To appear in the special issue "Mechatronics in Automotive Systems" of the International Journal of Vehicle Design

Dr.-Ing. Ralf Rothfuß received the Diploma in Technical Cybernetics at University of Stuttgart in 1992. In 1997, he completed his PhD in the area of non-linear control. He joined Robert Bosch GmbH in 1997 and is currently working in the area of modelling, simulation and control of mechatronic systems. Areas of interest: non-linear control, modelling, simulation, CAE-environments and industrial design process.

Dipl.-Ing. Mikel Lasa studied mechanical engineering at the Universities of Navarra (Spain) and Compiègne (France). He joined Robert Bosch GmbH in 1997, where he is pursuing a PhD in the mechatronic domain.

Dipl.-Ing. Hans-Martin Heinkel studied Electrical Engineering at University of Stuttgart. Since 1986 he is working at the central research department of Robert Bosch GmbH. Areas of interest: Design of mechatronic systems, real-time simulation.

Dipl.-Ing. Pierre Tirgari graduated in 1995 from the Ecole Supérieure des Sciences et Technologies de l'Ingénieur de Nancy (FRANCE) in the Department of Fluid dynamic Engineering. Between 1995-1997 he worked as research engineer in the Environmental and Applied Fluid dynamics Department of the von Karman Institut for FLUID DYNAMICS, Brussels. Since May 1997, he is working for Robert Bosch GmbH in the research and pre-development division in the department of simulation of mechatronic systems. His scientific areas of interest are the modelling of hydraulic and magnetic components.

Key words: Systems engineering, modelling and simulation of mechatronic systems, model exchange, design process, flatness-based non-linear control, Electronic Stability Program.

Complex mechatronic systems, for example brake or injection systems (ESP, Common Rail), consist of a large number of components from different physical domains. The interaction of these components is essential for the function and behaviour of the overall system.

The design of mechatronic systems can be facilitated using a methodology called systems engineering [8, 30,36,13]. Systems engineering unifies the product and process issues in the design process of mechatronic systems so that the design methodology, design process, CAE-environment and the organisational structure of the company are considered simultaneously.

As a consequence, the controller and plant have to be planned at the same time. The controller and plant are designed in order to realise functions such as braking force regulation and control of the air-mass flow. The plant can be changed during the design process in order to fulfil a pre-defined function. By contrast, the classical control problem consists of designing a controller for a given - and usually unchanged - plant.

Furthermore, different models are used to describe different viewpoints of the plant during the design process. These models are usually built using different simulation tools due to the multi-domain nature of the mechatronic components. As a result, the model exchange between different simulation tools plays a crucial role in the design of mechatronic systems.
1. INTRODUCTION

Today, the reduction of development costs and time-to-market dominates the design process of almost every technical product. This is especially true for mechatronic systems in the automotive industry. These systems consist of components from different physical domains, which are closely coupled and therefore interact with one another [19]. Moreover, the behaviour of the components is highly non-linear. The focus of this paper stems mainly from the automotive industry which is highly competitive and where cost and time requirements play a crucial role.

For time and cost reduction, it is fundamental to achieve a first-shot-yield during the design process. Use of modelling and simulation is therefore essential. Moreover, dividing the system behaviour into different functions (e.g. braking, control of the air-mass flow, etc.) facilitates the handling of complex mechatronic systems. The aim of this process is to achieve the desired overall functional behaviour (i.e. braking of the car) rather than designing a component in a particular domain (e.g. the main brake cylinder or the ESP controller). The design process is an appropriate combination of "top-down" and "bottom-up" methods in order to meet customer requirements. As a result, the plant and controller have to be designed simultaneously due to the high degree of interaction.

During the design process a large number of simulation runs are required using models at different levels of abstraction. Simulation models are required in particular during the early stages of the development process for the verification of alternative designs and parameterisations. Furthermore, these models are necessary for understanding the overall system. Since mechatronic systems cover a wide range of physical domains including control, specialised simulation tools are available in the field of mechatronics. Unfortunately, the exchange of simulation models is difficult because of different numerical methods, model description languages, etc., of the individual tools.

