The design methodology for the verification of hybrid dynamical systems

Michal Pluska, David Sinclair

***Abstract*—This work presents the OHMS methodology. The main aim of it is to design a model of a complex system easy to process by formal model checking procedure. The outcome is a verification report showing safety of the system. As the novel approach the complex mathematical notation is hidden form the user and use object base approach with graphical notation. It gives the user better experience and more flexibility in the design. On the other hand at the end of the process the user is still provided with the formal verification report helping in the correct design.**

***Index Terms*—Hybrid system, formal verification, design methodology, systems engineering**

I. Introduction

HYBRID dynamical systems came to existence as a merge of the control engineering with an embedded software engineering. The control engineering deals with analog data describing work of the system where the embedded software can be seen as rules directing the system, described in the digital logic. Because of that, hybrid system must be characterised by both discrete and continuous system state changes.

II. State of the art

In the discipline of the design of the complex system the popularity is gaining the model design approach. It is based on developing a model of a real world system in the design. The next step is to perform a reasoning about such model. The most typical provides test of the system according to the prepared before specification. Unfortunately, testing of the system in practice is performed as checking only one working trajectory of the system at a given time. It is identical to the performing a simulation of the system. However, it can longer the whole development process depending on how each test cover the possible model spectrum. In general it is possible that testing whole spectrum of the complex system will use more project time than the design phase. Finally it is possible that not every working trajectory was considered because time or other project constrain and some critical spot can be missed.

The hybrid dynamical systems are often classify also as the complex systems. Moreover they can be seen as not straight forward to design. A lot of time and effort can be spend on any step in the process. In many cases the design of the hybrid dynamical system model is only a part of the overall solution. The following tasks are responsible for the optimization of the design and verification against requirements. In most of the complex and hybrid systems exist critical requirements connected to the safety specification of the system. Such description must be verified on each level of the system complexity. As a standard over here the V-model development is used [1]. It was proved in many designs over the time as helpful. However it needs support to trace changes between design levels and more important separation of the levels. In the summary mentioned above generic design process for the verification purpose use tests, despite limitation of this solution. On the other hand there are better approaches for the system verification than tests. On of them is use of the formal methods which gives mathematically proved full reasoning about the system in the design. Unfortunately it comes with the price for the designer to use. The are no established design processes that can utilize it. Moreover they they required extensive, specialized knowledge on the field and are not easy to be used [2]. Some of the existing general design methodologies [3] can use a formal methods in some part of the process. However, they do not directly relay on them.

Some of the above problems and approaches how to solve them were also mentioned in the literature [4]. It is a comprehensive description of different formal methodologies to solve industrial case problem, described as a controlling of a steam boiler. Published methods can be divided in to few categories. One are focus only on formal specification of the requirements and then checking its consistency. No further simulation or verification of the designed system is performed. The other tries to verify the completed design, lower number, however they do not address the issue of collecting requirements of the system. More over the verification is performed on the simplified system.

The above methodologies can be easily blended together. However, to avoid each other weakness need some support. The blend is understood by building the model of a complex system in a formal way supported by tools. In addition, by using the formal methods, it is possible to employ the model checking methodology instead of extensive testing of the system. However overhear the model checking methodology comes with its own price as well. It is based on the temporal logic which is a strict mathematical formalism. Unfortunately it is hard to be used in the daily design.

The other design aspect, which was not covered in the mentioned above book, is an architecture of the system. In most examples it is assumed that the architecture will reassemble the physical parts of the system. It is valid approaches but on the other hand it may not explore any other architecture more suited for the particular solution. The common problem, only briefly highlighted in the publication, is time needed for learning how to use each of the tool or methods, is described

Contact person: michal.pluska2@mail.dcu.ie; School of Computing, Dublin City University, Dublin 9, Ireland

This work was supported in part, by Science Foundation Ireland grant 03/CE2/I303_1 to Lero – the Irish Software Engineering Research Center (www.lero.ie)

as in months. Moreover the time used: for finding a solution of the problem and applying the solution has to be also considered. The design time for every solution was described as few weeks on the top of the time required for learning the methodology.

Non of the described solutions can address the whole design life cycle. In is required to use at least two different, described in literature, approaches to solve the industrial case. It can be easily assumed that the time used for the design will be doubled. In addition the time used for testing solution is not considered at all.

