one simulation platform to another requires more or less effort, depending on the type of

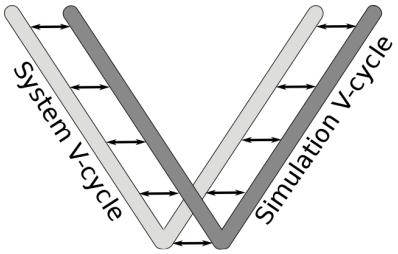

Figure 1. Simulation in the development cycle.

systems developed and tests which can be achieved. Figure 4 illustrates the chronology of platforms. It is difficult to establish a precise location for each method. Depending on the activities and policies of the development team and the type of system to be deployed, these phases will be forced to slide, and thus appear more or less early in the development cycle.

- 1-MITL: "Model in the loop" consists of recovering a model describing the behavior of the system to be validated by simulation.
- 2-SITL: "Software in the loop" consists of recovering a program that implements the model in the target language. This implementation can lead to bias due to model implementation constraints. A new phase of validation and/or verification is required to ensure the semantic equivalence.
- 3-CITL: "Controller in the loop" consists of validating the behavioural equivalence of the program after controller integration. All control systems is still simulated.
- 4-HITL: "Hardware in the loop" consists of replacing successively simulated systems by physical systems. The environment remains simulated.

Our work proposes the use of formal and semi-formal approaches to improve the process of V&V of the "Model in the loop" platform . The MITL platform wants to be independent of the actual system, which means that the language used are not specific to a given architectural solution. The developer can focus on the functional aspects without worrying about compatibility with physical components and thus providing to simulation "users" a previous model which describes the behavior of the system.

The simulation user is in charge of validating the system. To do this, he describes the assumed system environment on the basis of requirements to validate simulation assumptions. This phase of environmental modelling can be done in parallel with the modelling phase of the system, or completely independently as shown in Figure 2. Starting from a common set of knowledge (expert) and specifications, the user describes evolution of environment model and simulation assumptions. This allow, through this experimental frame definition, take into account model usage requirements reducing consequently the model field. These restrictions or deviation must be qualified or quantified by measures, we call it metrics.

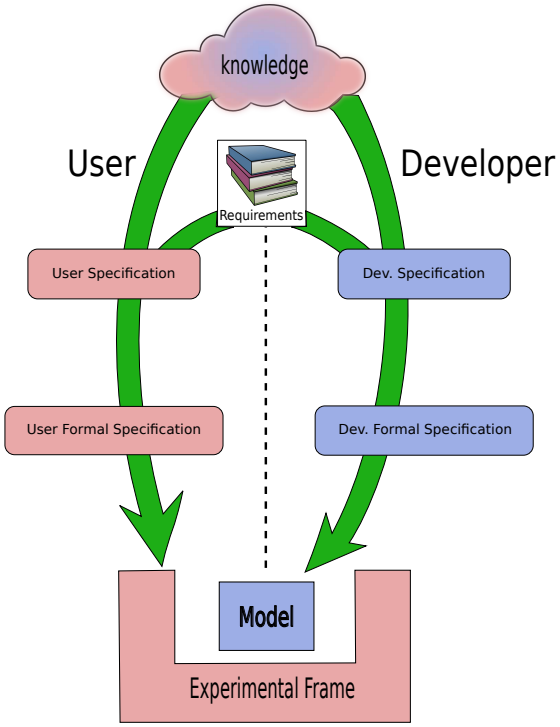

Figure 2. Model / Experimental frame Development

According to validation needs, the environment (experimental frame) or the model will be more or less detailed, which leads to the use of modelling and simulation environments with different levels of abstraction depending on V&V objectives.

The experimental frame features three components as shown in Figure 3.
• A *generator*, which generates a set of inputs segments for the model or the system.
• A *transducer*, which observes and analyses the output segments of the model or the system.
• A *acceptor*, which selects the data of interest to the model or the system while monitoring whether the experimental conditions are complied with.

