INTERNATIONAL DESIGN CONFERENCE - DESIGN 2008
Dubrovnik - Croatia, May 19 - 22, 2008.

DESIGN 2008

SYSTEMATIC DEVELOPMENT OF CONTROLLERS BASED ON THE PRINCIPLE SOLUTION OF SELF-OPTIMIZING SYSTEMS

J. Gausemeier, S. Kahl, C. Low and B. Schulz

Keywords: design methodology, specification techniques, mechatronics, self-optimization, self-optimizing control

1. Self-Optimizing Systems

Nowadays, most mechanical engineering products rely on the close interaction of mechanics, electronics, control engineering and software engineering, which is aptly expressed by the term mechatronics. The aim of mechatronics is to optimize the behavior of a technical system. The conceivable development of information technology will enable mechatronic systems with inherent partial intelligence. These systems are called self-optimizing systems. In order to structure such self-optimizing systems, the hierarchy of complex mechatronic systems suggested by [Lückel et al., 2001] was adopted and extended to include the aspect of self-optimization, as shown in Figure 1.
The basis of this hierarchy is provided by mechatronic function modules (MFMs), consisting of a basic mechanical structure, sensors, actuators and a local information processing containing the controller. The combination of MFMs, coupled by information technology and/or mechanical elements, constitutes an autonomous mechatronic system (AMS). Such systems also possess a controller, which deals with higher level tasks such as monitoring, fault diagnostics and maintenance decisions as well as generating parameters for the local information processing of the individual MFMs. Similarly, a number of AMSs constitute what is called a networked mechatronic system (NMS), simply by coupling the associated AMSs via information processing. On each level the controllers can be enhanced by the functionality of self-optimization. Thus the system elements receive an inherent partial intelligence. The behavior of the overall system is characterized by the communication and cooperation between these intelligent system elements. From the information processing point of view they are considered as a distributed system of cooperative software agents.
The behavior of self-optimizing systems emerges from the self-optimization process, which is expressed as a series of three actions:

1. **Analysis of the current situation.** Here the self-optimizing system collects all relevant data about its actual state and its environment. Such observations may also be made by communicating with other systems indirectly.
2. **Determination of the system objectives.** This means, the self-optimizing system autonomously derives its current objectives by using the acquired data. The objectives form the basis for the adaptation of the system behavior.
3. **Adaptation of the system behavior** by adapting the parameters and/or the structure of the system if necessary.

From a given initial state, the self-optimization process proceeds, on the basis of specific influences, into a new state, i.e. the system undergoes a state transition.

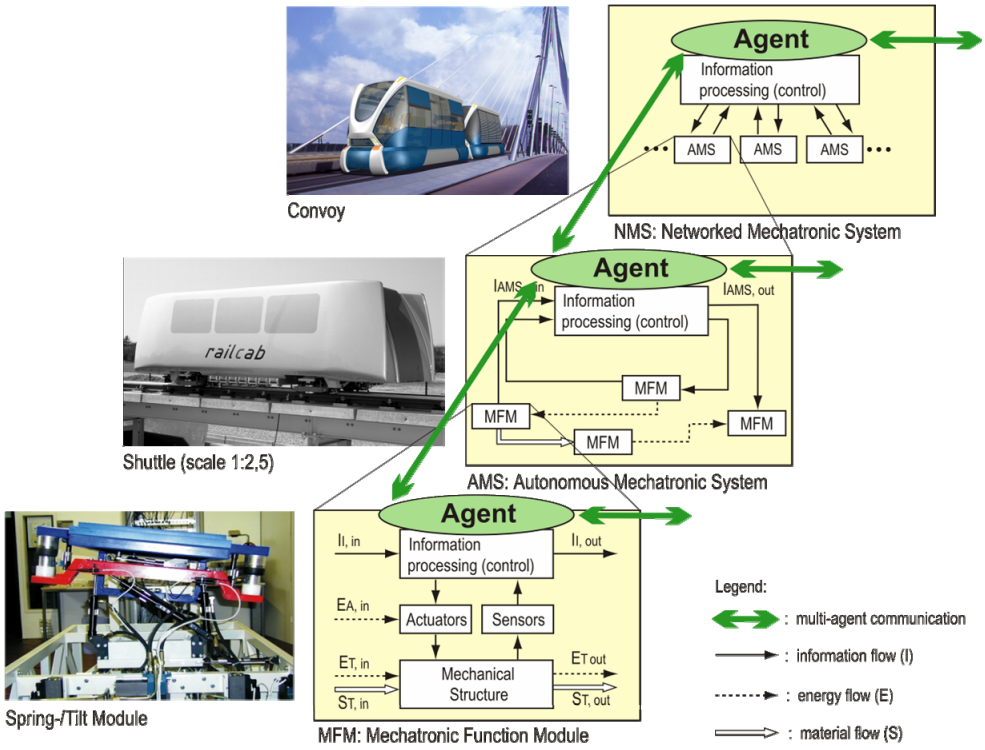

Figure 1. Structure of mechatronic systems [Lückel et al., 2001]