Nevertheless, the behaviour of mechatronic systems and their components has to be verified in the early stages of the design process in order to reduce costs and time-to-market even if hardware...
samples are not available. As a result, model exchange is an essential requirement for the design of mechatronic systems.

The only way to fulfil these demands is the extensive use of systems engineering methods [33, 8, 30, 36, 13] throughout all stages of the design process. Systems engineering provides a unified methodology including both product and design process. The design methodology and the design process are considered together with the CAE tool environment and the organisational structure of the company. Hereby, design decisions can be verified with customer demands even if hardware prototypes of the system are not yet available at the time for an experimental verification. Establishing complete test criteria is therefore an essential part of the design process.

In this paper, the use of systems engineering for the design process of mechatronic systems is described using the example of an ESP system [40]. The next section briefly describes the structure of the ESP system with some focus on the engine management system and the hydraulic unit. By considering the air-mass regulation within an electronic throttle control (ETC), typical requirements for a safety-related design of automotive systems are presented. In Section 3, the results are generalised and an overview of the general design process is given based on a V-cycle [8, 30, 28, 6].

Particular problems arise during the modelling of mechatronic systems due to the various physical domains involved. In general, excellent results for all physical domains cannot be obtained from one simulator. Therefore, the simulation of the entire system requires the system level models of the components to be exchanged between different simulators. The various aspects of this model exchange will be discussed in Section 4 and a pragmatic approach will be introduced to overcome the important problem of model exchange [17].

Fig. 2: Simplified schematic of an ESP system with inputs and outputs including engine management and ABS/ASR functions [15].
2. MECHATRONIC SYSTEMS – THE ESP SYSTEM AS AN EXAMPLE

In the following section, the ESP system is used as a typical example of a modern mechatronic system. It consists of a large number of components that are used to achieve certain functions (braking, stabilisation of yaw motion, etc.). Many of these components have to be described using non-linear models in order to cope with the wide operation regime. The ESP system has an interface to the engine management and the brake system. Lastly, it is a safety-critical part of the car.

ABS and ASR functionality has now become almost standard in modern passenger cars. The ESP (Electronic Stability Program) algorithm not only improves the basic ABS and ASR functionality in critical situations with respect to longitudinal dynamics (complete deceleration, acceleration), it also supports the driver in critical situations with regard to lateral dynamics by initiating highly dynamical active braking at the four individual wheels based on a modified hydraulic ABS/ASR unit. The ESP attempts to keep the longitudinal, lateral and yaw velocity of the vehicle within certain limits in order to guarantee the stability of the vehicle [14, 15]. For this purpose, according to the simplified block diagram in Figure 2, the ESP algorithm controls the state variables yaw velocity and side slip angle, and calculates an appropriate yaw torque that enforces the desired behaviour of the vehicle. The non-measured quantities are estimated using a dynamic observer.

The ESP algorithm influences the ABS brake slip and ASR drive slip control using the hydraulic modulator as an actuator. In addition, the yaw torque control is influenced using an interface to the engine management. The hydraulic unit is shown in Figure 3. It mainly consists of the brake master cylinder (1), pressure sensor (2), precharge pump (3), and the hydraulic modulator (4). The latter comprises the floating circuit (5), different valves (suction, switching) (7)-(8), damping chambers (9), return pump (10), inlet and outlet valves (13)-(14), and the wheel brakes (15).

Fig. 3: Hydraulic modulator of the ESP system: 1 brake master cylinder, 2 pressure sensor, 3 precharge pump, 4 hydraulic unit, 5, 6 braking circuit, 7 suction valves, 8 pilot valves, 9 damping chamber, 10 return pump, 11 non-return valves, 12 accumulators, 13 inlet valves, 14 outlet valves, 15 wheel brakes [6].
A description of the design process for the entire ESP would exceed the scope of this paper. Therefore, only the throttle plate shown in Figure 4 is considered in the following section. The throttle plate is by itself a mechatronic system used for controlling the air-mass flow in the injection system as shown in the following chapter.

