

ULRR

The design methodology for hybrid system verification.

Item Type	Meetings and Proceedings
Authors	Pluska, Michal;Sinclair, David
Citation	9th IEEE International Conference on Cybernetics Intelligent Systems;09/2010
Publisher	IEEE Computer Society
Download date	2026-03-11 07:46:20
Item License	https://creativecommons.org/licenses/by-nc-sa/1.0/
Link to Item	https://hdl.handle.net/10344/1794

The design methodology for hybrid system verification

Michal Pluska
Lero at DCU, School of Computing,
Dublin City University,
Dublin 9, Ireland
michal.pluska2@mail.dcu.ie

Dr David Sinclair
Lero at DCU, School of Computing,
Dublin City University,
Dublin 9, Ireland

Abstract: Hybrid systems are gaining interest in control engineering because the ability to provide information for verification of the system by using hybrid automata. Moreover the importance of the model based approach is growing in the software engineering. In research related to the hybrid systems any structured design methodology did not get enough attention up to now.

Keywords: *hybrid automata, design methodology, system verification, complex system, control software*

1. INTRODUCTION

Hybrid systems are built as a combination of computational systems and real-world physical parts. The computational parts can be seen as one or more embedded computers connected by a network and interacting with the physical world via sensors and actuators. The control engineering techniques helps to coordinate this set-up. The computational part of the system is therefore affected by the real-world physical part. In addition it has to deal with the real-time properties of the physical world. From the other perspective, hybrid systems can be seen as a group of cooperating devices where the computing device is only one and responsible for coordinating the whole system and ensuring it works as intended.

The complexity of hybrid systems makes its design a very challenging issue. Obtaining the correct design can be problematic, and extensive testing of various prototypes may be necessary. Unfortunately this can raise the design costs and the time required to finish it. The other problem to tackle during the design is correctness of the system. This problem is addressed with a design methodology using formal methods for verification of the system in the design [1].

Most widespread approaches to hybrid system design and later verification reassemble the bottom-up design methodology. The work begins with information about the system gathered in the form of physical rules and equations. Objects in the design are related to physical objects of the system and the interactions between them are made according to the physical equations. Those interactions can also be described as an agent-based approach to the description of a system. This bottom-up methodology is focused on detailed description and its relevance to the physical world. Describing the system from the beginning of the design may lead to too

many detailed equations in the initial stage. This is clearly visible with large systems described by many physical laws. Currently abstraction of those systems, useful during design, is possible only by using less realistic physical equations. Moreover there is no established rule for keeping track of the changes. Every case is explored individually with different levels of abstraction, and its solution is found by numerous experiments. The number of considered details can be problematic for the design engineer and can easily be a cause of error. In this approach any verification is limited to the component level. It comes with the assumption that system build from the verified components will be verified as well. It may not be the case and complex systems usual tend to be simulated during design.

A. System simulation

System design by this approach is verified by numerous simulations. The aim of simulation is to avoid extensive testing after the manufacture and hence reduce the cost and time. Unfortunately the degree of confidence in the correctness of the simulated design may be limited. The main cause of this is that too large a set of input data will cause unpredicted interactions with the environment that are impossible to check. Building a prototype of the system suffers from identical problems as the system complexity rises.

One of the biggest assets of hybrid systems related to its possibility of combining control engineering and software engineering is verification of the system and its properties.

B. System verification

In comparison to the simulation, formally verifying high-level designs of complex systems can be very useful for hybrid systems, many of which are safety critical. By building a formal mathematical model of the system it is possible to use automated model-checking methods to prove that all requirements are met or all possible system input sequences are checked [2]. Simulations only allow some of the potential system inputs to be checked.

On the other hand, the usefulness of formal methods is limited by the lack of a well-defined methodology that would make it broadly applicable. There are no tools for interactive model building and analysis interpretation. Moreover the complexity of the system in design can be overcome by using appropriate abstraction of model's details what is currently not supported at all. There is also a need for aids to translate informal requirements specifications into formal specifications. A final aspect worth mentioning is that formal methods specification and verification might be problematic for practical engineers, because their focus during design is on different aspects.