2. Development of Self-Optimizing Systems

The established design methodologies of technical systems, for instance [Pahl et al., 1996] and [VDI Guideline 2206, 2004], lay the foundation for the development of self-optimizing systems. Nevertheless these methodologies have to be fundamentally extended. On a generic level, the development of self-optimizing systems starts with the domain-spanning conceptual design, followed by the domain-specific concretization and ends with the system integration. The result of the conceptual design phase is the principle solution. It describes the main physical and logical operating characteristics of the system in a domain-spanning way. On the basis of this jointly developed principle solution, further concretization will take place separately in the domains involved. Finally, during the system integration phase, the outcomes from the individual domains are integrated to form an overall system.

Within the conceptual design phase, we use a set of semi-formal specification techniques to describe the principle solution of a self-optimizing system [Frank, 2006]. For a complete description several views on the self-optimizing system are needed. Each view is mapped by a computer onto a partial model. The principle solution is made up of the following views: requirements, environment, system of objectives, functions, active structure, shape, application scenarios and behavior. This last view is considered as a group because there are various types of behavior (e.g. the dynamic behavior of a multibody system, the cooperative behavior of system components etc.). The relationships between the partial models are also modeled, which leads to a system of coherent partial models, as shown in Figure 2.

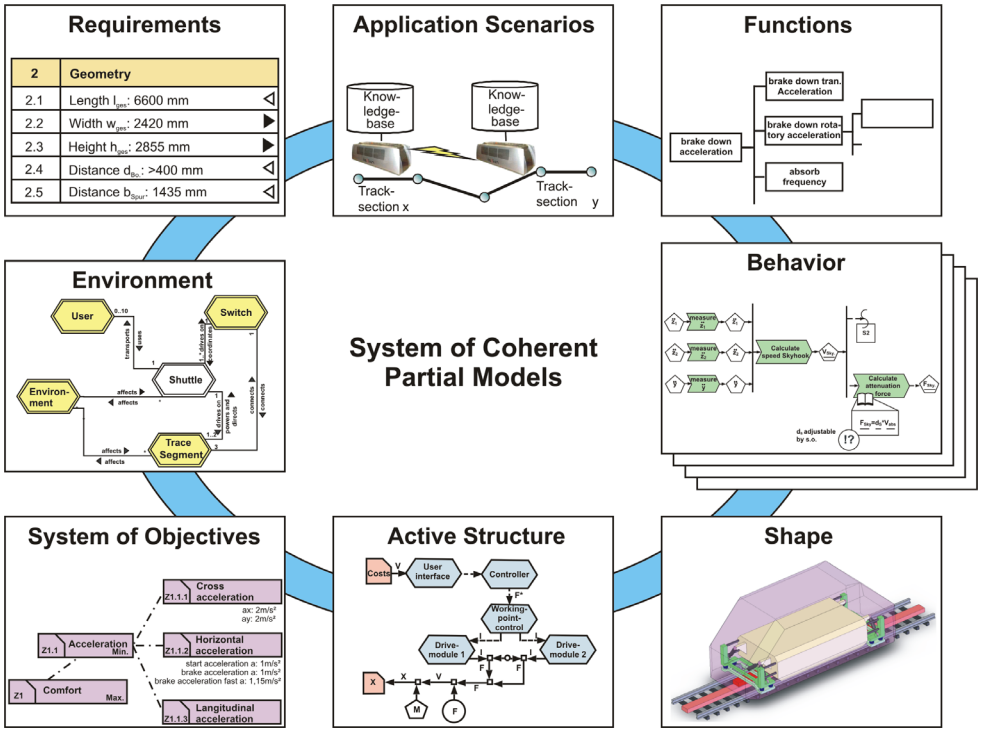

Figure 2. Partial models describing the principle solution of self-optimizing systems