2.1 Control of the Air-Mass Flow

As shown in the previous section, the ESP uses an interface to the engine management in order to control the vehicle. The most essential part of the engine management is the ETC (electronic throttle control). The following section describes two important aspects in the design process of the ETC:

a) The function of the ETC is to control the air-mass flow. For environmental reasons, a precisely controlled low idle speed is important in order to achieve low fuel consumption.

The corresponding control loop with the A/D converter has the following structure:

During the idle mode, the throttle plate is operated with relatively small values of $\phi$. For this, either a high resolution A/D converter has to be used in combination with a simple controller or a less expensive A/D converter together with an advanced controller that can handle the effects caused by a rough discretisation. Therefore, the function of the entire system has to be optimised with respect to cost and precision requirements around $\phi=0$. If neither solution is satisfactory, the mechanical structure of the plate has to be modified.

In general, the plant may have to be modified during the design phase of the controller for a mechatronic system. This may result in a complete redesign of the control scheme (including the controller). This differs from the situation of the classical controller design where the plant is already defined and the design is relatively fixed. Nevertheless, a control-oriented point of view during the design process can save time and reduce costs when the controller and plant are designed simultaneously to achieve the desired function.
b) When realising a desired function of a mechatronic component, it is often useful to formulate the design problem as a tracking problem, i.e. by imposing a motion of the state variables of the system along a predefined desired trajectory. The tracking problem is important in the area of mechatronics.

For the throttle plate, the control task consists of imposing a rest-to-rest motion

\[ \phi_f(t) : \phi(0) \to \phi(T) \text{ with } \dot{\phi}(0) = \dot{\phi}(T) = 0 \]

for the throttle angle \( \phi \) within a predefined time interval \( T \). The system can be described for the controller design using a simple model based on the following differential equations for the state variables \( x = [I, \phi, \dot{\phi}]^T \) [34]:

\[
\begin{align*}
L \dot{x}_1 &= -Rx_1 - C_m x_2 + u \\
\dot{x}_2 &= x_3 \\
J \ddot{x}_3 &= C_m x_1 - C_f(x_2) x_2 - D x_3 + M(x_2, x_3)
\end{align*}
\]

The coefficients have the following meaning: \( R \) is the resistance of the motor, \( C_m \) is the motor constant, and \( L \) describes the inductance of the motor. The constant \( J \) is the moment of inertia, \( C_f(x_2) \) represents the mechanical spring constant, and \( D \) describes the mechanical damping. The torque in (1c) is described as

\[ J M(x_2, x_3) = C_f(x_2) \dot{x}_3 - M_{pre}(x_2) - M_{fr}(x_2) \text{ sign}(x_3) \]

In addition to this nominal function of the throttle plate, an important safety requirement has to be fulfilled for the ETC. If the power supply of the electric motor fails or if an error occurs in the control scheme, the flap returns to the limp home position. This position ensures a minimal airflow in order to keep the engine running. A consequence of this safety feature is that some parameters of the model state equation (1) are different for angular blade positions over and under the limp home position \( \phi_{LHP} \). Thus, the following relations are obtained for the spring constant \( C_f(x_2) \), the pretension of the springs \( M_{pre}(x_2) \) at \( \phi_{LHP} \), and for the Coulomb friction \( M_{fr}(x_2) \):
This yields a model (1), (2), which is non-linear with a sudden change in the parameters around the limp home position $\phi_{LHP}$.

The control problem for (1), (2) is to follow a desired trajectory $\phi_d(t)$ for the angular position $x_2$. This tracking problem has an elegant solution if the system is differentially flat in the sense of [10, 11, 12, 35, 31, 25, 3]. The flatness property can be exploited in order to design a control input $u$ that imposes a desired trajectory $\phi_d(t)$ and stabilises the motion around this trajectory (refer to [34] for details).

\[
C_f(x_2) = \begin{cases} 
C_{f1} : x_2 > \phi^* \\
C_{f2} : x_2 < \phi^* 
\end{cases} 
\]  

(2a)

\[
M_{pre}(x_2) = \begin{cases} 
M_{pre1} : x_2 > \phi^* \\
M_{pre2} : x_2 < \phi^* 
\end{cases} 
\]  

(2b)

\[
M_{fr}(x_2) = \begin{cases} 
M_{fr1} : x_2 > \phi^* \\
M_{fr2} : x_2 < \phi^* 
\end{cases} 
\]  

(2c)

Fig. 5: V-cycle of the design process for mechatronic systems [8, 28, 6].