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

C. Hybrid automata

A hybrid automaton is an abstraction of a finite state machine, which allows continuous variables. The discrete actions are modelled by moving through a finite set of control locations. In addition the continuous actions are modelled by real variables which values change continuously over the time according to the physical equations describing them. Those equations belong to the ordinary differential equations (ODE) type [3].

The model checking algorithm of hybrid automata analyses the defined properties to determine if they are violated in any state reachable by the automata [2]. In other words it computes the set of states of the linear hybrid automata which are reachable from a set of initial states by iteratively performing time and transition steps. In some cases backward reach ability analysis is possible. As the other advantage the hybrid automata can be designed in parallel configuration, which is useful for describing large, complex systems. Each part of the system can be described by separate hybrid automation with the possibility to communicate between automata. The model checking of the hybrid automata is restricted to the linear case for effective automatic algorithmic analysis. Variables describing flow of the system are independent from current state of automata. All non linear hybrid automation must be approximated to the linear version, by any suitable technique [4].

DESIGN METHODOLOGY

The proposed modelling and verification of the hybrid system is divided into stages (see Figure 1). This partitioning allows the correct complex system to be built from the start and avoids problems which otherwise would only be found after the initial design is complete. Our design methodology is focused on describing the system requirements by examples of its usage in use cases. It allows verification of the gathered requirements and represents a starting point for the analysis. The analysis of those objects will hierarchically decompose them according to the abstraction levels that build the system model. This is done with respect to the abstraction level boundaries. This stage of the methodology design system model has all the features described in the requirements and is a backbone of the model used in the system verification process. All the necessary information is gathered during the design of the system model. The final verification of the system and the possible parameters of the system are done by a hybrid automaton. This approach builds and verifies the system model independently of the hardware. This makes it possible to use control software already verified on a different hardware platform, depending on the needs of engineers. As seen on figure 1 the proposed design methodology is divided into steps. Each of them is focused on different aspect of the a system design. This division can be also seen as layers, where each of them bring additional info into the design. The design steps will be explained in the example described later in this

work. The final step gives a hybrid automata model. This model will be used for verification and may change depending of its outcome.

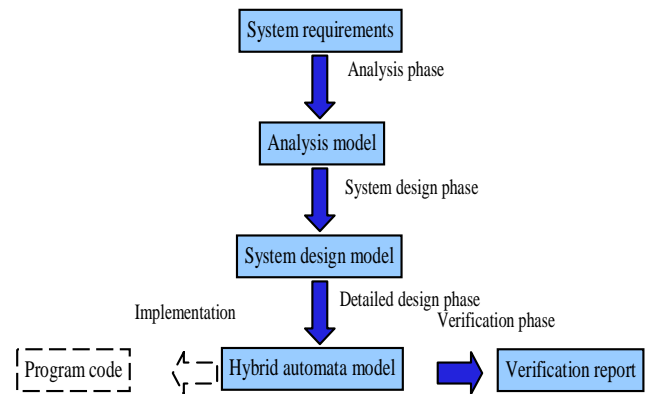


Figure 1. Design methodology diagram

DESIGN METHODOLOGY IN EXAMPLE

In the modern vehicle amount of cooperating mechanical and electronic components is increasing each year. This complex system is often used as an example of the hybrid system. Some of the existing design already has around 60 computing units (Volkswagen Phaeton). Among many factors driving this rise are the increasing importance of controlling complex vehicle systems to meet requirements of safety (e.g. air bags), fuel economy, environmental requirements (e.g. CO2 emission levels), comfort and convenience (e.g. air condition), multimedia or entertainment services. Automotive system is controlled by so-called Electronic Control Units (ECU). An ECU consists of a microcontroller and memory, next to power electronics to drive sensors and actuators. The software in side of the ECU implements control algorithms combines the sensor values and calculates some meaningful actuator signals. In a study between automotive software developers it was found that in recent years cost of software development rose and exceeded one third of total vehicle production cost [6][7]. Moreover a trend can be seen where more functionality will be put in software to allow reduction of hardware sensors cost [8]. This trend stresses the importance of emerging hybrid systems and also increases the need to build them in correct way.