III. Methodology

The design methodology OHMS (Object-oriented Hybrid Systems Methodology) version 2.0, based on [5] tries to address issues presented in the above section of this work. It provides the high level, top down approaches to the design and verification of the hybrid system. Moreover address all the issues described above and aims to cuts the total design time. The total design time is understood as a time form capturing the first requirements to the verification report in form readable by the designer [6].

The methodology is tailored for a designer who deal with complex systems on a daily basis. This branch of science, more generally, is named system engineering. It combine different fields of engineering and focus on how complex project is designed and managed over its whole life cycle.

Hybrid dynamical systems are often described by more than one complex equation. Understand those equations and its derivative with relation to the system could be enormous task. From the different perspective the initial problem can be seen as to tackle whole system role and its initial complexity. It can be summaries as to find correct blocks building system and equations describing it. On the other hand it should be possible to hide as many complex equations as it is possible, just not to disturb the designer. Moreover the system designer need tools to capture description of the system and relations between its building blocks without relaying only on mathematical formalism.

The approach presented in this work, The OHMS methodology, relay on the tools supporting design of the complex system or hybrid dynamical system. The tools are integral part of the design process. Challenges for the designer during process of building any hybrid dynamical system can be summarised as follow:

- Identification of the objects building hybrid system and finding relations between those objects.
- Finding which of those objects are hybrid (gives this nature to the system) and should be modelled accordingly.
- Finding parameters describing the system and representing them in useful way in the model of the system.

IV. Methodology requirements

Described above challenges for the designer linked with the general design methodology guidelines [7], gives an opportunity to prepare some requirements for the methodology with aim in a verification of hybrid systems. Moreover by the design this process should help to build a model of the hybrid system [8].

1) To manage the scale of a real world application: Most of the problems in design of a complex system can be classified as not trivial - typical system can be build from large number of block. Moreover some of those object exchange information between each other. It is necessary to manage such design and prevent any interlocking or faults. On the other hand the partitioning of the complex system into the building block should be flexible to give possibility of exploring different architectures of the system.

2) Clear link between description of a system and its formal specification used for formal verification: Most of the design processes [3] is focused on developing a description of a system in form of its model. The further use of the system model is left for the practitioner without any clear guidelines for the formal verification. Most of the guidelines is focused on the implementation of the system in some kind of the use case based scenario [9]. Currently there is no clear link between system description and its formal specification. Moreover practitioner is force to try any formal verification on its own without any additional information.

3) Possible to be used without extensive knowledge of the formal methods: The formal methods can be useful in verification of complex system, however the design engineer may not have extensive knowledge about them to use them properly. The complex systems always require the knowledge from different engineering fields. In addition the formal methods are not typically included in typical engineering curriculum.

4) Clear definition of the methodology structure: The methodology to be successful must be understood by the practitioner at the first place. To be understood and used with efficiency the methodology should have clear structure [10]. The methodology is defined as a chain of steps to solve a problem. The clean structure defines each step with all necessary input and outputs. Moreover the defined guidelines can help in having a clean methodology life cycle [11].

5) Intermediate validation: This requirements is related to the one defined above. The methodology defined as a chain of steps should have possibility to give a practitioner reinsurance that it is used in the correct form. This possibility can be given by intermediate validation. It establishes a set of rules and place in the methodology chain where it can be justify if the design is on the correct track.

V. OHMS methodology

The design methodology is focused on describing the system requirements by examples of its usage in use cases diagrams. It allows verification of the gathered requirements and is a starting point of the analysis part. The use case diagrams, describing requirement's, helps to find main objects of the system. The objects of the system are transformed from actors appearing in the use cases. The analysis of those objects hierarchically decompose them according to the abstraction levels. During this action it is important to describe all data produced and needed by each object. This is done with respect to the abstraction levels boundaries. The data

transferred between objects is used for identification of a hybrid automaton and helps in its further design. When there is more than one hybrid automaton this data is used to connect and synchronize them. The final whole system verification is done with help of the hybrid automata's [12]. It gives also opportunity for parametric verification.

The methodology is described in steps, which must be performed in order. Each step can be seen as a milestone on the way to design and verify the system. It is a iterative cycle, where the outcome of each cycle is verification report. The report can be used for the refinement of the model. The main steps of the methodology, finishing with respective models of the system are presented as follows:

1) The requirements phase, which product is a requirements model
2) The analysis phase identifies components of the system and arrange them in hierarchy when possible. Components can be understood as a object. This phase produces the analytical model of the system.
3) The design phase identifies state variables of each object and define interface for each object behavior. It ends with design phase model of the system.
4) The detail design phase specifies the behavior as a hybrid automaton and define object dynamic evolution. It extends the previous one model of the system.
5) The verification and results phase. It is a final of the process cycle and gives a verification report used for refinement of the above models.