This principle solution forms the basis for the subsequent domain-specific concretization. It is indispensable to transfer all design concepts formulated in the principle solution for the deployment of different domain-specific designs without any information loss. At this point, clear design goals have to be understood by the specialists and sufficient system information must be available as prerequisites before design concretization could be continued [Böcker et al., 2006]. Having the right information in hand, the specialists can concretize the design using their domain-specific methods and techniques. For instance, mechanical engineers design the optimum geometry; software engineers analyze the state transitions; control engineers select the controlled system variables, etc. The development of systematic approaches from the domain-spanning principle solution towards domain-specific design concretization is necessary to ensure a seamless development flow, but remains hitherto a challenge to the design community.

3. From the Domain-Spanning Principle Solution towards Domain-Specific Controller Design

A systematic approach has been developed to show control engineers how to identify the control concepts in the principle solution, extract them from the principle solution, and subsequently transform them into the specification of control engineering. The systematic approach is illustrated in Figure 3. The activities and the outcomes of each phase are described in more details in the following sub-sections. Besides allowing control engineers to integrate various control techniques at an abstraction level avoiding the design details of the self-optimization process, the systematic approach bridges the gap of the development flow between the top-down approach used in specifying the principle solution and the bottom-up approach used for controller design.

We clarify the points mentioned by the design of controllers needed by a self-optimizing motor drive. The focus here is the control of the angular dynamics, which is elementary for a large number of applications, e.g. machine tools and general drives for automation purposes. The control structures of the motor drive can be reconfigured. Different control structures demand different amount of resources and deliver different performances. The functionality of the controllers can be totally (or partly) implemented on a CPU[1] or a FPGA[2]. The controllers run alongside the other applications on the same computation platform. They are competing for the available resources, such as memory, CPU time, and surface area on a FPGA. In this application, resources are allocated at run time. The run time switching between the different implementations of the controllers is done by means of partial run-time reconfiguration [Schulz et al., 2007].

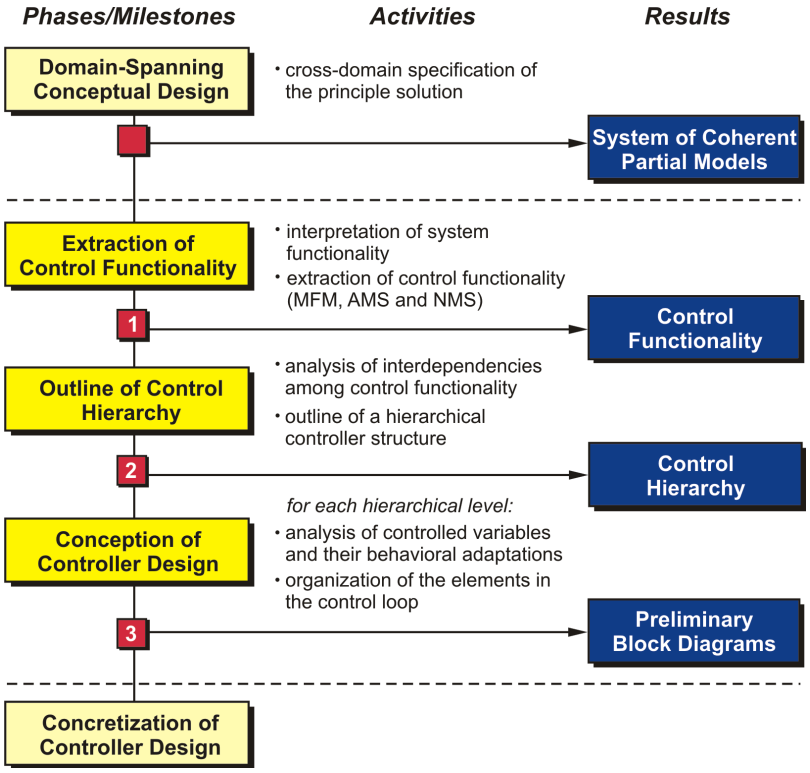

Figure 3. A procedural model from the domain-spanning principle solution towards the domain-specific controller design

3.1 The Principle Solution as exemplified by the Self-Optimizing Motor Drive

Using the semi-formal specification technique, the principle solution of a self-optimizing motor drive is formulated. The principle solution of the self-optimizing motor drive presents a generalized solution concept which could be adapted into different applications, such as hydraulic pumps or as electrical part in the drive train of a hybrid car. Figure 4 shows a cut-out from the partial model behavior – state, behavior – activity, and active structure. The interconnections between these partial models are also indicated. Each of the partial models is described in detail below.