The proposed design methodology for hybrid system verification will be introduced by describing its possible usage. The example described in this work will be a development of the Electronically Controlled Suspension (ECS) for a passenger vehicle [9]. The nature of the active suspension system is hybrid because its need of adaptation controls strategy to constantly changing environment. The control input depends on discrete state where the system is at particular moment in time. This influences the continuous

state of the system, which, in turn, determines the transition between discrete states [10].

The suspension of the vehicle has developed over years to a high level of complexity and sophistication. In the past car makers used metal spring elements but after years have switched to use hydro-pneumatic or pneumatic elements to isolate the vehicle for road irregularities [11]. The modern suspension system combines mechanical and electrical system controlled by sophisticated algorithm.

The roles in which those systems work can be transformed into a list of requirements for the system. Starting from information about vehicle dynamics and leading to detailed description of the ECS task.

Gathering of the detailed requirements can be done by developing use case studies. Use cases can help see requirements from different perspectives. All of them can be linked in one, like on figure 2, where the actor named vehicle wheel dynamics represents the behaviour of the wheel and by that is a part of the suspension system. The actor named vehicle wheel pneumatics represents the active parts of the vehicle pneumatics system. The actions which they can take and, related to the suspension, are change the vehicle level and set the vehicle level respectively.

The basic goal of the system is to maintain the vehicle chassis of the same level. The change in vehicle level caused by vehicle dynamics actor should trigger the action of the actor, vehicle pneumatics. On this, high abstraction level, the value of variable describing the difference between changed vehicle level and desired vehicle level is not calculated. There are two possibilities the vehicle level set up by the actor vehicle dynamics can be higher or lower compare to the desired vehicle level. The actor named vehicle pneumatic system should behaviour respectively to that; it has to minimize that difference. There is a need for constant recalculation of the difference between desire and actual vehicle level.

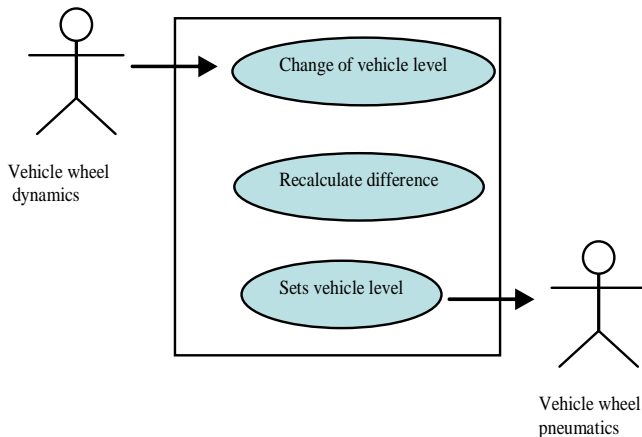


Figure 2 Vehicle active suspension use case diagram

This approach describes behaviour of active suspension system and basic services that ECS delivers, which could be routed from the source of information to its sink as, the

possibility of observing and recording the change of vehicle level cause by the different behaviours of the vehicle on the road, see figure 3. This recorder change should be used for recalculation of a difference between given and desired vehicle level. The difference should be used as set up information for actuators responsible for vehicle level.

Each part of the pneumatic system has its own behaviour related to time or in different words reaction time in addition the different physical properties describe each subsystem. This has to be reflected in the design and taken into the account during verification.

In some cases reaction of the vehicle pneumatic may not be fast enough, compared to the changes of vehicle level made by vehicle dynamics. This situation has to be highlighted to the designer by tool. It is a problem of not functional requirements like reaction time of a whole active suspension system on a change coming from vehicle dynamics. On the other hand it can be one of the verification parameters as well.

In this example the speed of the vehicle will have direct influence on the reaction time of the suspension system. On the other hand some of the road irregularities may be so small and shouldn't have any influence on the suspension system. It also can influence reaction time of the system; moreover it defines resolution of the system. This not functional requirements highlight the problem of trade-offs in the system and stress need of parameters and possibility of version tracking. This example shows importance of tracking requirements in the design process. It is done by a list of requirements and each of them should have at least one solution or task assigned to them as an answer to the requirement. Moreover the verification process must allow verification of multiple parameters describing the system