The detail design stage is a final stage of the methodology, related to the design of a hybrid system model. It is also an input to the model checking (verification) algorithm. The input consist of two complementary parts:

- The model description in a form of the hybrid automaton.
- Verification procedure code used to perform the verification.

The tool supports the design process by providing list of small steps to be performed. Each step is defined and described in a sufficient way. It is also connected to the other step and makes a logical chain. It is possible to be measured by visible rationality of the methodology. Moreover each following step is a logical extension of the previous one and adds more information to the model of the system.

The methodology guides user in the design process and prevent him/her from omitting any step and making any inconsistency to the model. This is highlighted as soon as possible, indicator of that is testability and is traced to the missing place in the methodology. Any extra information required after the test is inserted to the model without any problems. Moreover the tool where model is build gives possibility for the process extensions and tool itself. However the formal verification of the hybrid dynamical system relay on the information provided by the designer. The designer is also user of the verification tool and if it is possible the tool provides him with a support in the process. The formality of the verification procedure is not visible for the tool user as long as it is possible. On the other hand tool practitioner has possibility to build system described by those equations. From the user input tool tries to build the equations itself and keep the usage of the obscure mathematical formula's to the limit. Moreover if any of the formula is necessary for description of the system is can be entered in the whole. On the other hand, the user may prefer to enter many small partial equations which combination describe the system, more than one complex multivariate equation describing it. This approach is supported also.

The OHMS methodology is build around the tool-set based on a Eclipse IDE [13]. One of the main components used in this tool-set are graphical interface based on eclipse plug in named GEF it support graphical approach to the modelling language . It is combined with modelling framework, which provides generation of a code by rules described in XML. It generates two types of script:

- Verification procedure code in a form of a script
- Description of a system in object code.

The complexity of a code shows also importance of the methodology as a help to find all parameters describing hybrid system.

Eclipse is a software development environment comprising an IDE and a plug-in system to extend it. The Graphical Editing Framework (GEF) allows developers to create a rich graphical editor from an existing application model [14]. GEF employs an model view controller architecture. It gives opportunity that even simple changes applied to the model by the user are instantly visible. Other part of the tool-set is the Eclipse Modelling Framework. It is an Eclipse-based modelling framework and code generation facility used in building tools and other applications like this one supporting the OHMS methodology.

VI. Methodology Structure

Domain specific languages (DSL) have advantages compared to a general purpose language. They allow, express the problems and their solutions on an abstraction level that match the abstraction level to the problem domain. It also helps with productivity or re-usability if it is constructed correctly. The main difficulty can be the balance between domain specific language input, general language as a base and the maintenance of both. One of the main features of UML [15] is an ability to customize it and extend according to specific needs. It gives a possibility to extend UML terminology and adapts it for specifics domains. It allows extension of syntax for a new constructs which are not included in standard notation or changes this notation. Possible is also extension of a Meta Model to add or change semantics. Finally customizing UML hells in adding information during transforming one model in to another or for verification purpose [16]. As the other aspects creating custom variant of general language allows to use the existing modeling tools and methodologies. In addition it makes the modeling process easier for the practitioner, expert in the design domain not tool usage. Custom language is less abstract for the user. The type of customization depends on the domain where it will be used and how the model should be extended to utilize this domain. The approach for customization of UML is divided in to several groups based on

work that has to be done and difference from original standard [17].

The easiest extension of UML language can be seen as adding keywords, where keywords can be seen as reserved words attached to standard UML elements. The keywords are defined in a specification and put in a list of predefined keywords. This list can be also a step towards more radical extensions of the standard notation. This approach uses profiles, which are a limited extension of a reference meta model. It aims to adapt the meta model to a specific platform domain. The main extension construct is the stereotype, which is defined as a part of a profile. By that the stereotypes are used to add keywords, constraints, images or properties (tagged values) to model elements. However the profile idea does not allow for a specialization by inheritance of meta types from reference meta model. It is more focused on the adapting existing meta model with construct that are specific to a particular domain, each of this adaptation is written in a profile. Compared to that, the extension procedure, named middle weight aims to make meta models more specialized. It can be considered as a redefinition of a class extension mechanism used in the software engineering. It expands UML meta model and allows to add domain specific types. There is even more complex approach to extend UML notation. It is based on selecting, interesting for a domain, items and constructs and combining them into new language. The biggest advantage of this approach is possibility to customize behavior and make a very domain specific language. Only in basis rules relating to UML apply. The approach presented here as the last is used in the OHMS process.

