AUTOSAR Seminar WS2008/2009 - Assignment: Simulation of Automotive Systems in the Context of AUTOSAR

Krasnogolowy, Alexander
March 31, 2009

Hasso-Plattner-Institut for IT-Systems Engineering
University of Potsdam
Prof.-Dr.-Helmert-Straße 2-3
14482 Potsdam
Department "Systems Analysis and Modeling"
Prof. Dr. Holger Giese

Abstract. In the automobile industry, the usage of embedded systems becomes more and more important. Today, many innovations are realized by software solutions[15]. The increasing complexity on the one hand and the high performance requirements on the other hand, leads to a new challenge in assuring the quality of this systems.
This paper takes this problem into account and shows how the quality objectives of automotive systems can be assured with the help of simulation techniques in every development stage. For this purpose, the development stages are introduced and it is shown which kind of simulation can be used in which stage.
Furthermore, the usage of AUTOSAR leads to new possibilities in simulating automotive systems. Through the hardware independent software architecture, applications can be already simulated in an early development stage. Tools supporting such a simulation are presented and discussed.

1 Introduction

Simulation is the transformation of a system to a model and conducting experiments with that model [1] to analyze the dynamic behavior [2]. The model of the system can be a model of an an existing system (descriptive model) or a system, which is has to be build (prescriptive model). If the system is simple, it can be transformed for example to a mathematical model and the given problem can be solved analytically [1].

However, real world system are usually very complex and analytical analyses are difficult, because the software consists of many lines of code and is interacting with other components[3]. Embedded system are also influenced by the scheduler of the underlying operating system and also interrupts, which can occur randomly. Moreover, an application has many different execution paths and states, which have to be investigated. So, because of the high complexity and the different combinations in the control flow of the embedded system a simulation is executed. In a simulation only the most interesting cases are considered and for every simulation step one possible execution path is gone to observe the behavior of the system [6]. It is possible to analyze the timing behavior, the memory usage and to check the logical behavior of the software. To see if the behavior of the system is correct, it can be tested on the real embedded system with its real environment, but this is very expensive and assumes the availability of the real system. However, in an early stage of the development, the real system is usually not available. Therefore the simulation is done at a high level for example on a workstation, which emulates the developed software and the environment, which makes a simulation and also changes of the embedded system very easy and cheap [5] because no special hardware is needed.

Moreover, if the simulation is done in an early stage, less redesign must be done in late development stages, which increases the design productivity[4].

This assignment is structured as follows. In the first part, a short introduction into the development process of automotive software systems (and embedded systems in general) is given which is generally based on the V-model. These foundations are used afterwards to show in which stages simulation is involved and which kinds of simulation exist. For this purpose, Model-in-the-Loop, Software-in-the-Loop and Hardware-in-the-Loop Simulationes are introduced.

After the theoretical part of this assignment, the next part describes how a simulation with the help of tools can be done in practice, if the existing software architecture is based on the AUTOSAR Standard. A tool and an extension of MATLAB/Simulink developed by dSPACE [7] is introduced, the differences are explained and the relationship between them is discussed. Afterwards, also a short overview about a tool from the bachelorproject 2007/2008 about the performance evaluation of AUTOSAR architectures is given and compared with the solutions of dSPACE. Finally, the differences between the simulation of AUTOSAR based system compared to the traditional systems is summarized.

2 Simulation in the Development Process

2.1 Development Process of Embedded Software Systems

To structure the process of developing software systems, many different process models are available. These are for example the waterfall model, the Rational Unified Process or the Extreme Programming. Another one is the V-model [9] which is very suitable for the development of embedded systems, because it focuses on quality assurance aspects which is given by the formal procedure, the creation of intermediate results and test activities on the different development stages[10].

The V-model consists of a number of phases. On the left hand you have the requirements analysis, system design, architecture and module design and at last the implementation. For every step, a test activity on the right hand is specified. These are unit testing, integration testing, system testing and user acceptance testing[9].

In addition to this, a system is usually developed in a sequence of intermediate product-appearances. The first appearance is the model, which is iterative developed. This means that first, low level requirements are established, detailed model design decisions are made and after that validated through model checking, state transition and model integration tests and of course a simulation of the model (see section 2.1.1). The resulting model should reflect an abstract view of the system, which emulates the required system behavior[8]. The process of developing an AUTOSAR based system is described in the AUTOSAR Methodology. It is not a complete process description but defines dependencies of activities on work-products[22]. For example it illustrates design steps from the system configuration until the generation of an ECU executable. This includes among other things that software components and the hardware have to be modeled, implemented, selected and mapped to ECU[22]. A detailed overview about this methodology can be read in assignment[19]. Tools supporting the development with AUTOSAR are introduced in section 3.