The experimental frame provides a first complete model of the environment.

To assist the simulation user in his task, we introduced a methodology associated with SysML profile, called *Simulation Modeling Language* (SiML).

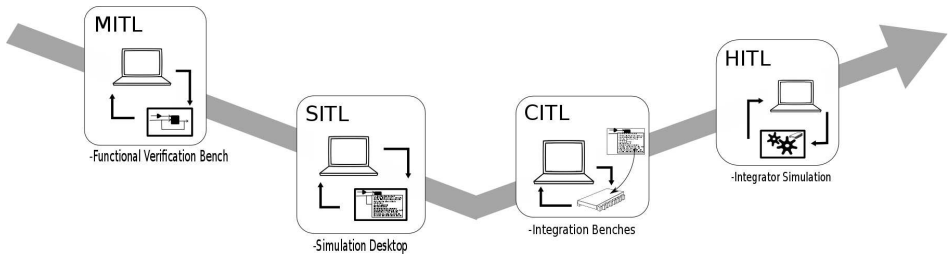

Figure 4. Simulation platforms chronology

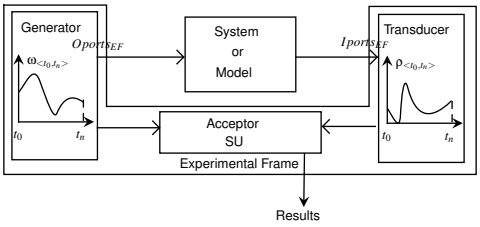

Figure 3. The experimental frame and its components

3. A PROFILE FOR SIMULATION

Improving the validation of the simulation, and more generally the V&V, goes through the use of formal / semi-formal methods. By observing current needs, we realize easily that a major problem for adopting these methods lies in the lack of methodology. Which formalism to use? In which phase of the development cycle? Experts able to answer these questions are rarely in the industry. It seems essential to propose a language combining the concept of experimental frame to assist the use of formal methods and thus improve the V&V chain of the MITL platform .
Using a new language involving semantics and syntax completely unknown does not seems to be appropriate. The current size of conception teams of complex systems makes it difficult and costly to adopt new practices. The possibilities offered by UML (Unified Modeling Language)make it a good alternative allowing *profile* creation. Profiles allow you to extend UML for a specific domain through extension mechanisms (stereotypes) that enrich the semantic and syntax of the language. This is the selected solution to develop a simulation oriented profile. This profile is shown in Figure 6 (Appendix A).

It describes the EF concepts and simulation needs (usage assumptions, modelling assumptions abstraction level, etc.). Its purpose is to provide simulation user to support to describe its objectives and simulation assumptions in semi-formal way. This formalism allows a set of predefined measures to be provided and then gives the user feedback on the appropriateness of the simulation. It could be guided on the choice of the most appropriate simulation model based on its validation objectives. Predefined behavior assumptions will determine which model is most suitable to meet simulation objectives.

By following the steps of Figure 2, the simulation user needs to validate, or redefine the model and its assumptions iteratively. Using the profile allows a systematic application of our metrics, therefore automatic. Guided by the user, profile will provide necessaries informations for their calculations. Formal methods allow a larger set of metrics to be used. It is therefore decided to use automata to describe the behavior of the *generator* and *transducer*. We shall see in the next section how it is possible to connect them, to enrich our metrics, with the completeness property of model checking [1].
The assumptions have a significant role in the SiML profile . Associated with one or more automata (*generator, transducer, model*), applying an assumption will restrict their behaviors. The assumption that an event "a" never appears will reduce the possible evolutions of an automaton taking into account the event "a". Associated with a variable, an assumption may restrict this variable, but may also restrict other variables (eg Let A, B, C three variables: If $B = A + C$, a limitation on $A \rightarrow$ a limitation on B).