The requirements for designing a controller for mechatronic systems can be summarised as follows:

- The overall objective is to realise certain behaviour for the entire system (controller and plant). The optimisation is based on the overall objective.
- In classical control, one often tries to build a rather complex controller for a single and well-defined plant. Mechatronic controller design has to deal with multiple plants with possibly changing structures during the design process. Simple controllers (as well as simple plant models) are often used here.
- It is of increasing importance to use controller design methods for non-linear mechatronic systems that allow the predefined desired trajectories to be tracked in a wide range of operations. An already well-established approach is one based on flatness (refer to the industrial applications in [39, 20, 34, 32, 2, 4] and the references therein.).
3. DESIGN PROCESS FOR MECHATRONIC SYSTEMS

In this section, the aspects from the previous section are collected in a more general description of the design process of mechatronic systems from a systems engineering point of view [22, 23, 24]. The design process of mechatronic systems can be divided into several stages as shown in Figure 6. During the system requirement analysis, the supplier and the client specify the desired functions of the product. This results in a written specification that forms the basis for the subsequent design process.

Figure 6: Design process for mechatronic systems.

The functional behaviour of the entire system is defined based on this specification and decisions regarding design can also be made at the system design level (e.g. the decision to use a solenoid as an actuator rather than a piezo actuator). Hereafter, single design processes for the components can be defined on the component design and development level.

Since the whole design process is neither "top-down" nor "bottom-up", it can be represented in the well-known V-cycle [8, 28, 6]. Figure 5 shows the use of simulation models at different levels. The models can be used to test the behaviour of the component versus test cases that are derived from the specification. Therefore, the proper function of the components can already be verified at early stages of the design process using simulation models even if hardware samples are not yet available. The test cases can be applied to the real hardware at a later stage (see Figure 5). For a more detailed description of tests performed during the early stages of the design process see [37, 16]. Note that recursions in the early stages of the process (based on models) are fast and inexpensive whereas ones used in the later stages of the process (using hardware samples) are expensive and time-consuming.

In the following section, the hydraulic modulator as shown in Figure 3 is used to discuss some aspects of the design process. Basic design questions can often be answered by means of rather simple component models. More complex models (on the component level) are required for decisions relating to details. The focus of the latter is mostly on a particular physical domain:
Detailed models of the hydraulic valves and pumps are available, which are based on physical and geometrical parameters. These models largely depend on the numerical capabilities of the simulation tools. These tools are specialised for the simulation of hydraulic effects such as the transition from laminar to turbulent flow, cavitation, etc. The mathematical basis is formed mostly by ordinary differentials or differential-algebraic equations.

Specialised simulation tools are also available for the electrical and magnetic domains. These tools are specialised for conservative models (energy conservation, bi-directional signal flow) that can be expressed in terms of differential-algebraic equations.

The controller is mainly designed using non-conservative models (unidirectional signal flow). Thus, the resulting mathematical system of ordinary differential equations can often be solved sequentially by sorting the equations.

In the mechatronic domain a large variety of domain-specific simulation tools are used. Some of these simulation tools are listed below:

- **AMESim** [18]: Simulation of hydraulic systems with geometry-oriented and physical parameterisation.
- **ASCE-T/SD/CT** [9]: (Real time) simulation of mechatronic systems, discrete and analogue simulation, and development environment for embedded control systems.
- **MATLAB/Simulink** [27]: Signal processing, control design, (Real time) simulation of mechatronic systems, and development of automatic control systems.
- **Saber** [1]: Electronic circuit simulator; simulation of mechatronic systems.