The next step in the development process is the automatically generation of code from this model. The code is embedded in an experimental hardware - the prototype. In the same manner as the model, the prototype is developed in its own V-development cycle. This means that on the right hand side we have activities like unit test, software integration and acceptance test, a system integration test and a simulation, too. The automatically code generation with different tools is also shown in section 3.

The last step is the replacement of the experimental hardware with the real one - the final product. For the final product, production requirements and detailed design decisions are made whereas on the right side of the V-model the system is tested, certificated and release criteria are checked. These three product-appearances (model, prototype and final product) leads to the so-called multiple V-model (see figure 1). The advantage of this separation is that it is cheaper and quicker to change a prototype or a model than to change the final product [8]. Moreover, if the model is validated to be correct, you just need to

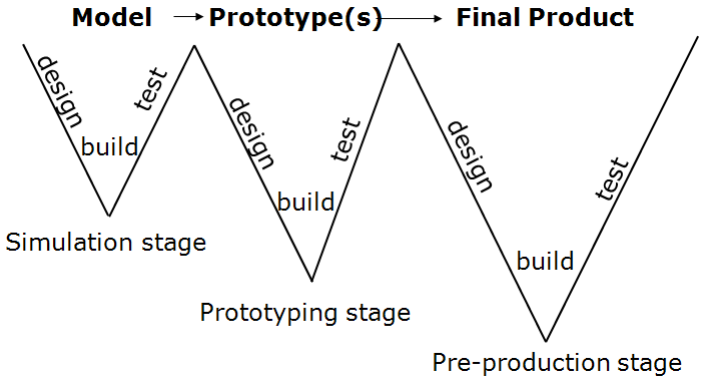


Fig. 1: Multiple V-model

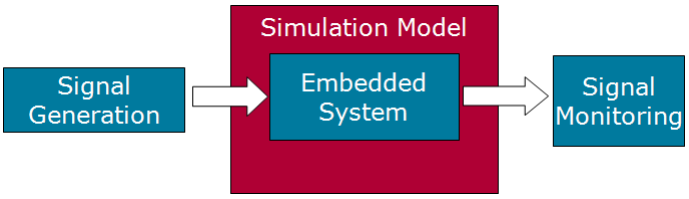
test the prototype against the model and do not need to start the development from the beginning.

However, the multiple V-model does not consider the fact that a complex system is not developed as one piece but split into smaller components which are developed separately. This development principle is one of the main aspects in AUTOSAR where a system is developed in seperate Software Components. This decomposition is usually done on the left side of the V-model in the design phase where it is determined which components should perform which task. Later, in the system integration stage the components are recomposed to a whole system. This principle, that a system is divided into components which can be divided into sub-components, sub-sub-components and so on where each component is developed in a separate V-development life cycle, is called the nested multiple V-Model [8].

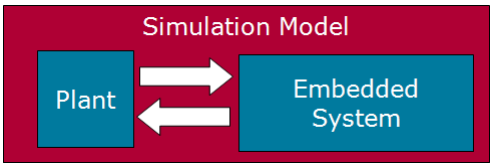
In every stage of the multiple V-model (model, prototype, final product) a simulation of the intermediate product is done. For that, special simulation and test methods are used. These are the Model-in-the-Loop Simulation and Rapid Prototyping in the model stage, the Software- and Hardware-in-the-Loop Simulation in the prototyping stage and the final product is tested by a system test in the last stage[8].

2.1.1 Model-in-the-Loop Simulation and Rapid Prototyping

After the requirements for the system are specified, the functional specification is done, which can be tested by a Model-in-the-Loop (MiL) simulation [12]. Therefore, the functional specification must result in an executable model - the simulation model. The goals of simulating the model are to prove the concept of the design, to optimize it and to develop and verify requirements for the next development step (prototyping). Testing and simulation in the model stage is usually done in the following steps: one-way simulation, feedback simulation and rapid prototyping [8].



(a) One-way simulation



(b) Feedback Simulation

Fig. 2: Simulation types

The one-way simulation means that input signals are generated manually or by a tool and fed into the simulation model. The algorithms described by the model are applied to the input and an output is generated which is recorded and analyzed (see figure 2(a)). However, there is no dynamic interaction between the plant and the simulation model.