We classify assumptions into three categories:
• Rational: Assumptions fall within the physics.
• Modeling: Assumptions caused by technical limitations or modelling choices.
• Operational: Assumptions resulting from system specification, simulation specification and objectives.

Although, our work initially focused on the simulation user, it appears that the use of SiML profile by system developers can significantly increase possibilities of metrics and their relevance to the simulation validation. The interaction between the EF and the system model is performed through compatibility between automata describing the *model*, the *generator* and the *transducer*. The following section illustrates our approach through an example and describes the coupling of automata which leads obtained metrics.

[1] The model checking explores the model state space to see if it meets the specifications described through a temporal logic.

4. METHODOLOGY

Our method has initially three automata. Compatibility between these three models complicates understanding without involving other concepts, the study will be limited to the compatibility between two *extended deterministic finite automaton* which we will call through this paper *automaton* .This part derives restriction from article [Foures et al. 2012].

Automaton: An *automaton* is a 6-tuple: $A = \langle X, Y, S, s_0, \delta_x, \delta_y \rangle$ o:

- X is a finite set of input events;
- Y is a finite set of output events;
- S is a finite set of states, $s_0 \in S$ is the initial state;
- $\delta_x : S \times X \rightarrow S$ is *external state transition function* that defines how an event changes states;
- $\delta_y : S \rightarrow Y^{\phi} \times S$ is the *external state transition function* that defines how an input event changes a state, $Y^{\phi} = Y \cup \{\phi\}$ and $\phi \notin Y$ define *null event*.[2]

It is possible to connect two *automata* to make automata network. *Automata network* is a 6-tuple: $C = \langle X, Y, D, C_{xx}, C_{xy}, C_{yy} \rangle$ o:

- X(res. Y)is finite set of input (res. output) events, such that $X \cap Y = \varnothing$.
- $D = \{M_i\}$ is a finite set of names of subcomponents.
- $\{M_i\}$ is an index set of sub-components. M_i can be either an atomic or coupled automaton.
- $C_{xx} \subseteq X \times \bigcup_{M_i \in D} X_i$ s a set of input couplings where X_i is a set of input couplings where $i \in D$.
- $C_{xy} \subseteq \bigcup_{M_i \in D} X_i \times \bigcup_{M_i \in D} Y_i$, is a set of internal relations of the coupled component.
- $C_{yy} = \bigcup_{M_i \in D} Y_i \rightarrow Y \cup \{\varepsilon\}$, is a set of output couplings, where Y_i is the set of output events of subcomponent $i \in D$.

4.1. Examples

An example of our analysis method for *"experimental frame - model"* compatibility is given hereafter .

■ **Phase 1** SiML completion.
The first step is to complete the SiML profile using requirements. For example: • Simulation assumption: the variable z will never happen in this environment ($\neg z$).• Rational assumption: The variable A is always less than 50 ($\{A|A \in \mathbb{I}; -\infty \leq A \leq 50\}$). • ... .
By successive stages the profile allows requirements to be formalized to achieve a formal specification of the simulation environment and the model (as suggested in paragraph B). All assumptions will restrict the use of the model. The final formalisms choice to describe modelling assumptions has not yet been subject of detailed studies.