[1] CPU: Central Processing Unit

[2] FPGA: Field-Programmable Gate Array

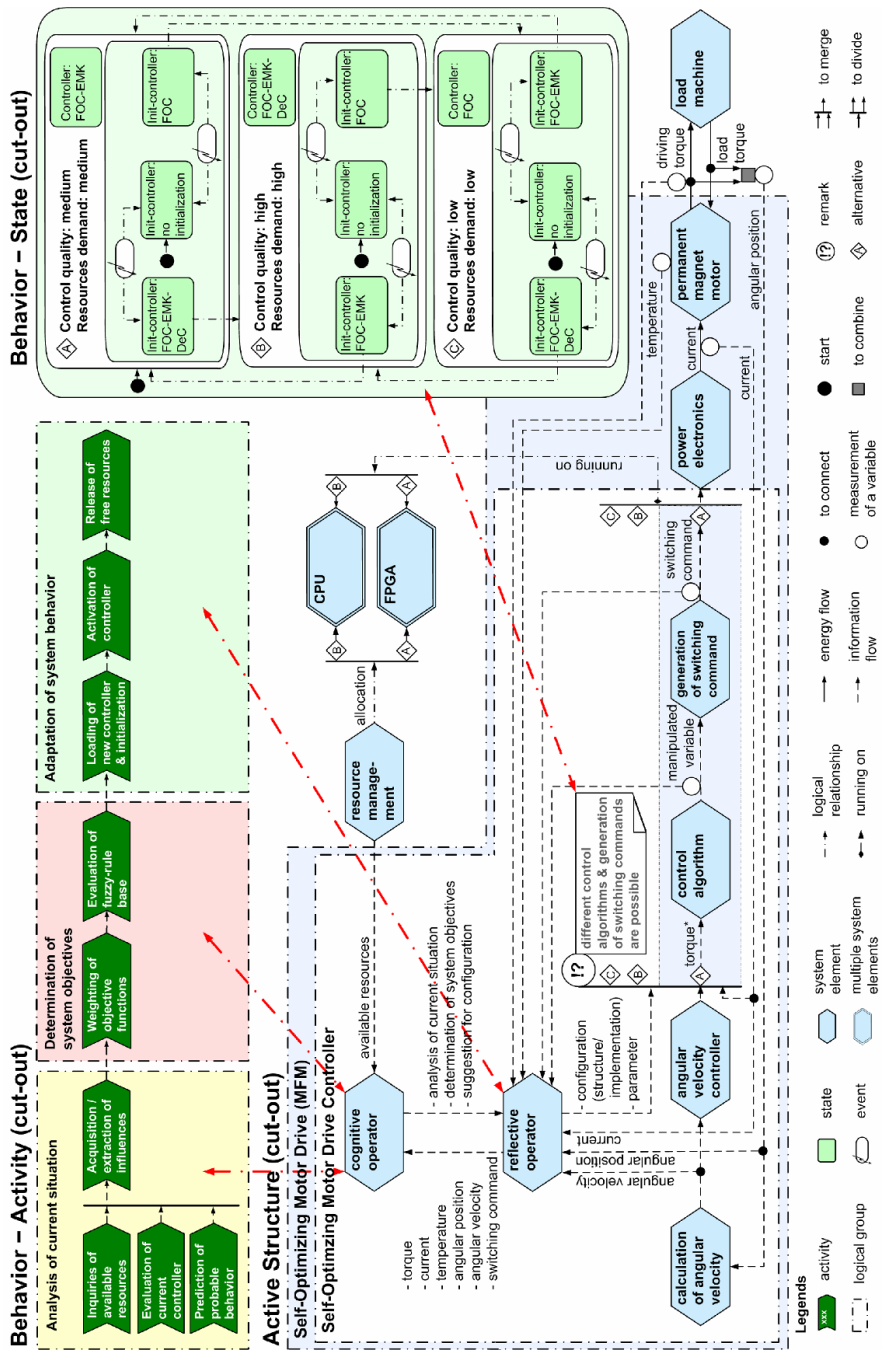

Figure 4. Cut-out from the principle solution of a self-optimizing motor drive

Active Structure: The active structure depicts the system elements that represent solution patterns and active principles, together with their characteristics as well as the relations between these system elements. Relations here refer to the flows of material, information, and energy, as well as their logical relations. Figure 4 shows an exemplary section of the active structure of the self-optimizing motor drive, which is a mechatronic function module (MFM). The motor drive consists of a self-optimizing motor drive controller, power electronics, and a permanent magnet synchronous motor. Besides that, the motor drive is equipped with a resource management module, a CPU, a FPGA module, and a load machine. The self-optimizing motor drive controller is built based on the concept of the Operator Controller Module (OCM), which is divided into three levels, i.e. cognitive operator, reflective operator, and controller [Oberschelp et al., 2004]. The controllers directly control the angular dynamics of the motor drive and have to fulfill hard real-time requirements. The controller structures can be reconfigured in response to the state transitions and the underlying adaptive processes. The switching between the controllers is specified by the alternative controller structures labeled with A, B, and C in the active structure. On top of the controllers, the reflective operator is responsible for the switching of the controller structures. It is event oriented and operates in hard real-time. At the highest level, the cognitive operator uses a preemptive optimization to improve the behavior of the motor drive. It does not interact in real-time with the motor drive.

Behavior – Activity: The partial model behavior – activity describes logical sequences of system activities which includes all operation and adaptation processes. Figure 4 shows a cut-out of the partial model behavior – activity of the self-optimizing motor drive which describes the self-optimization process. The cognitive operator analyzes the current situation and subsequently determines the currently active objective function. The current objective function is weighted on the basis of the available resources, the evaluation of the currently active controller, the prediction of probable system behavior, as well as the influences from the environment and within the system itself. The weighting of the objective functions is carried out by means of a fuzzy-rule base. As the environment changes, the rules to select a controller can also be changed. If necessary, a new controller will be loaded, initialized, and subsequently activated. Finally, the unused resources will be released. These activities of loading, initialization and activation of controllers until the release of unused resources are executed by the reflective operator.