The next step is the feedback simulation. This kind of simulation is more complex than the one-way simulation because a second model is used - the plant model. Instead of just generating signals and recording the output, the plant model behaves more dynamically. It prepares the input signals for the model of the embedded system creating an output which is fed back into the plant model resulting in a new input for the embedded system (see figure 2(b)) [8]. Such a plant model consist, among other things, of the motor, vehicle dynamics and the environment [11]. An example for one-way simulation as well as the feedback simulation will be shown in section 3. If the feedback simulation was successful and the real environment is available, the plant model can be replaced with the actual plant or a close equivalent [8]. A computer is connected to the sensor and actuators of the car and the model of the embedded system is executed. The computer transfers the output signals from the model to the actuators and the signals from the sensors of the car to the model. This is called rapid prototyping which offers a fast and early detection of errors in the specification and concept [12].

2.1.2 Software-in-the-loop Simulation

The Software-in-the-loop Simulation (SiL) is quite similar to the the MiL Simulation with the difference that not the model of the embedded system is executed, but the code, which is generated from that model. To execute the code, the code must be compiled first for a target processor. This can be the processor of the

host PC or the processor of the embedded system. In the first case, the software would run in an environment which is not restricted in resources and performance (a PC is usually much faster than an embedded system). The goal of this test is to verify the behavior and validate the simulation model from the model stage [8]. In the other case that the code is compiled for the processor of the embedded system, it can be executed on the host PC running in an emulator. The goal of this test is the verification of the correct execution on the target processor with restrictions on bit width and other processor specific limitations.

2.1.3 Hardware-in-the-Loop Simulation

The Hardware-in-the-Loop Simulation takes place in the prototyping stage. The difference to the SiL Simulation is that the test environment is not a PC or an emulator but it is a hardware part where the software is loaded [8]. First, the hardware part is just an experimental board, which contains the real target processor and memory but has only simple signal generators and an oscilloscope monitors the output of the embedded system. If the whole plant is still simulated by the PC and just the code is executed on the real target processor then it is also named a "Processor-in-the-Loop Simulation" [11]. Later, the experimental board is replaced by a prototype, which is nearly equivalent to the real system and is connected to hardware components of the car. With that, the embedded system can be simulated together with its surrounding hardware.

3 Tools for Simulation of AUTOSAR

In this section it is shown which tools are used in the above mentioned stages of the development life cycle process in the context of AUTSOSAR and which kind of simulations are supported by this tools. After introducing them, a brief comparison of the features is given.

3.1 MATLAB / Simulink / TargetLink

MATLAB is a tool by MathWorks, which allows to solve technical computing problems. It offers an add-on based environment to solve particular classes of problems [16]. One of this add-ons is called Simulink, which provides an interactive graphical environment for a Model-Based-Design and simulation of embedded systems. To model and simulate AUTOSAR based systems the add-on TargetLink is needed which is a tool by dSPACE providing an AUTOSAR blockset and a code generator. Modeling with TargetLink focuses on the application layer of the AUTOSAR architecture (for more information see assignment from [20]). It is possible to model AUTOSAR compliant SWCs and analyze their behavior on the host PC (MiL or SiL Simulation) or on an evaluation board (PiL). The code which is generated can be used for the target ECU [13].

The usage of TargetLink primarily takes place in a very early development stage of the multiple V-development cycle. After the requirements were verified, the developer can start with the functional specification of the embedded system.

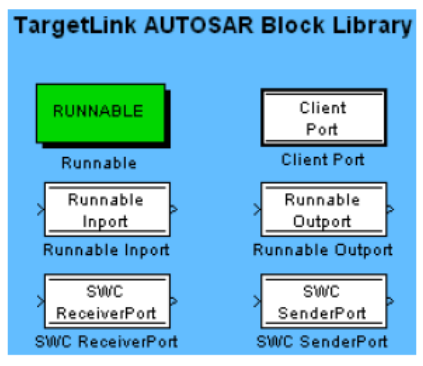


Fig. 3: TargetLink AUTOSAR blockset

To create a model, the blocks of the AUTOSAR blockset (see figure 3) and some certain blocks of Simulink can be used.