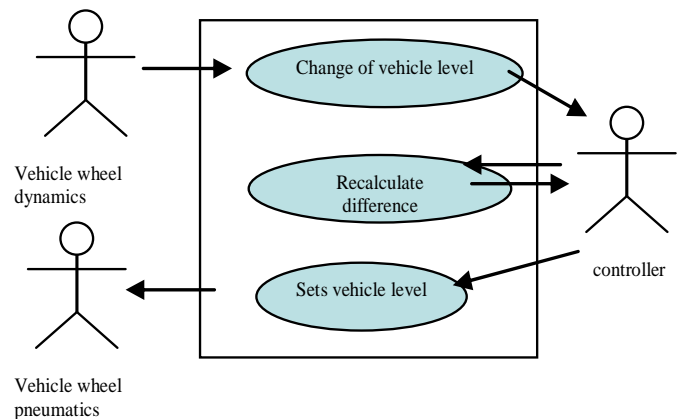


Figure 3 Active suspension controller use case diagram

The flow of information or data between actors can be described in consistent scenario, where vehicle wheel dynamics actor is a main source of data. This linear scenario is a scenario with no branches in it. The only exception will be error handling not shown in this work. Any variation in the scenario should lead to other scenario having different prerequisites. In relation to the control engineering it can be

seen as there is only one main control loop and all possible errors should be taken in to account during design, by preparation of error handling on later, more detailed, stage of design. Possibility of such branches is highlighted by the verification.

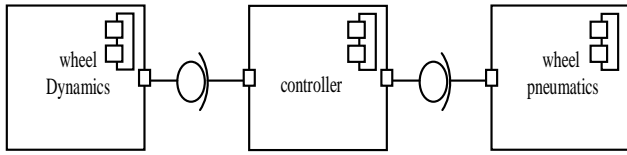


Figure 4 Basic suspension partitioning

During analysis of use case graphs those actors can be described as a data stores similar to the data flow diagrams. Each of graphic symbols has its representation in formal logic language to allow transformation from use case like diagram, see figure 3, to the other one described in the figure 4. The actors are basis for the finding and definition of main system components, later related as a main system objects, like on figure 4. Each of those components or objects can be described by hybrid automaton allowing verification.

The starting point of analysis of vehicle pneumatic system is shown on the figure 3 where compressor actor or release valve actor interacts with pneumatic actuator. The transfer of this description to the data store can be seen in the figure 4. Each of those elements has a possibility of embedding a subsystem describing its behaviour in more detail, see figure 5. Moreover each of them has to track information needed for the hybrid automata later in the design process During verification each automata must be explicitly design not embedded in the other one. However on earlier stages of the process it can be hidden for easier design. There is no direct feed back to controller on the work of those elements; it may lead to a deadlock of the simulation model. It is done by environment of the system in design.

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 [12].

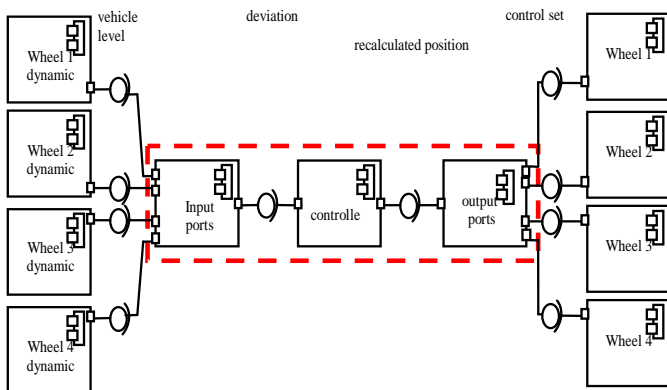


Figure 5 Controller hierarchy

This example and most of the hybrid systems considered for the design can be defined as a system which constantly tries to

adjust itself. ECS tries to adjust position of vehicle chassis with disturbance (errors) coming from the road. The classical control system with feedback consists of a controller performing adjustment calculations, sensors observing the disturbance in the system's environment and the environment itself. This is a control round or loop which will be use in this methodology for more detail design. The basic elements of this control loop are taken from partitioning the system in to the objects, figure 5. It is done to allow express information transferred between objects in time, see figure 6.