Since the interaction between components is an essential property of mechatronic systems, simple models of the neighbouring elements (e.g. the solenoid, power stage) are required for designing particular components (e.g. the hydraulic part of a valve). The simulation of a detailed hydraulic valve model, for instance, requires a simple model of the power stage, and vice versa. Use of different simulation tools during the design process is only efficient if the model can be exchanged between the tools. Direct model exchange is difficult due to tool-specific modelling languages, different numerical solvers, etc.

### 4. MODEL EXCHANGE

An analysis of mechatronic systems shows that the component models can usually be built using just a small number of domain-specific elements. These are called the standard elements [22, 23, 24]. In the hydraulics domain typical examples of these standard elements are an orifice, pressure-relief valve, chamber, pipe with and without wave propagation, in the electronics domain, resistor, capacitor, transistor, etc., as well as DC motors, AC motors, and in the mechanics domain, a mass with friction and end stops, spring, gear box, clutch, etc. In the following section, a pragmatic approach is presented to overcome the aforementioned problems of model exchange between different simulation tools. At the centre of this approach is the Robert Bosch Standard Simulation Library [21, 22, 23, 24].

### 4.1 Robert Bosch Standard Simulation Library

The main features of this library are the following:

- The library provides a structured collection of models for some important elements of each domain. These elements are referred to as standard elements.
- Each standard element is described by means of mathematical equations (including a detailed description of parameters, state variables, inputs, outputs, state events, etc.) that provide a physical understanding. These equations are exactly the same for all implementations of the standard element in the different tools supported by the library. Tool-specific details (such as
handling of state events, step size control, etc.) are taken into account so that the behaviour of the standard element is the same in each simulation tool.

- The library contains physical documentation for each element as well as a complete list of parameters, state variables, inputs, and outputs. The availability of this documentation does not depend on the use of a particular simulation tool. The documentation and the model are accessed via Intranet (Figure 7) in order to avoid dependence on the computer platform in use.

- During the first steps of the design process (see Figure 6), the specification of the ESP valve is given in terms of a simple model at the system level. This is implemented in one tool (e.g. ASCET) using standard elements.

- This simple model can be built in other simulation tools such as SABER or AMESim by means of the tool-specific schematic entry. The structure is copied using the standard elements (Figure 8). The problem of model exchange is reduced to the simpler problem of parameter exchange. For this, a computer-aided solution can be provided.

- Standard elements are also useful for the detailed design of components. Each simulation tool contains specialised models for its original domain. These are called core elements. Since interaction between different domains plays a crucial role for components of mechatronic systems, one has to take into account the influence of neighbouring domains. For the detailed design in AMESim of the hydraulic part of the ESP valve, for example, a simple model of the solenoid is necessary in order to represent the influence of the magnetic circuit. In the same way, for the detailed investigation of the solenoid in SABER, a simple model of the hydraulics is required. In both cases, the simple models can be built using standard elements.

- Once detailed component models have been built and validated using measurements, one can easily fit the parameters of the simple model with standard elements using the results of the more detailed model in one simulation tool. It is therefore easy to keep all the models depicted in Figure 8 up-to-date during the whole design process by propagating the parameter values from one tool to the other.

The RB Standard Simulation Library approach has the following advantages:

- A structured set of models is available at an appropriate level of abstraction. While in the case of co-simulation a large number of parameters are used to describe the entire system, the
models in the approach presented above include only the parameters that are needed for investigating the current problem. The understanding of the problem is thereby augmented.

- In most cases, this allows us to solve the problem with only one simulation tool, which is more appropriate from a user’s point of view.
- The parameterisation of the models in different simulation tools can easily be kept consistent.

The RB Standard Library currently supports the simulation tools ASCET-SD/CT, MATLAB/Simulink, AMESim, and SABER. In addition, each component model is described using VHDL-AMS [17] as a unified representation. Each tool that supports VHDL-AMS is automatically included in the Standard Library approach. The VHDL-AMS description is considered to be a reference since its behavioural semantics are well defined. These descriptions shall be used as a single source for all simulators using code generation.