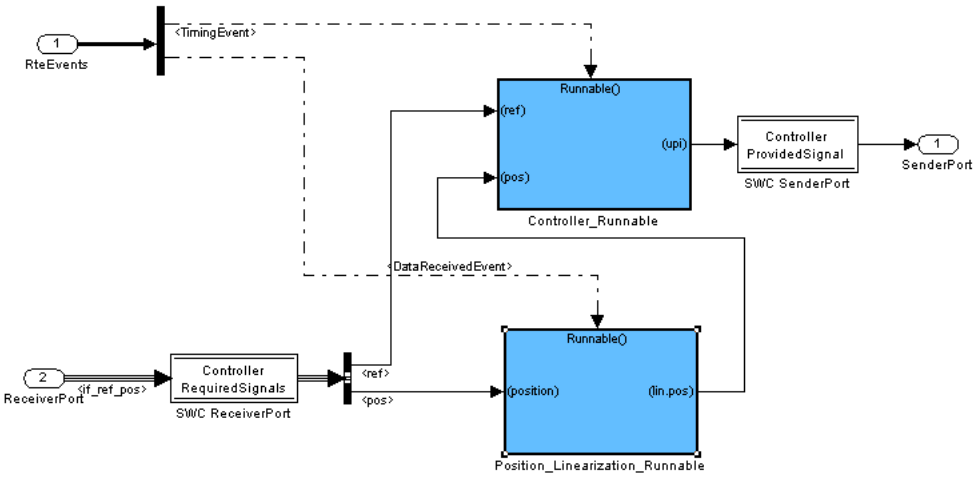


Fig. 4: Example of the inner view of a SWC

3.1.1 Principles of Modeling with TargetLink

AUTOSAR compliant SWCs can be modeled as TargetLink subsystems. However, they are not the same as SWC. "Modeling software components with TargetLink basically means modeling a software component's runnables and specifying each runnable's communication."[13]. This means that a subsystem in TargetLink is not just a container for Runnables like a SWC, but rather how

the Runnables are connected to each other. In anyway, the subsystems can be structured in a way that every subsystem corresponds to a SWC. However, it is not possible to nest subsystems into another one and so it is not possible to model AUTOSAR composition components. For communication with Runnables of other subsystems SWC In- and Outports of the AUTOSAR Blockset can be used. An example of a modeled SWC in TargetLink is shown in figure 4. There you can see how the Receiver Port of the SWC is connected to the Runnables and how the Runnables communicate among each other. Moreover, you can see that the subsystem gets two inputs, but has only one SWC input port. If a MiL simulation is done all ports would be used, but if code for the SWC is generated only the SWC input port is considered, because only this port is an AUTOSAR port.

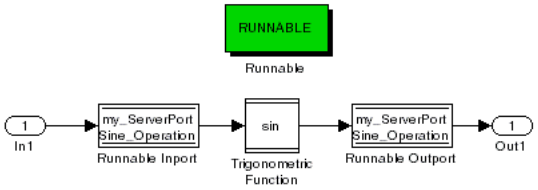


Fig. 5: Example for a Runnable which computes the sine of a value

To model an AUTOSAR compliant Runnable, the developer has to model a subsystem which describes the functional behavior of that Runnable and contains the Runnable-block (see figure 5).

The AUTOSAR standard specifies that Runnables are triggered by RTE-Events (see assignment [23]). This property can be set for a Runnables in TargetLink but is not regarded in the simulation. The execution of a Runnable is only based on the data-flow or a function-call-trigger. So, it is possible to simulate such RTE Events by modeling them manually as a stateflow [13]. An example is shown in figure 6, where the controller subsystem gets the RTE Events from a stateflow (left side) as an input which are redirected to the runnables (see figure 4).

The communication between Runnables can be modeled with the In- and Outport blocks of the AUTOSAR blockset. They can be used to transfer data elements, operation arguments and interrunnable variables.

Runnables can provide operations, which are triggered by an OPERATION_-INVOKED_EVENT RTE event. For that, they have to act as a server. A server Runnable, for example, consist of an inport block, the operation block and an outport block (see figure 5). To call this server, the client must uses the Client Port block which transfers the arguments to the Runnable and get the results from it.

3.2 Simulation with Simulink / TargetLink

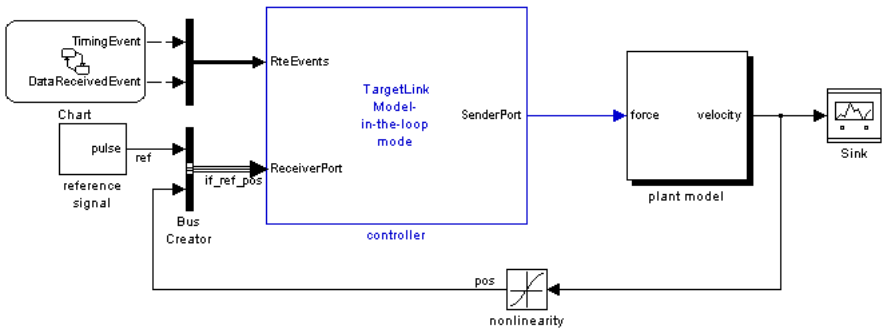


Fig. 6: MiL Simulation

TargetLink supports three modes of simulation: MiL, SiL and PiL. The MATLAB/Simulink engine is used to simulate TargetLink models. Because of this, TargetLink blocks are just enhanced Simulink blocks with code production properties.