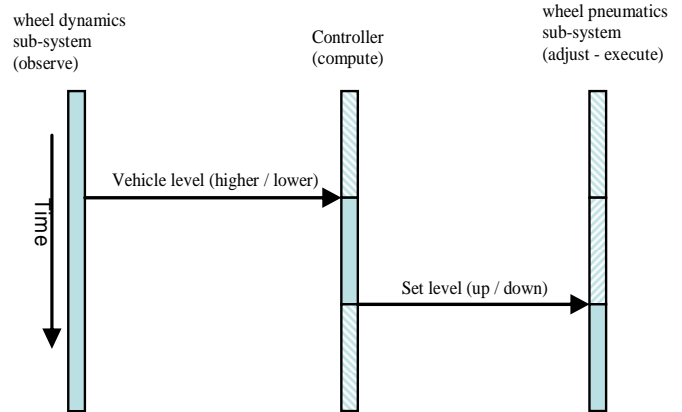


Figure 6 Basic controls round

The control information between objects can be referred to the role of the system, the wheel dynamics object provide information about current vehicle level. It should be done it value possible to be used to set up new vehicle level. The absolute value of the vehicle in this case is not important for the system. Only the change of the level or its deviation is important. In this case the verification must ensure that the change of the level will be always inside design boundaries. Another aspect is the delay in time after which vehicle systems will react on changes of the vehicle level. It brings the problem of filtering some of the vehicle level changes. Changes might be too small; the resolution of sensor or too fast, the system will not be able to react.

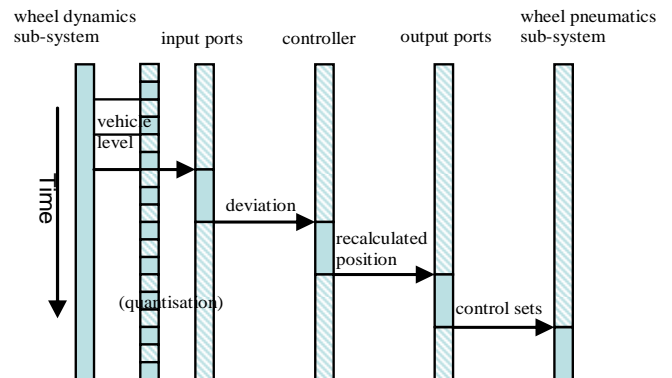


Figure 7 Control data flow

This analysis phase ends with overall structure of the system as a group of defined object and links between them, see figure 7. As can be seen on figure 5 objects are linked by shared interfaces which are used to pass state of variables between objects as information about state of the system. In different designs this may be used to coordinate different hybrid automata used for the verification. The model of the system should not only consist a software controller but also a model of it environment. It is necessary for the verification of the system and finding system parameters. Information exchanged between objects, found during analysing the system, is labelled like on figure 7. This information and labels are used to describe hybrid automata edges, see figure 8.

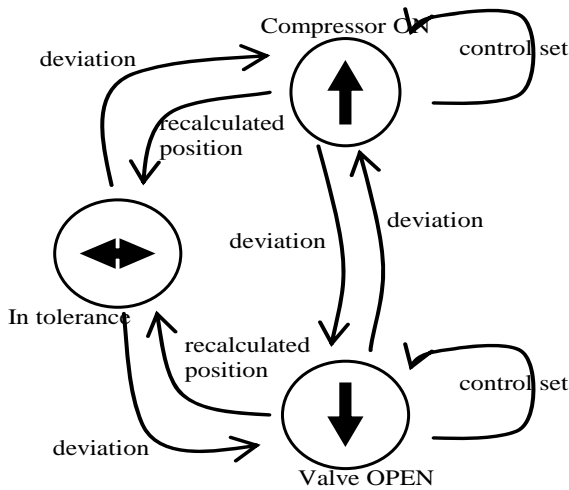


Figure 8 Controller

This description is a starting point of transforming controller to the parameterized one, see figure 9. Parameter analysis of the hybrid system gives a possibility for fining optimal configuration of variables describing the system and by that better performance of the system. However final description of the parameters must be specified in requirements. Results are also validated if they pass safety requirements of the system. This design is final design of controller automata, from which the code for the verification tool will be generated.

The figure 9 shows ECS controller with parameters where y is a current vehicle level and its desirable value is zero. Any disturbance of this value is measured against parameters, limits in which system can change itself.