Behavior – State: The partial model behavior – state describes the envisaged system states, the state transitions, as well as the events that trigger a state transition. In our application example, the operating conditions of the motor drive include the constant load operation, the acceleration operation and the dynamic load operation. With the help of a load machine, these different operating conditions are simulated, i.e. by setting the load torque or the driving velocity, in each case with temporal changes. Such operating conditions require the motor drive to operate in three different states, each of them with low, medium, or high control quality and resources demand. Different controller structures are required in different states. Considering both resources demand and control quality, the most viable system state is selected depending on the environmental influences, desired performance and available resources. Both the initialization of the new controllers and their switching take place, as the system transits from an initial state into a new state, on the encounter of specific triggering events. However, frequent switching and/or toggling on and off between the controllers have to be avoided.

3.2 Extraction of Control Functionality

Having formulated the principle solution, the functionality to be implemented on the controllers has to be extracted. For this purpose, control engineers will first need to interpret the system functionality specified within the partial models. The understanding of the system functionality is used to guide the extraction of the control functionality at each hierarchical level of the system. Depending on the complexity of the product, the functionality of the controllers is accordingly mapped onto the hierarchical level of MFM, AMS or NMS, as per the mechatronic structure shown in Figure 1. As self-optimizing systems usually require multiple control algorithms, the precise insight into their control functionality will help to enable the decomposition and the integration of control solutions in the later stages. The partial model functions provides a good reference for the extraction of the control functions.

In our application example, the system functionality of the motor drive is the ability of maximizing the control quality and yet minimizing the resources demand while the system is operating under changing environmental influences. The control functionality is the control of the angular dynamics, which is achieved through the adjustment of the angular velocity by controlling the torque of the motor drive. The highlighted function blocks at the upper right of Figure 5 are the extracted control functionality from the partial model functions. The linkages between the control functions and their associated system elements in the active structure are also indicated.