A typical MiL scenario with a Feedback Simulation is shown in figure 6. On the left side we have a reference signal which generates an input for the model of the embedded system. The calculated results are fed into the plant resulting in a new input for the model of the embedded system. To observe the behavior of the system, a so-called "'sink"' (right side) is appended to the system which generates a graphical output of certain values during the simulation. Such an output is shown in figure 7.

After the behavior of the system in the MiL simulation is verified, the simulation type can be changed to a SiL simulation, which takes place in the prototyping stage. To be able to execute a SiL simulation, code must be generated first from the model. The code which is generated is an AUTOSAR compliant C code, one the one hand the code of the SWCs is generated and on the other hand the code of the RTE. Because you cannot define any deployment of SWC to ECUs, the behavior of the RTE is just as it is executed on one ECU. However, the code of the SWC can be used for example in SystemDesk which can generate ECU-specific RTE code which is illustrated in the next section. For any other non-AUTOSAR elements in the model like the signal generator in figure 6 or the sink, no production code is generated. If the SiL is in progress, also non-AUTOSAR elements are executed from the simulation engine as it was done in the MiL simulation. For more information how the data is transfered between non-AUTOSAR blocks, generated code and the simulation engine see [14].

Because the SiL Simulation takes fixed-point effects into account (in opposite to the MiL simulation) the simulation results can differ from the MiL results. In

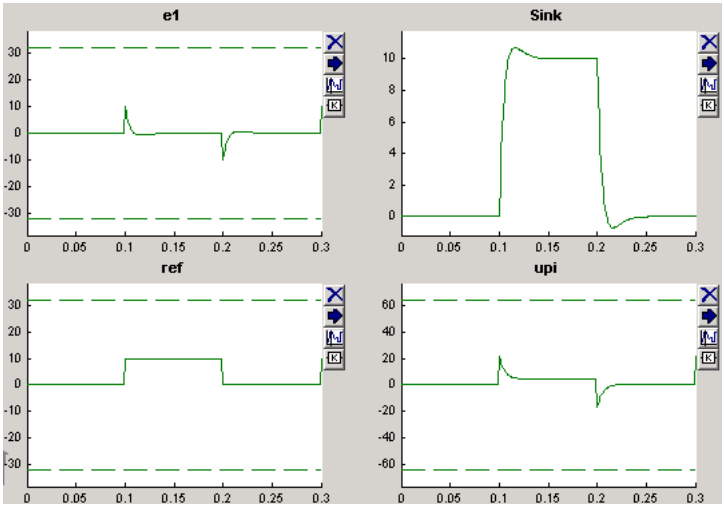


Fig. 7: Output from a MiL Simulation

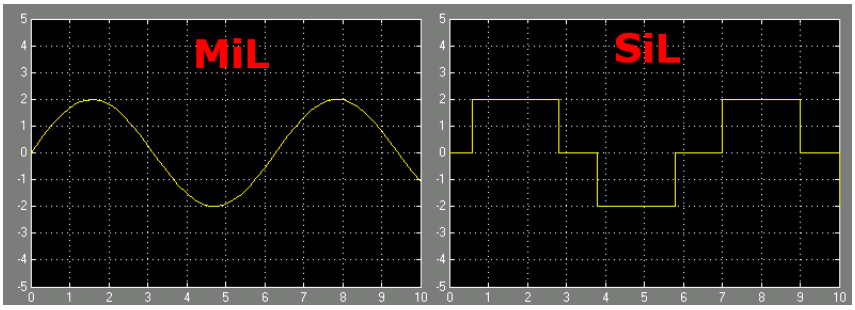


Fig. 8: Calculation of a sine curve in MiL and SiL with wrong dimensioned data types in the code

figure 8 a sine curve is computed in the MiL and SiL mode. Because the data type in the model had no restrictions (32 bit floating point) but was just an 8 bit integer (restrictions of embedded system) in the generated code, such big differences can occur.

To check wheather the generated code also works on the real hardware, the simulation mode can be changed to PiL mode. In this mode, the target processor specific compiler must be chosen. After the code is generated and compiled, it must be downloaded to the evaluation board. Now, the system can be simulated with hardware specific effects like limited stack size, bit width and performance restrictions.