All of this data gathered during design of the hybrid system model are used for the verification purpose. As a verification tool in this methodology the HyTech [3] is propose. It is the most complete tool based on hybrid automata and as the only one allows a parametric approach [4]. Moreover HyTech is better suited to high level system description, where the continuous variables either simple dynamics or it is possible to transfer them for the one with simple dynamics. This would be an advantage in this top down design approach. Depending on the verification aim it is possible to show if the design has any

deadlock or what are the limits of parameters describing the system. Decomposition of the hybrid system in layers having adding more details to the design ensure that the engineer will cover all requirements and the system will meet the specification.

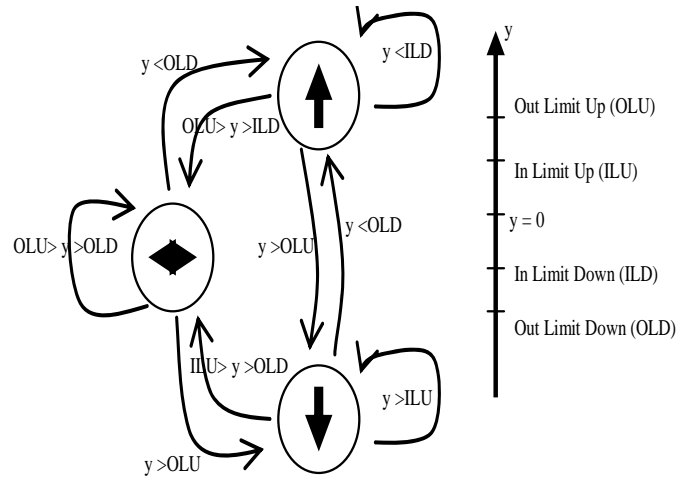


Figure 9 Controller with parameters (y is a vehicle level in ECS)

EVALUATION

As was mentioned before the methodology is supported by the tools currently in the development. The graphical language used in methodology can be seen as a domain specific language build on the top of UML. As an example the UML and its extensions are also used in the AUTOSAR project to model various parameters of automotive systems [13].

The other project related to the design of automotive hybrid systems in formal way is AutoMoDE [14], where the code generation ASCET tools were combined with the research outcome of the AutoFocus [15] project. Proposed over there approach is based on the HyCharts [16], which tries to extend state charts for the continuous domain. This problem was overcome by partitioning the system to the continuous and discrete space in the earliest stage of the development. It is not different from other existing approaches, like mentioned before Matlab. Moreover research around the Matlab tries to extend its usability by verification with use of formal methods [17]. Unfortunately, all that approaches relay on early partitioning in to discrete and continuous domain. This may be still appropriate for detailed design of the specific part of the system.

However as it can be seen from broader perspective currently hybrid system design methodologies are focused on the separate design of each of the system objects. It can be described as a bottom up approach. This design is focused on correctness or finding significant parameters of each object on its own. Moreover approach is used also for verification of the system, where is assumed that each formally correct object is

connected to other verified objects and by that whole system is verified. That may not be a case for the complex system.

The new and different approach, proposed in this work, is based on a top down design inherited from automotive system design approach. It is focused on a whole system and its role. It would tackle the complexity on a higher level with the possibility to consider more than one structure of designed system. Moreover it would allow verifying the correctness of whole system.

The problem of composition of each verified object in the system is explored in many works [1], but it also may lead to a problem where an object verified with a set of parameters for one task, even an obvious one, may not be the best choice for other task. In high level description of a system parameters or symbolic constants are often used with not specified real values. In most cases those parameters would gain values later in the design process during implementation. The parametric analysis of the system would determine necessary and sufficient constraints for parameters of the system to operate safely. Computing limitations in which system will not violate safety requirements would help define the optimal set of parameters for the system work. The case study used as an example was also described by different approaches [10]. However over there the focus was on the verification itself without any information how to handle the complexity of the system in design. The other assumption over there was that the designer is familiar with formal methods as a verification tool. This may not be a true for every system engineer. Proposed in this work design methodology allows the system engineers to have benefits of formal methods by using more understandable and valid for them approach.

SUMMARY

This work proposes design methodology focused on describing the system requirements by examples of the system tasks in the use cases diagrams. This approach allows verification of the gathered requirements. It is also a starting point for the analysis of the system needs. The use case diagrams helps to find main objects of the system. Those objects can be seen as evolved from the actors in the gathered use case diagrams.