We believe that the use of OCL constraints [OMG], properties of temporal logic [Di Giampaolo et al. 2010], mathematical relations, pattern modelling [Abid et al. 2011] and automation mechanisms of LTL properties [Nikora and Balcom 2009] can be used to define them efficiently.
■ **Phase 2** Modeling of the environment.
During this phase the user will describe simulation scenarios, stimuli sequences that he will implement (*generator*) and the observable assumed behavior of the model (*transducer*).
Formally: A coupled automaton $C_{12} = \langle X, Y, D, C_{xx}, C_{xy}, C_{yy} \rangle$ where: $X = Y = \{\varnothing\}$,$D = A_1, A_2$, $C_{xx} = \{\varnothing\}$, $C_{xy} = \{(a!, a?), (b!, b?), (c!, c?)\}$,$C_{yy} = \{\varnothing\}$, where A_1, A_2 are two automata representatives respectively specification of the frame and the model: • $A1 = \langle X_1, Y_1, S_1, s_{0_1}, \delta_{X_1}, \delta_{Y_1} \rangle$ where $X_1 = \{\varnothing\}$, $Y_1 = \{a!, b!, c!\}$, $S = \{0, 1, 2\}$, $s_{0_1} = 0$, $\delta_{X_1}() = \{\varnothing\}$, $\delta_{Y_1}(0, a!) = 2$, $\delta_{Y_1}(0, c!) = 1$, $\delta_{Y_1}(1, b!) = 2$ and • $A2 = \langle X_2, Y_2, S_2, s_{0_2}, \delta_{X_2}, \delta_{Y_2} \rangle$ with state variables $A \in \mathbb{I}; 0 \leq A \leq 100$ where $X_2 = \{a?, b?, c?, d?, z?\}$, $Y_2 = \{\varnothing\}$, $S = \mathbb{I} \times \{0, 1, 2, 3\}$, $s_{0_2} = (0, 0)$, $\delta_{X_2}((0, A), z?) = (1, 100)$, $\delta_{X_2}((0, A), a?) = (2, A + 10)$, $\delta_{X_2}((1, A), b?) = (2, A + 10)$, $\delta_{X_2}((1, A), c?) = (3, A)$, $\delta_{X_2}((2, A), d?) = (3, A)$, $\delta_{Y_2} = \varnothing$. $(A1, A2)$ can be represented graphically, see phase 2 in Figure5.

There is a cycle between Phase 1 and Phase 2 in order to complete the *SiML* model. Modeling will perhaps generate new hypotheses. For example, the user can add new assumptions to reduce a variable domain and limit the state space of the experimental frame. The modeller may also be forced to restrict the model in view of technological constraints.
■ **Phase 3** Perimeter compatibility.
The compatibility of I/O is a prerequisite for the use of simulation in accordance with the experimental frame. Compatibility should include the name, informations, type, range and accuracy. This accuracy is function of many parameters. These assumptions may also affect the compatibility.

For example: A variable $\{X_{EF} | X_{EF} \in \mathbb{I}; 0 \leq X_{EF} \leq 70\}$ can be compatible with $\{X_M | X_m \in \mathbb{I}; 0 \leq X_M \leq 50\}$ if an hypothesis defined under certain usage conditions $0 \leq X \leq 30$. If compatibility is not complete, each conflict is shown to the user and discordant inputs/outputs are "corked" (replaced by internal events).
■ **Phase 4** Models exploration.
Model (resp. the experimental frame) is totally explored to

[2] δ_y can be split into two functions: the output function :$\lambda : S \rightarrow Y$ and the internal transition function $\delta_{int} : S \rightarrow S$

know all the possibilities associated with the model (resp. the experimental frame) (eg total number of states, variable resolution, ...). This phase is not associated with any hypothesis, it is a complete exploration.
Formally: This is the set of words m recognized, causing a change from the initial state to a state marked by an automaton A. $L_M(A_1) = \{m \in (X_1 \cup Y_1)^* | \delta^*_{X_1 \cup Y_1}(s_0, m) \in S_1\}$.

■ **Phase 5** Automata composition.
We carry parallel product called parallel composition (denoted by //) in automata theory. The EF and all assumptions are associated with this product to restrict the model. Parallel product reveals common transitions of automata, but can represent the evolution of the model that does not belong to the alphabets iteration.
Phase 3 may seem unnecessary, but it sets the compatibility statically. The hypothetical size of models is important, it is essential to perform a static compatibility test before exploring the models and dynamic compatibility (time costly). Using the example of phase 3, if two variables have incompatible validity domain, it is good to detect statically, rather than find it out after exploring several thousands of states. The user can still accept the inconsistency of phase 3 to understand in further detail with dynamic exploration of phase 5.