3.3 SystemDesk

SystemDesk by dSPACE is "a tool supporting the development of distributed automotive electrics/electronics systems according to the AUTOSAR approach."[17]. In the multiple V-development cycle it can be attached to the modeling stage as well as the prototyping stage. One the one hand, you can use SystemDesk to model the software architecture with AUTOSAR Software Components including Runnables and Ports but also to model the hardware topology with the ECUs and the network communication with busses. These modeling activities taking place in the modeling stage. On the other hand, you can also use SystemDesk to generate C code and based on that, execute a SiL, simulation which is done in the prototyping stage. For more information about how to model in SystemDesk, see assignment [18]. This section just focuses on simulation possibilities in SystemDesk and how models can be interchanged with TargetLink.

3.3.1 Simulation in SystemDesk

SystemDesk provides a feature for a non-real time SiL simulation. The simulation is based on the C code of the SWCs and the RTE. The C code for the SWCs can be token from the production code generation of TargetLink. The code for the RTE is generated by SystemDesk depending on the deployment of SWCs to ECUs and the ECU configuration. SystemDesk uses a C compiler which generates Windows DLLs for the simulation engine[17].

SystemDesk supports one-way simulations as well as feedback simulations. For the one-way simulation you can use stimulus generators or replay recorded data from a previous simulation. The feedback simulation requires a model of the plant, which can be importeted from a Simulink model.

The documentation[17] of SystemDesk gives some important properties for the simulation process. The simulation is executed in "virtual simulation time", which means that the time passes faster or slower on the host pc compared to the time on the target processor. However, it is possible to slow down/speed up the simulation clock to approximate the real time.

The scheduling, which is based on the configuration of the operating system is considered, which means that the correct order of tasks and runnable calls are simulated.

SystemDesk makes a "Zero time assumption", which means that tasks are executed instantly in virtual simulation time. From this, it follows that it is not possible to detect time critical effects, for example if a deadline is met or not.

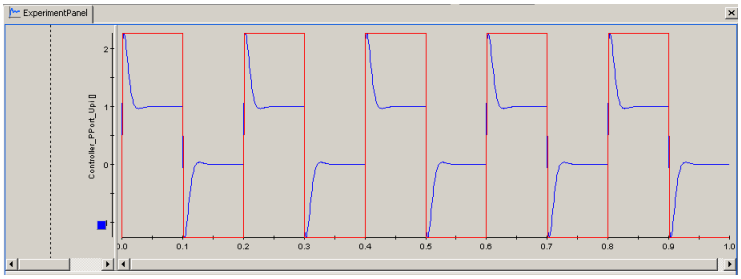
SystemDesk provides a simulation at different verification stages which are described in [17]. These are the logical level verification, system level verification and the implementation verification.

On the logical level the Virtual Functional Bus Concept(VFB)[21] defined by AUTOSAR is used. The VFB concept states that SWC are implemented independently from the underlying hardware (at this stage, the hardware topology in SystemDesk is not specified). The VFB is a communication mechanism where the SWC can exchange data, not knowing weather the data transfer happens on the same ECU or to another one over a bus[21]. More information about the VFB concept can be read in assignment [23]. SystemDesk provides a simulation of the VFB, which requires the architecture of the SWC, the internal functional behavior and the connection between the SWC. All SWCs are automatically mapped to one ECU which is called the Virtual Processor Unit (VPU). Runnables are usually activated by Tasks, however, because there is no OS configuration available in the logical level verification stage, a default scheduling is computed by SystemDesk. This looks like the following: Cyclic Runnables are mapped to a cyclic task and another task takes care of acyclic Runnables which are triggered by RTE Events[17].

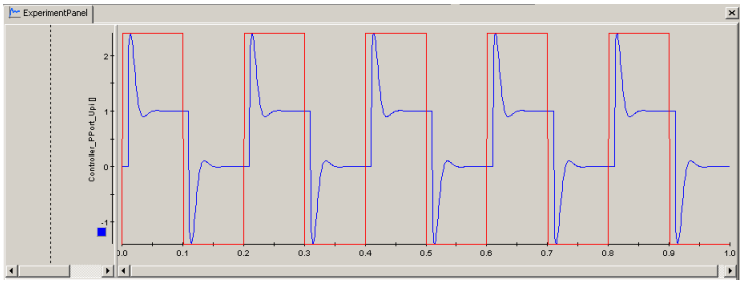
If the development proceeds and the information about ECUs as well as the buses are available and configured in SystemDesk, further effects can be simulated in the so-called system level verification stage. The properties of this stage are that the SWCs have been mapped to ECUs and more details of SWC are defined. SystemDesk provides a feature to simulate the effects of the busses. For a CAN Bus the simulation can take into account the arbitration and the bus capacities. So it is possible to simulate the bus usage as well as the communication delays.