Usually, as the start of the information gathering, is to define boundaries of the system in design. The hybrid system are specific cause its complexity and possibility to incorporate many artifact from many not directly related domains. On the other hand the verification of the system is defined by many strict rules and has to be formally defined to be performed correctly. It gives an opportunity to limit the scope of information about the system. Only the required are gathered. The goal of the verification is to show that the model of a system (and system itself) performs in the given range of the parameters. Normally they are the safe operations of the system. Hybrid systems can be seen also form the control engineering domain perspective as having the typical control loop. Based on that, it is possible to limit scope of the required information describing the system. By this approach the use case is focus on intended normal operation of the system and each of its components. The method, to limit the scope of the use case required for the system verification, overcome the biggest problem of the use case - how to combine different use cases of the system and what is real boundary of the system [18].

VII. Example

The OHMS methodology is presented by solving well know train-gate controller example, first described by the [19][20]. The system to be explored guards the railway crossing. The main part of the system, the controller, has to close the gate before the train crosses. The approaches proposed below show how the OHMS process transform the system requirements and use them as an input for the verification process. The actual computations of the safety parameters are performed in the model checker based on the PHAVer tool [21].

The methodology starts from gathering the requirements of the system. It is supported by the the domain specific language (DSL) named over here as a Actors DSL. The name is connected with the focus of the language (see figure 1). By it usage the designer have opportunity to express requirements of the system by use cases.

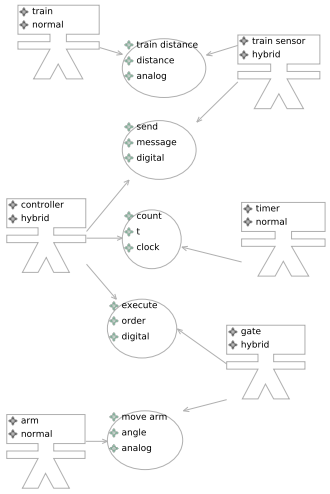


Figure 1: Actor DSL view

Object DSL is used as the support for the analysis step of the design methodology. Initial description of the project in this language is based on the automatic translation from Actors DSL. The information is preserved from the earlier stage of the methodology and transformed in to the new structure. Emerged structure must be filled in with more information by the designer (see figure 2). The embedding of hybrid automaton as a part of the other automaton is not considered in this process. It brings unnecessary problems for the verification tool. During verification each automaton must be explicitly design, not embedded in the other. However on earlier stages of the process it can be hidden to easier the process. There is no direct modelling of a feed back from the environment to the controller. This may lead to a deadlock of the simulation model. However the change of data flow is obtained from the environment of the system. In most cases hybrid systems are used in the place of control systems. The basics partitioning of the system in design can be related to partitioning of the system from classical control theory.

The flow of an information or a data between artifacts can be described in consistent scenario, with some of the artifacts as a source of the data. This can be compared to the data flow based design of the system. It force the designer to build a linear scenario with no branches in it. Any variation in the scenario leads to other one. Each scenario have own different prerequisites, apart from error handling. In relation to the control engineering it can be seen as there is only one main control loop. All possible errors has to be taken in to account

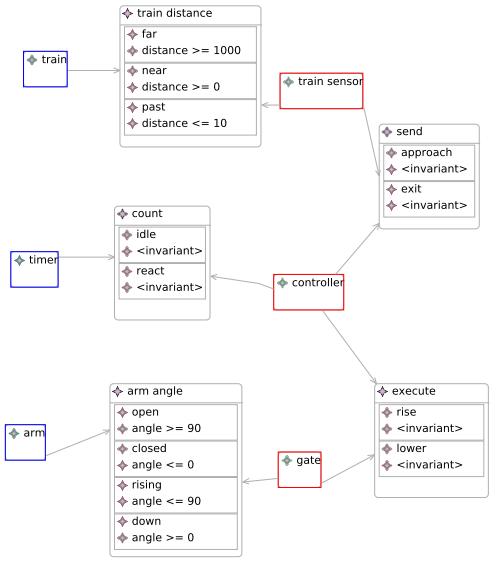


Figure 2: Object DSL view

during design, by preparation of error handling in later step of the process. Possibility of such branches is highlighted by the verification report. It refers to the overall aim of the OHMS process - the design of hybrid system for a verification.