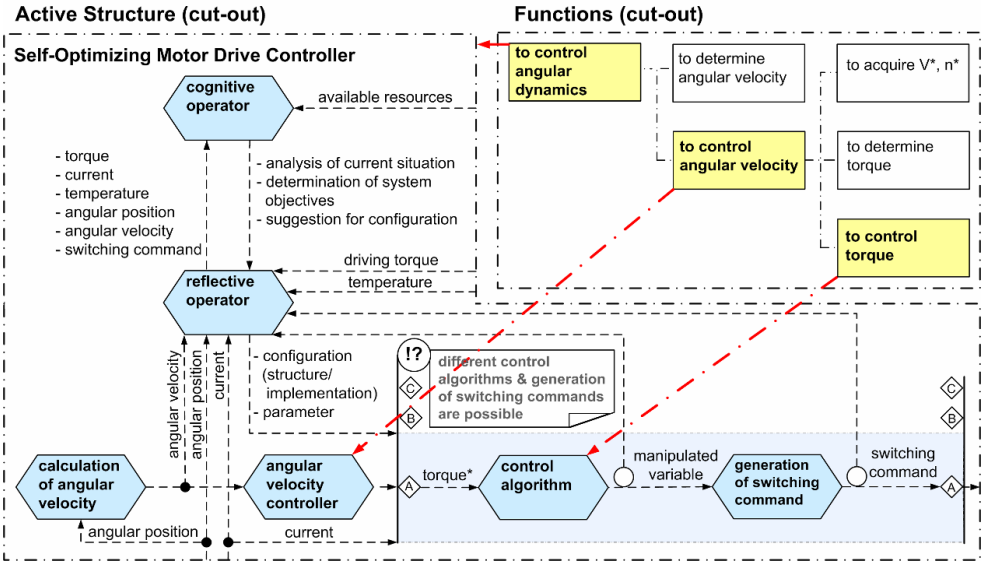

Figure 5. Extraction of control functionality from the principle solution

3.3 Outline of Control Hierarchy

The outline of a control hierarchy can be done in two successive steps. The first step is the analysis of the interdependencies among the control functionality. The main interdependencies can be identified from the linkages of the extracted control functions with their associated system elements in the active structure. The types of interdependency partly determine how the control algorithm should be derived. Therefore, understanding of the interdependency is important here for effective coordination among the control algorithms. The second step involves the outline of a control hierarchy based on the identified interdependencies. In order to realize the main control function, different controlled system variables can be involved, depending on the physical characteristics of the plant and the selected sensors and actuators. This may involves further decomposition of the control hierarchy, if necessary. Each of the control functionality in the hierarchy refers to a particular task to be carried out by a control algorithm in a cascaded control structure. Therefore, the hierarchical levels in the control hierarchy resemble the cascaded controller structure to be developed. The control functionality which has to be delivered in different degree of quality is marked as duplicated function blocks.

As an example, the control hierarchy of the self-optimizing motor drive is outlined. Referring back to the active structure in Figure 5, it is to be noted that the ‘control algorithm’ has ‘to control the torque’ though it has two information flows of ‘current’ and ‘torque’ as the inputs. On one hand, the main dependency between the ‘angular velocity controller’ and the ‘control algorithm’ is the adjustment of the driving torque. On the other hand, the technique used to control the torque has to be made clear to be able to outline the control hierarchy. It is to be understood that the adjustment of torque, is a result of current control, which directly influences the angular velocity of the motor drive. Therefore, in

order to control the torque of the motor drive, the current of its permanent magnet synchronous motor has to be controlled.

Adopting such a strategy, we build up a control hierarchy as shown in Figure 6. The function 'to control the angular dynamics' is first decomposed into the function 'to control the angular velocity', then decomposed into the function 'to control the torque', and further decomposed into the function 'to control the current' of the motor drive. The dependency between the control functions are marked: 'adjustment of driving torque', 'adjustment of current', and 'adjustment of control signal'. The controlled variables, which serve as the reference input and actual output at each hierarchical level, are also indicated in the control hierarchy. On top of that, the current controllers are represented as multiple function blocks. Implementation wise, this implies the switching between the controller structures which realize the functionality of current control with different degrees of quality and resources demand. This information can be extracted from the alternative control algorithms A, B, and C, as indicated in the active structure.