■ **Phase 6** Analysis phase.
Compatibility results are given to the simulation user. A set of metrics (detailed in the next paragraph) give a simulation confidence index. It could be chosen to refine assumptions and repeat a cycle of simulations.

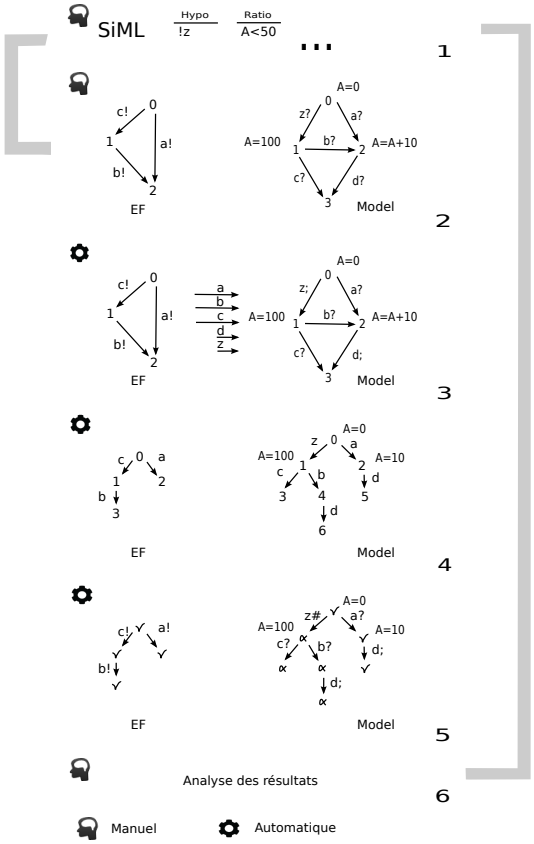

Figure 5. Methodology

5. METRIC

By definition, a metric is a way to find the distance between two points. Applied to our method, it seeks to measure the distance between the behavior of the simulation and the behavior specified by the user of the simulation. Measurements are then used to quantify the confidence that the user can grant to the simulation. Some metrics are known and recognized. For example, the calculation of "good properties" of a Petri net [Murata 1989]. In our approach, the experimental frame can reduce the possibilities of the model, these properties would be impacted and would not provide more information. It is possible to say a metric is quantifiable or qualifiable. Without going into details, a simple example permit to understand these two concepts: the water temperature can be quantified through the set of real numbers (250K, 383K, ...) or qualified (solid, gaseous, ...). Depending on the objective of simulation, it can be important to know the water temperature, but sometimes a simple information about its state is sufficient. The reader will understand intuitively that this abstraction is significant, the state space and the system complexity are greatly reduced.
This study does not attempt to list and define the set of metrics necessary to validate the simulation. As ever, the purpose of simulation, the type of simulated system, the simulation "accuracy" will lead to the use of different metrics. A compromise "*computation time/accuracy of the metric*" is mandatory.
Several works propose metrics for embedded systems. In [Karlsson et al. 2008] they use the formal approach to improve the coverage simulations, they are mainly interested in coverage metrics by calculating the number of places or transitions from a Petri net. The aim is to validate the model and not the validity of the simulation. The method includes two phases: firstly, they generate a set of stimuli, and if a too important part of the model is uncovered, they explore through model checking.
Others contributions [Chidamber and Kemerer 1994, Brito e Abreu and Melo 1996, Lorenz and Kidd 1994] proposed sets of structural metrics such as "*Coupling between object classes (CBO)*", "*Polymorphism Factor (POF)*" or "*SIZE2*" (the number of attributes and the numbers of local methods defined in a class).

We distinguish four types of measures: quantifiable, qualifiable, "A priori" (before execution), "Posteriori" (after execution). These metrics are measured from the superposition of the complete exploration of models (Phase 4) and the re-

sulting automaton composition (Phase 5), others based on information provided through the SiML profile. Following is the list of metrics that we are already capable to extract from the SiML profile. A thorough study of the possibilities and their formal definition is in progress.