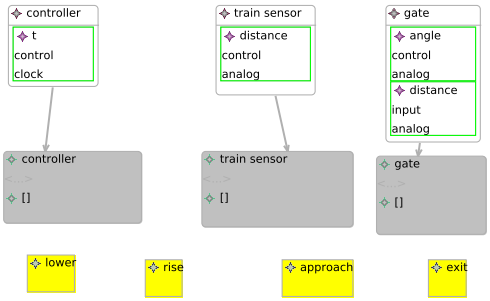
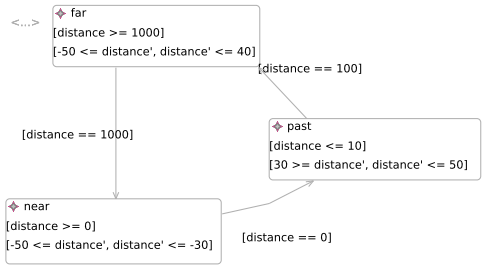
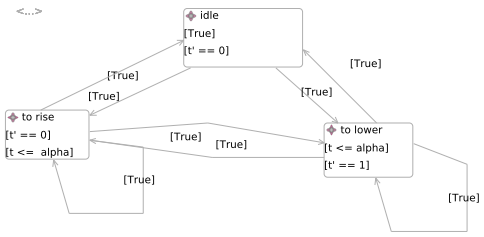


Figure 3: States DSL view

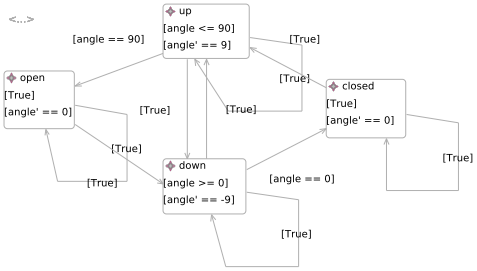
The States DSL is the last of the methodology milestones directly related to a system model creation. In this step final behaviour of each necessary component is specified. Over here the detailed specification of the system artifacts is based on the input from previous methodology steps. This DSL must capture the whole of the system behaviour and still need to manage the system structure, already described. The form and structure of the behavioural part must describe the system. However, it is also linked to the model checker and must be transformed into form understandable for the model checker. This part of the process finally structures all information gathered in the previous steps. The designer has an opportunity to finally fully specify the system model in a form of hybrid automata's. Then use this specification for a formal verification. The structure of this DSL was divided into two complementary parts described as follows. A system states DSL gives high level view of the system and highlights each of the automaton (see figure 3). In addition it specify the finally variables describing whole system and its scope. The lower level description of the system is linked to each of the automaton. The Automaton DSL describe each of the automaton (see figure 4) artifact from the system states DSL.



(a) Train sensor automaton



(b) Controller Automaton



(c) Gate automaton

Figure 4: Automaton DSL view

More over on the there are two possible view on the data building the system model. The one described above is structured view on the automata's building the system. The other one is focused on the cooperation between those automata's and is named sequence DSL (see figure 5).

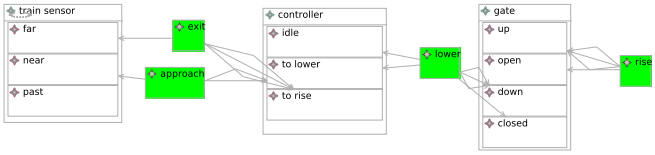


Figure 5: Sequence DSL view

The results of the formal verification can be presented to the engineer in form of chart which is more easy to evaluate. In this example the charts show the correlation of the train distance form the sensor with the angle of the crossing gate arm. The 90 on the Y axis mean gate full open and 0 gate close respectively. The X axis shows the distance of the train from the crossing (see figure 6). The 10 meter mark (red line)

show the zone where the gate should be fully closed. Other situation brakes the safety requirements for this example.

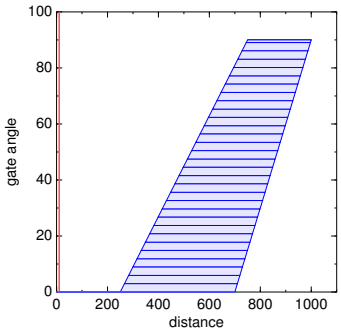


Figure 6: Verification interpretation

VIII. Results

The methodology provides guidelines for high level, top down approaches to the design and verification of hybrid dynamical systems. It aims to cut the total design time and more important provide the formal verification of a design. Other aspect is to provide this abilities in the less demanding, for the user, structure.