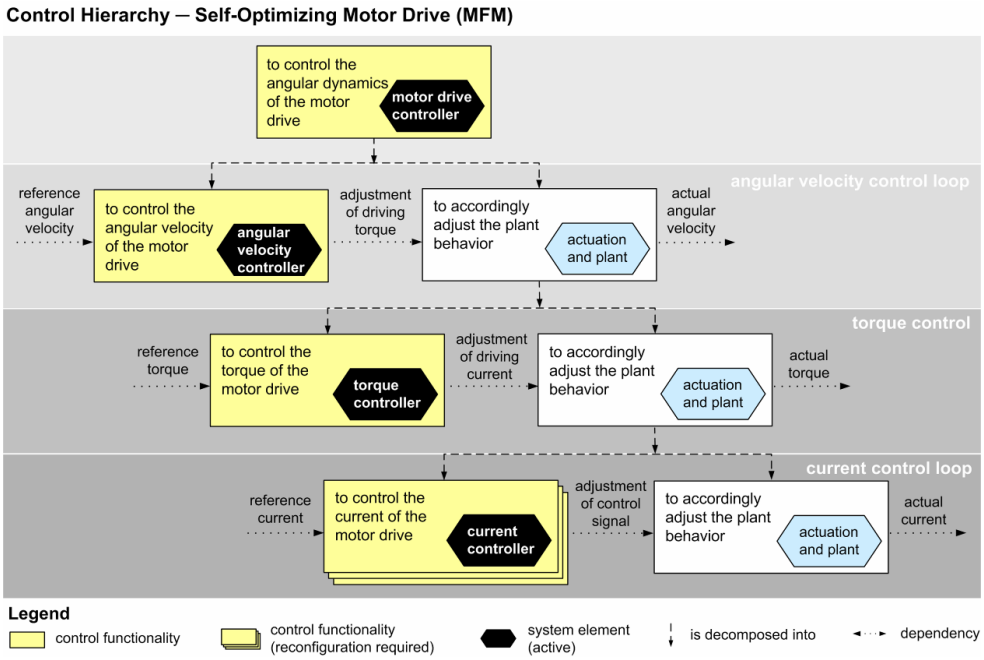

Figure 6. Outline of the control hierarchy

3.4 Conception of Controller Design

Generally, the favorable or proven controller structures for a standard application are usually well-known. A viable control solution that should work accordingly can be predetermined based on the information extracted from the principle solution and the experience of the engineers. The conception of controller design involves the analysis of controlled variables and their behavior adaptation. That means, the necessary feedback loops are identified from the information flows and the required switching of the controllers at each hierarchical level is known. Subsequently, the initial controller structures are outlined. The blocks and the feedback loops can be drawn with reference to the active structure. Control-specific information that can be extracted from the active structure includes, for instance, comparison elements, correction elements and measurement elements. These elements are organized together within the controller loops.

Within this phase, control engineers have to decide if an open-loop control will be sufficient, or a closed-loop control is necessary. In addition, a feedforward control can be used to eliminate undesirable effects of disturbances on the system output. A proper combination of open-loop and closed-loop controls is usually less expensive and will give satisfactory overall system performance. Besides the active structure, the behavior ─ states provide information about the adaptation of the controller structure under different application scenarios. The availability of different systems states as well as the triggering events of the state transitions can now be understood. The outcomes of this phase are the conceptual controller layout in the form of preliminary block diagrams. The preliminary block diagrams are used as the preliminary control system descriptions, which were conventionally represented in the form of system equations. By applying formal design techniques, the preliminary block diagrams can be further concretized to implement each of the control functionality.
Figure 7 exemplifies the preliminary block diagrams for the control of angular dynamics of the self-optimizing motor drive. The block representing the electrical plant will contain power electronic and equations describing the electrical part of the motor that covers the voltage-current behavior. The blocks representing the mechanical plant includes the dynamics of the motor and the generation of driving thrust. The blocks representing the controllers will be added with control algorithms. In this example, while the torque is controlled by an open-loop controller, the PI-controller is used for both the angular velocity control and the current control. The current control involves the control of a vector of current components: the *d*-current and the *q*-current. On one hand, the *d*-current has to be controlled to zero in order to avoid energy losses in the motor. On the other hand, the *q*-current has to be controlled for the adjustment of the torque driving the motor. The focus here is the switching between the current controllers when the motor drive transits from a particular state into another. As shown in Figure 7, the concept of using different control structures and the generation of switching command are clearly indicated within the controller loops.
All controller structures used to control the current are based on a Field Oriented Control (FOC) scheme. The properties of these controllers are well known by control engineers and have been presented by several authors, for instance [Blaschke, 1972]. The difference between the controller structures A, B and C lie in the internal structure of the current control block. In control structure A, the current control block contains a FOC structure where the output of the PI-controller is directly the output of the block. Therefore, the dynamics of the control loop is determined by the PI-controller. Besides the PI-controller, the current control block in controller structure B contains a feed-forward for Back-EMF compensation. As such, the dynamics of the control loop is improved as compared with using the FOC alone. In controller structure C, both compensation for the Back-EMF and the decoupling of the current are incorporated in the feed-forward for the FOC. Switching between these controller structures leads to the behavioral adjustment of the motor drive in delivering the self-optimizing capability.