The analysis of those objects decomposes them hierarchically according to the abstraction levels in the similar way to the abstraction layers existing in the EAST-AML [19] approach. During this action it becomes important to describe all data produced and needed by each object. This is done with respect to the abstraction levels.

Those data transferred between objects are used to identify a hybrid automaton and are helpful for its further design. In case when more than one automaton is needed to describe system behaviour those data would be used for connecting and synchronizing those automata.

The final verification of the system and possible parameters of the system is done by hybrid automaton. The information needed for the verification is automatically gathered during the design of the system.

By verifying the model of a system with the model checker the system designer has a formally proven model system which can be use for the actual implementation in the ECU form the example used above. By that the only thing to be verified is actual implementation and it correctness against the model.

References:

- [1] D. Sinclair; Using an Object oriented methodology to bring a hybrid system form initial concept to formal definition; Lecture notes in computer science; issue 1201; Springer Verlag; 1997
- [2] R. Alur, T. Hezinger P. Ho; Automatic Symbolic Verification of Embedded Systems; IEEE Transactions on Software Engineering; vol. 22; 1996
- [3] T. A. Henzinger, The Theory of Hybrid Automata, Proc. Of the 11th Annual IEEE Symposium on Logic in Computer Science, p. 278-292, 1996
- [4] T. A. Henzinger, P. H. Ho, H. Wong-Toi, A User Guide to HyTech, Tools and algorithms for Construction and Analysis of Systems, p. 41-71, 1995
- [5] T. Pender; UML Bible; John Wiley & Sons; 2003
- [6] A. Pretchner, M. Broy, I.H. Krueger, T. Stauner; Software engineering for automotive systems: a roadmap; Future of Software Engineering; 2007
- [7] M. Broy, I.H. Krueger, A. Pretschner, C. Salzmann; Engineering automotive software; Proceedings of the IEEE; 2007;
- [8] K. Haenninen, J. Maeki-Turja, M. Nolin; Present and future requirements in developing industrial embedded real time systems – interviews with designers in the vehicle domain; Engineering of computer based systems; IEEE; 2006
- [9] R.K. Mehra, J.N. Amin, K.J. Hedrick, C. Osorin, S. Gopalasamy; Active Suspension Using Preview Information and Podel Predictive Control; IEEE International Conference on Control Applications; 1997
- [10] N. Elia, B. Brandin; Verification of an Automotive Active Leveller; Proceedings of American Control Conference; IEEE; 1999
- [11] J. Darling, R.E. Dorey, T.J. Ross-Marlin; A low cost active anti-roll suspension for passenger cars; Journal of Dynamics Systems, Measurement and Control; 1992
- [12] R. Dorf, R. Bishop; Modern Control Systems; Pearson; 2008
- [13] B. Tavernier, Calife: A Generic Graphical User Interface for Automata Tools, Electronic Notes in Theoretical Computer Science, vol. 110, p. 169-172, 2004
- [14] P. Cuenot, P. Frey, R. Johansson, H. Loenn, M.-O. Reiser, D. Servat, R. Tavakoli Kolagari, D.J. Chen; Developing automotive products using the EAST-ADL2, an AUTOSAR compliant architecture description language; Proc. of ERTS 2008, Toulouse, France.
- [15] A. Bauer et al; AutoMoDe – Notations, Methods, and Tools for Model Based Development of Automotive Software; SAE 2005
- [16] F. Huber, B. Schaeetz, G. Einert; Consistent Graphical Specification of Distributed Systems; Lecture notes in computer science; issue 1313; Springer Verlag; 1997
- [17] R. Grosu, T. Stauner; Modular and Visual Specification of Hybrid Systems: An Introduction to HyCharts; Formal methods in system design; Vol. 21; Springer; 2002
- [18] A. Cavalcanti, P. Clayton; Verification of Control Systems using Circus; Proceeding of IEEE International Conference on Engineering of Complex Computer Systems; 2006
- [19] M. Rapp, P. Braun, M. von der Beeck, C. Schroder; Automotive Software Development: A Model Based Approach; SAE-2002-01-0875; 2002