If the production code and the production ECUs are known, a more detailed simulation can be executed in the so-called implementation verification stage. The timing of the application is determined by the underlying operating system, which considers scheduling effects. To simulate this scheduling effects, SystemDesk uses an AUTOSAR compliant operating system, which simulates the correct execution order of task and Runnable calls. However, the execution time is usually faster on the host pc than on the target hardware. To get better results which are more realistic, it is possible to slow down the SystemDesk clock.

An example of a simulation result is shown in figure 9(a) and 9(b). Both figures show a measurements of two different variables. The rectangular signal is created by the Plant SWC and the other signal shows the calculated variable of a Controller SWC. In figure 9(a) both SWCs are deployed to one ECU. It can be seen that the controller responds immediately if the edge of the plant is rising. In opposite to figure 9(b) where the SWCs are deployed to another



(a) Simulation with one ECU



(b) Simulation with two ECUs

Fig. 9: Simulation results in SystemDesk

ECU (which are connected over a bus). Such a reconfiguration can be done very easily in SystemDesk. It can be seen that the controller responds delayed because the signal of the plant needs some time to be transferred over the bus. So, it is possible to try different configurations of the system in SystemDesk and to check what are the effects. This possibility is a very important advantage of AUTOSAR based system.

3.3.2 Relationship between SystemDesk and TargetLink

Both tools can be used in an early development stage as well as in the prototyping stage where code is generated and afterwards simulated. The order, in which the tools are used in the development process, can be different. On the one hand, it would be possible to model the software architecture first in SystemDesk and then the functional behavior in TargetLink, which generates code that is used for the implementation of the SWC in SystemDesk. On the other hand, it is also possible to define the functional behavior first in TargetLink and then the software architecture in SystemDesk. Finally, both tools can be used in parallel to design an automotive system.

TargetLink has a feature which is called Data Dictionary (DD). The DD is "a container that holds all relevant information about an ECU application"[24]. This includes tasks, variables but also AUTOSAR SWCs. Figure 10 shows a diagram where you can see which tools are connected to the DD and which

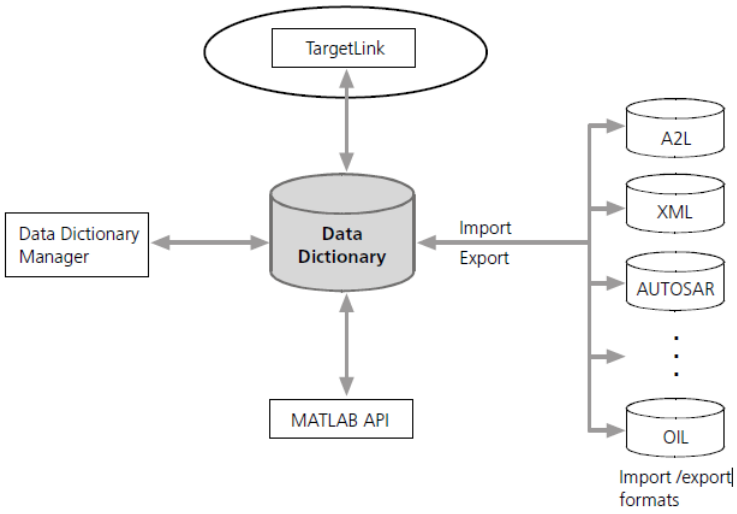


Fig. 10: Data Dictionary

formats are supoorted to export and import the data. If for example a subsystem in TargetLink is modeled, a SWC can be created in the DD to reference that this subsystems is mapped to a SWC. This can also be done with the Runnables, and their ports. The advantage of that DD is that the data is kept consistent for example between the TargetLink model and the AUTOSAR model. If one is changed, the other one is changed too. Moreover, it is possible to export the AUTOSAR model as an XML file and to load this XML into SystemDesk. A use case is, for example, that you first create your software architecture in SystemDesk, export this to XML and importing it into the DD. Afterwards, the DD objects are mapped to TargetLink objects for example a SWC to a subsystem and so on. And the development of the functional behavior of the SWC can start at this point and doesn't need to modeled again.

SystemDesk also offers a possibility to integrate Simulink models as atomic software components [17].