■ A priori:

• quantifiable: The percentage of I/O compatible in relation to the domain of the I/O, to the unit,...

• qualifiable: Completeness index: If the entire SiML model who define I/O is completed imply the completeness is good.

■ Posteriori:

• quantifiable:: Coverage $M_{/EF}$ is the percentage of states explored in the model through the experimental frame, in relation to the total number of possible states.

• qualifiable: Confidence Index is the confidence in the simulation. Assumptions and the EF behavior limit the model to one execution trace explored at 100% imply a high confidence index. Assumptions and the behavior of the EF partially limit the model imply a limited confidence.

6. CONCLUSION

In this article we covered the concepts of simulation and validation of the simulation, which show two different issues. One seeks to validate the model while the other seeks to validate the method validation (improving confidence in the simulation in view of simulation objectives). The presentation of the methodology and SiML profile showed two approaches to restricted the model. The use of different kinds of assumptions allow to have constraints that restrict the model (eg rational assumptions) and others like the generator, that restrict a set of sequences of events in the automaton. The use of SiML profile by all actors in the development cycle allows a better control of the simulation and its validity. It is important to formally define the set of metrics to help to validate the simulation. The issue of taking into account the time currently poses problems with states explosion but is essential for an industrial use. Experience has shown that the scaling is always a delicate moment. This factor was considered since the beginning of the study; it is an important step that should be taken into account. The use of qualitative reasoning [Kuipers 1994] seems to be an interesting way to limit the complexity of the simulation, limiting the state space variables and therefore the resulting automaton.

7. ACKNOWLEDGEMENT

We are grateful to Vikas Shukla, whose discussions and comments helped to improve this paper.

REFERENCES

N. Abid, S. Dal Zilio, and D. Le Botlan. Verification of Real-Time Specification Patterns on Time Transition Systems. Technical report, 2011.

F. Brito e Abreu and W. Melo. Evaluating the impact of object-oriented design on software quality. In *Software Metrics Symposium, 1996., Proceedings of the 3rd International*, pages 90–99. IEEE, 1996.

S. R. Chidamber and C. F. Kemerer. A metrics suite for object oriented design. *Software Engineering, IEEE Transactions on*, 20(6):476–493, 1994.

A. Clapaud. Psa : deux fois moins de tests physiques pour concevoir la peugeot 208. 2012.

B. Di Giampaolo, G. Geeraerts, J. Raskin, and N. Sznajder. Safraless procedures for timed specifications. In *8th International Conference on Formal Modelling and Analysis of Timed Systems*, 2010.

D. Foures, V. Albert, A. Nketsa, et al. Formal compatibility of experimental frame concept and fd-devs model. In *9th International Conference on Modeling, Optimization & SIMulation*, 2012.

M. H. Hwang and B. Zeigler. Reachability graph of finite and deterministic devs networks. *IEEE Transactions on Automation Science and Engineering,*, 2009.

D. Karlsson, P. Eles, and Z. Peng. Model validation for embedded systems using formal method-aided simulation. *Computers & Digital Techniques*, 2008.

B. Kuipers. *Qualitative reasoning: modeling and simulation with incomplete knowledge*. MIT press, 1994.

M. Lorenz and J. Kidd. *Object-oriented software metrics: a practical guide*. Prentice-Hall, Inc., 1994.

T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 1989.

A. P. Nikora and G. Balcom. Automated identification of ltl patterns in natural language requirements. ISSRE'09, 2009.

OMG. Omg object constraint language (ocl), superstructure. *OMG specification*, pages 1–256.

B. Zeigler. *Theory of modelling and simulation*. A Wiley-Interscience Publication. Wiley, 1976. ISBN 9780471981527. URL `http://books.google.fr/books?id=M-ZQAAAAMAAJ`.