Proposed above OHMS design methodology OHMS address those issues. It provides the high level, top down approaches to the design and verification of the hybrid system. Moreover address all the issues described above and aims to cuts the total design time.

The preliminary runs with the potential practitioners shows that there is a huge demand for detail explanation of the use case approach. The focus had to be stressed on the extensive explanation of view of the system, important for this design process. It was noticed that this step is crucial in the process. In addition the following steps of in the process relay on its correctness. This fact leads to the additional regiments for the OHMS methodology, its description and formalisation.

Non of the participants were familiar with the exact definition of the hybrid dynamical system. On the other hand each of them had some knowledge about the dynamical systems, understood as a system evolving in the time. Moreover participants were aware of the computer based possibility to control such system. This computer based control was generally understood as some kind of electronically discrete components being integral part of the system. One part of the participants group, particularly those with some knowledge of electronics engineering were aware of problems which can be summarised in the term of system digitisation work. This practitioner feedback was used for further tuning the OHMS methodology.

References

[1] I. Sommerville, *Software engineering. International computer science series*. Addison-Wesley, 1997.

[2] J. Woodcock, P. Larsen, J. Bicarregui, and J. Fitzgerald, "Formal methods: Practice and experience," *ACM Computing Surveys (CSUR)*, vol. 41, no. 4, pp. 1–36, 2009.

[3] J. Estefan. (2007) Survey of model-based systems engineering (MBSE) methodologies.

[4] J.-R. A. E. B. H. Langmaack, *Formal methods for industrial applications: specifying and programming the steam boiler control*, . Lecture notes in computer science, Ed. Berlin : Springer, 1996.

[5] D. Sinclair, "Using an object-oriented methodology to bring a hybrid system from initial concept to formal definition," in *Hybrid and real-time systems : international workshop, HART '97, Grenoble, France, March 26-28, 1997 : proceedings*, 1997.

[6] T. Austin, "Why have a systems engineering (se) capability for automotive product development?" *SAE TECHNICAL PAPER SERIES 2007-01-0782*, 2007.

[7] A. T. P. Botta, R. Bahill, "Fundamental Principles of Good System Design," in *Engineering Management Journal*. Citeseer, 2008.

[8] I. Ogren, "On principles for model-based systems engineering," *Systems engineering*, vol. 3, no. 1, pp. 38–49, 2000.

[9] G. Trombetti, A. Gokhale, and D. Schmidt, "A Model-driven Development Environment for Composing and Validating Distributed Real-time and Embedded Systems: A Case Study," in *in Model Driven Software Development-Volume II of Research and Practice in Software Engineering*. Citeseer, 2005.

[10] D. L. Parnas, "Information distribution aspects of design methodology," 1971.

[11] S. Kleinfeldt, M. Guiney, J. Miller, and M. Barnes, "Design methodology management," *Proceedings of the IEEE*, vol. 82, no. 2, pp. 231–250, 1994.

[12] T. Henzinger, "The theory of hybrid automata," *IEEE Symposium on Logic in Computer Science*, 1996.

[13] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *EMF: Eclipse Modeling Framework*. Addison-Wesley Professional, 2008.

[14] R. Gronback, *Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit*. Addison-Wesley Professional, 2009.

[15] T. Pender, E. McSheffrey, L. Varveris, and I. Books24x7, *UML bible*. Wiley, 2003.

[16] K. Berkenk
"otter, S. Bisanz, U. Hannemann, and J. Peleska, "The hybriduml profile for uml 2.0," *International Journal on Software Tools for Technology Transfer (STTT)*, vol. 8, no. 2, pp. 167–176, 2006.

[17] T. B. Jörg Petersen, "A uml meta model for specifying functional requirements of mechatronic components in vehicles," in *OMER 2 (Object-Oriented Modeling of Embedded Real-Time Systems)*, 2001.

[18] K. Bittner and I. Spence, *Use case modeling*. Addison-Wesley Professional, 2003.

[19] J. Bengtsson, K. Larsen, F. Larsson, P. Pettersson, and W. Yi, "Uppaal - a tool suite for automatic verification of real-time systems," 1995.

[20] T. A. Henzinger, P. hsin Ho, and H. Wong-toi, "A user guide to hytech," 1996.

[21] G. Frehse, "Phaver: Algorithmic verification of hybrid systems past hytech." Springer, 2005, pp. 258–273.