3.4 Bachelorproject 2007/2008

In the bachelorproject 2007/2008, a concept was developed how a simulation of an AUTOSAR modeled system can be done. For this purpose, the exisiting simulation tool chronSim[4] was used, which was based on a task model and a model for describing the hardware architecture (with busses), the deployment and real time requirements. The goal was to define a transformation from an AUTOSAR system to the traditional system with that a simulation can be executed. In addition, real-time requirements were defined in the AUTOSAR model which are considered in the evaluation of the simulation results.

For the simulation, the code of the tasks is compiled by chronSim and executed on the simulated target hardware. It can be characterized as a SiL simulation with similarities to a HiL simulation, because of the strong consideration of hardware aspects during the simulation.

4 Summary of the Differences in the Simulation of traditional and AUTOSAR based Systems

In section 2 of this assignment, the process of the development of automotive software systems in general including the simulation of such systems was illustrated. In section 3 it was shown, which tools are available to simulate AUTOSAR based systems. This section should give a small summary about the benefit of using AUTOSAR to simulate systems and what are the differences to the traditional methods.

Today, many innovations in a car are realized through softwre. The increasing complexity as well as the growing size of the systems leads to an outsourcing of the functionality to automotive suppliers. The suppliers develope their components in their own life cycle and use their own tools. In the past, the result was that every supplier had their own description of their produced components and the automobile manufacturer had to integrate the components of the different suppliers. However, they usually had of adapt their own code to make it working with the third-party components. With AUTOSAR, a standardized definition of the interfaces is available and for this reasons also a standardized testability of the components can be managed. The code doesn't need to be adapted if a new component is simulated from a different supplier which saves times and money. Moreover, the configuration of the whole system like a change in the deployment of SWC to ECUs can be done very easily and the new configuration can be simulated instantly.

With AUTOSAR, the application is independent of the hardware architecture. This new approach leads to an easier testability of components. In the past, the software was strongly coupled with the underlying hardware and therefore, the possibilities of a pure SiL simulation were limited. AUTOSAR considering this problem and introduced a VFB. As it was shown in section 3.3.1, a VFB simulation is possible which has the main advantage that it can be done in an early development stage and reduced the risk of finding errors in the software architecture very late which can lead to a big redesign and high expenditures.

Finally, the AUTOSAR consortium consists of many different automobile manufacturers, automotive suppliers and tool developers and it is still growing. Therefore, it can be expected that more and more tool support for simulating AUTOSAR architectures will be available in the future.

References

[1] Simulation Article, Roger D. Smith, 1998
[2] http://www.advanced-planning.de/advancedplanning-128.htm
[3] Computer Simulation: The Art and Science of Digital World Construction, Paul A. Fishwick
[4] http://www.inchron.de
[5] Using simulation tools for embedded systems software development, Jakob Engblom, Virtutech, 2007
[6] Software Engineering for Embedded Systems, Verification & Validation, Holger Giese, 2008/2009
[7] http://www.dspace.com
[8] Testing Embedded Software, Bart Broekman and Edwin Notenboom, 2003
[9] V-Modell 1997: Entwicklungsstandard für IT-Systeme des Bundes, "'http://www.v-modell.iabg.de/vm97.htm"'
[10] Softwareentwicklung eingebetteter Systeme, Peter Scholz, Axel Springer Verlang, ISSN 1439-5428
[11] Automatisierter Closed-Loop-Softwaretest eingebetteter Motorsteuerfunktionen, Sven Rebeschieß, Thomas Liebezeit, Uzmee Bazarsuren,
[12] Testautomatisierung in der Hardware-in-the-Loop Simulation, Dr.-Ing. Clemens Gühmann, Dipl.-Ing. Jens Riese, IAV GmbH Berlin
[13] AUTOSAR Modeling Guide, TargetLink 3.0, July 2008, dSPACE
[14] Production Code Generation Guide, TargetLink 3.0, July 2008, dSPACE
[15] Paper zum 3. Workshop Automotive Software Engineering, Informatik 2005
[16] http://www.mathworks.com
[17] SystemDesk Guide For SystemDesk 2.0, Release 6.3, November 2008, dSPACE
[18] Modeling of AUTOSAR using SystemDesk, Sebastian Wätzoldt
[19] AUTOSAR Methodology & Templates, Regina Hebig
[20] Software Architectue, Warschofsky
[21] Specification of the Virtual Functional Bus, http://www.autosar.org
[22] Technical Overview, http://www.autosar.org
[23] Runtime Environment & Virtual Function Bus, Nico Naumann
[24] Data Dictionary Basic Concepts Guide, TargetLink 3.0, July 2008, dSPACE