A APPENDIX

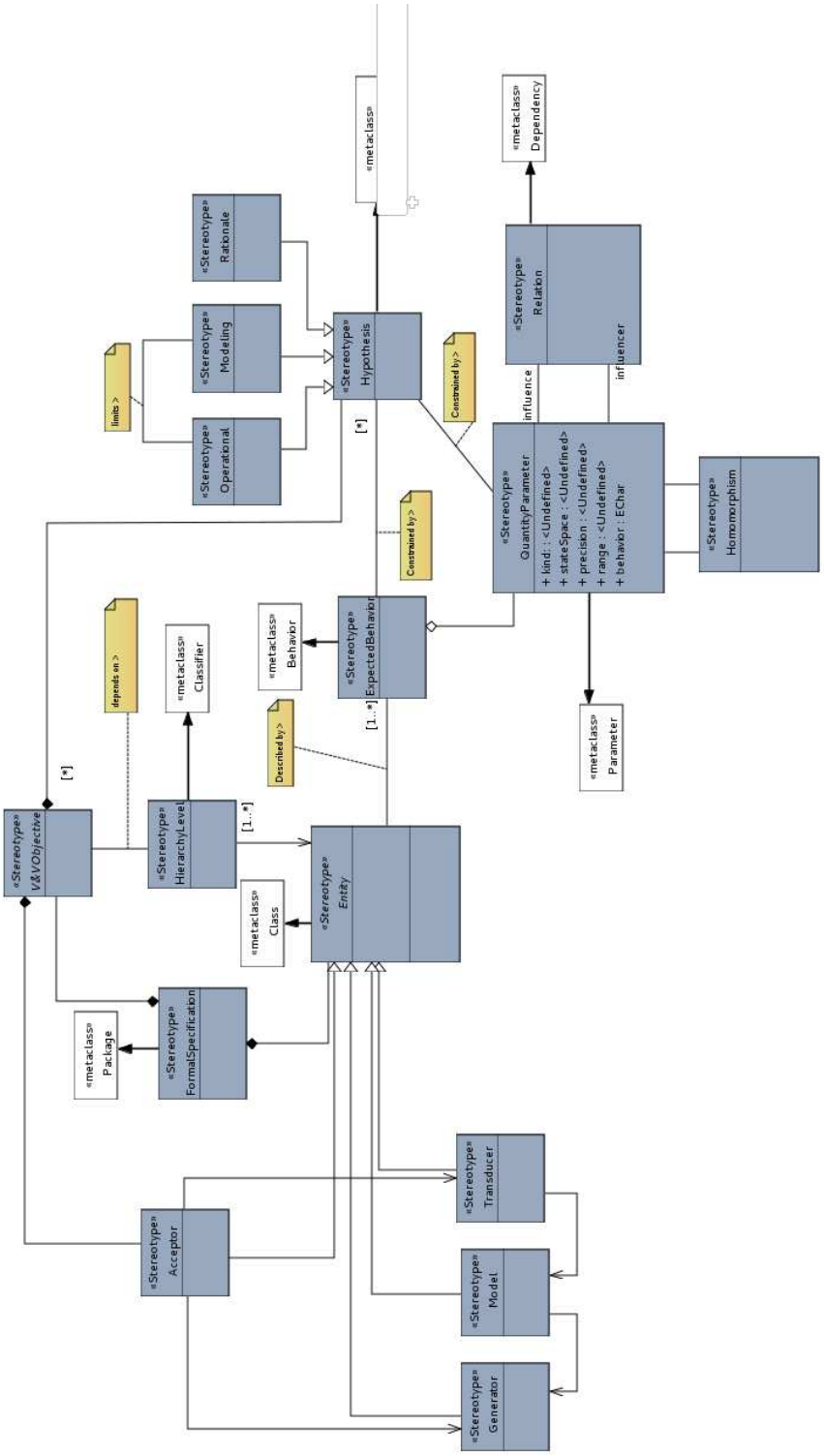

Figure 6. SiML profil