![Diagram of model exchange using the Robert Bosch Standard Simulation Library.](image)

**Fig. 8**: Model exchange using the Robert Bosch Standard Simulation Library.

### 4.2 Single Source Approach with the VHDL-AMS Language Unified Modelling

The creation and maintenance of all the standard elements of the RB Standard Simulation Library is a complex and tedious task. The variety of implementations in all the simulation tools can be reduced using the standardised modelling language VHDL-AMS [17].

More than 30 years ago, the Simulation Software Committee of Simulation Councils, Inc. (SCi) developed a unified modelling language for system simulation called Continuous System Simulation Language (CSSL) [38]. ACSL [29] is a language following these recommendations.

CSSL focuses on system simulation alone and does not address the needs of physical systems (e.g. electronics with conservative communication and highly structured components). With the IEEE standardisation 1076.1 (VHDL-AMS) in March 1999, this deficiency no longer exists. Furthermore, with the mathematically well-founded semantics of all VHDL-AMS concepts, the language is much more suitable for unified modelling [17].
VHDL-AMS is a superset of the IEEE standard 1076 (VHDL) and therefore all issues as far as a regular programming language are concerned (e.g. expressions, functions) are already precisely defined. Furthermore, it provides tools for structuring systems and communication. The power for describing complex discrete time systems might be the only drawback, since a VHDL-AMS compliant simulator has to support full VHDL. Special purpose simulators (e.g. for hydraulics or real time) may only support some language subsets as described below.

The following section contains a short introduction to some aspects of VHDL-AMS that are important for control domain modelling. Most of the information about VHDL-AMS can be found in [17]. Its discrete modelling capabilities (from conventional VHDL) are already well documented (e.g. [26]).

- Structural decomposition and communication: As heritage of VHDL any model might be structurally divided into smaller units (blocks, components). Each component is defined with an interface `entity` and one or more `architectures`. For the information flow between these components VHDL-AMS provides two different mechanisms:
  - Signal flow (non-conservative): Data is transmitted in one direction from one unit to another (block diagram approach).
  - Energy flow (conservative): Two units are physically connected and communicate energy between one another (bi-directionally). The physical connection might be a node communicating pressure and flow.

- Variable objects: `quantities` are objects within VHDL-AMS whose values are determined according to a system of differential equations (see below). The derivative and integral of a `quantity` are again a `quantity` (`q'dot`, `q'integ`). A `signal` is the discrete state variable the value of which changes only at discrete points in time and is determined within a `process` (see below). If its values are changed, an event occurs, which may activate some suspended process.

- Differential algebraic equations: The basic concepts for describing analogue behaviour in VHDL-AMS are differential and algebraic equations (DAE). They can be written as simultaneous equations. There is a simple form e.g. `q'dot == f(q)` and conditional forms e.g. `IF condition USE x'dot == v ELSE x'dot == 0 END USE`.

- A `process` is coded as a list of sequential statements, which are executed within a closed loop (forever), and represents a finite state machine. It can be activated by some events (on a `process`) making it perform a transition and is suspended until some condition is met (`wait` until condition).

- Discontinuous behaviour: The interaction between the analogue kernel (solving the behaviour described by DAEs) and the digital kernel (solving the event driven behaviour) permits the descriptions of discontinuous behaviour similar to the powerful means of ACSL. An analogue event (e.g. detecting a threshold crossing of some `quantity`) raises a discrete event and may activate a `process`. The process might force the analogue kernel to continue its calculation with reinitialised values of some `quantities`.

- For real-time simulation, an appropriate subset of VHDL-AMS can be defined with necessary descriptive power and reduction of many time consuming mechanisms. This subset supports only signal-flow communication, ordinary differential equations and basic event mechanism for signal assignment, `process`, `wait`, and `break` statements.

In Figure 9, the VHDL-AMS model for the throttle plate in section 1.1. is given as an example. For the sake of simplicity, the coefficients in the model are assumed to be constant.

Based on a reference formulation of a standard element, directly executable code for a particular simulation tool can be generated. For this, after parsing the VHDL-AMS code, the architecture is examined with respect to its concurrent and simultaneous statements. The system of equations is checked to verify that it fulfils the given restrictions.

Since the inputs and outputs of a component are explicitly known (only unidirectional communication) and because of the restrictions on the equations, the algebraic system equations...
can be easily sorted according to the order mentioned above. This simple symbolic calculation eliminates the need for a numeric DAE solver.

The code generation already works for some simulators including the hydraulic simulator AMESim [3] and the control simulator MATLAB/Simulink [12]. The resulting code is used to prove the feasibility of the approach without commercial interests.

Fig. 9: VHDL-AMS implementation of the model (1) of the throttle plate in section 2.1.

4.3 Model Exchange with Respect to the Hydraulic Unit

In this section, the use of the RB Standard Simulation Library will be discussed using the example of the hydraulic modulator of the ESP system.

The application of an ESP system for a particular car is usually done using simulation models of all the relevant components (hydraulic modulator, car, controller) at the system level as shown in Figure 10. The supplier of the ESP hydraulics and controller uses detailed simulation models in order to design the hydraulic actuator and controller. Today, the tool chain between the detailed component models provided by the supplier and the simple models used by the car manufacturer is by no means perfect. Moreover, customer and client do not necessarily use the same simulation tool. Suppliers and car manufacturers can overcome the problem of model exchange by using the RB Standard Simulation Library together with one of the commercial tools such as ASCET, AMESim, Simulink or SABER or any VHDL-AMS compatible tool.

During the early stages of the design phase, the components of the hydraulic modulator are implemented and connected in the hydraulic simulation tool. This gives a simple model of the overall hydraulic system that allows us to carry out primary simulation runs. The same standard elements are used in the simulation tools for the electronics and the controller design, in order to reproduce the simple model of the hydraulic modulator. All parties involved in the design have the same model of the overall system with the same behaviour and an identical set of parameters, which serves as a basis for further development.
More detailed investigations into the hydraulic or electronic components using tool and domain-specific elements can therefore be based on a common model of the entire system. If the hydraulic valve is changed, both the complex model and the corresponding simple model have to be modified. The parameters of the simple model can be obtained from the behaviour of the complex one using parameter fitting techniques. The domain-specific elements are only needed in the domain-specific simulation tool and it is easy to keep parameterisation consistent.

The RB Standard Simulations Library supports this type of model exchange and provides a model of the entire system that can be updated automatically during the entire design process. Specific tools used by the customer can easily be integrated due to the use of VHDL-AMS and the automatic code conversion.

This allows a consistent flow of information that is currently still far from well established in the design process. In this respect, the picture in Figure 10 shows a somewhat idealised perspective that has to be accomplished in the future.

CONCLUSIONS

Systems engineering is an important and efficient means to meet future requirements in the design of mechatronic systems. The most important challenges are:

- **Reduction of the time-to-market** while the complexity of the products and the interaction between their components (from the hydraulics, mechanics, electronics and control domains) is increasing.

- **Correctness of design**, i.e. minimising the risk of incorrect design decisions (function, costs, quality, manufacturability) using CAE-based validation of design proposals in the early phases of the design process.

In this respect, design methodology, design processes, CAE tool chain and the organisational structure have to be taken into account simultaneously. The use of systems engineering allows design alternatives and changes to be easily created, compared and verified with specifications throughout the entire design process.
References


[15] Robert Bosch GmbH. Fahrdynamikregelung ESP - Fahrsicherheitssysteme für Personen-

Munich/Germany, 2000.


[18] IMAGINE S.A., 5, rue Brison, F-42300 Roanne, France. AMESim - Advanced Modelling
Environment for Simulation, 1999.


systems using a VHDL-AMS based standard element library - a Common Rail example. In

simulation library - a pragmatic approach to model exchange and simplification. In Proc. 5th


Academic Publisher, 1989.