3.5 Concretization of Controller Design

Naturally, not all design considerations can be made during the conceptual design phase. Detailed design of the controllers is done using the de facto tools that supports the modeling, analysis, and synthesis of the feedback controllers. There are various established controller design techniques. On one hand, systematic approaches for parameter tuning are well known, e.g. the Ziegler-Nichols method. These approaches can be called heuristic approaches. On the other hand, there are also analytical approaches. The analytical control design algorithms can be structured in two ways, i.e. the formula-based algorithms (e.g. coefficient comparison for time constants and cross-ratio) and the graphic-based algorithms (e.g. pole-placement). [Schröder, 2001] [Umland, 1988]
In order to ensure a structured controller design process and to prevent design errors, the controllers are simulated together with the plant. This is required to predict the behavior of the non-linear part of the system and the uncertainties, which can be examined by simulation. Such simulations can be understood as the premise for the implementation of the controllers on their target-platform. Depend-

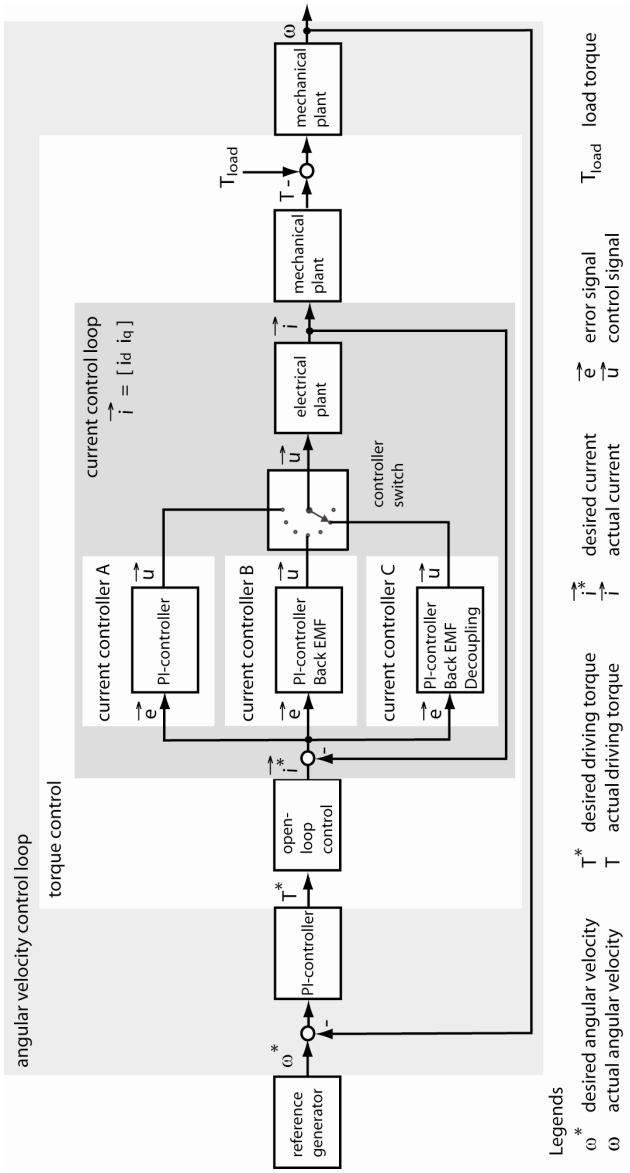

Figure 7. Preliminary block diagrams for the self-optimizing motor drive

ing on their applications, an analog (e.g. with op-amps) or a digital (with microcontrollers) target-platform may be used.

The laboratory prototype of the self-optimizing motor drive has been developed in the Institute of Power Electronics and Electrical Drives at the University of Paderborn. The parameters of the controllers are tuned using the cross-ratio-approach. The target platform is a rapid prototyping system with a FPGA-based and a CPU-based controller prototyping. In this context, the controllers and their switching concepts are validated by the Hardware-in-the-Loop-Tests (HiL-Test). The hardware includes the power electronics, the permanent magnet motor drive and an induction motor as the load

machine. The load machine emulates the impacts of changing load on the motor drive. Test in a real plant application, like hydraulic pump or machine tool, is the final step of the design concretization of the controllers.

Figure 8 shows the MATLAB-based implementation of the *q*-current controller with Back-EMF compensation and decoupling of the current, which was indicated as current controller C in Figure 7.

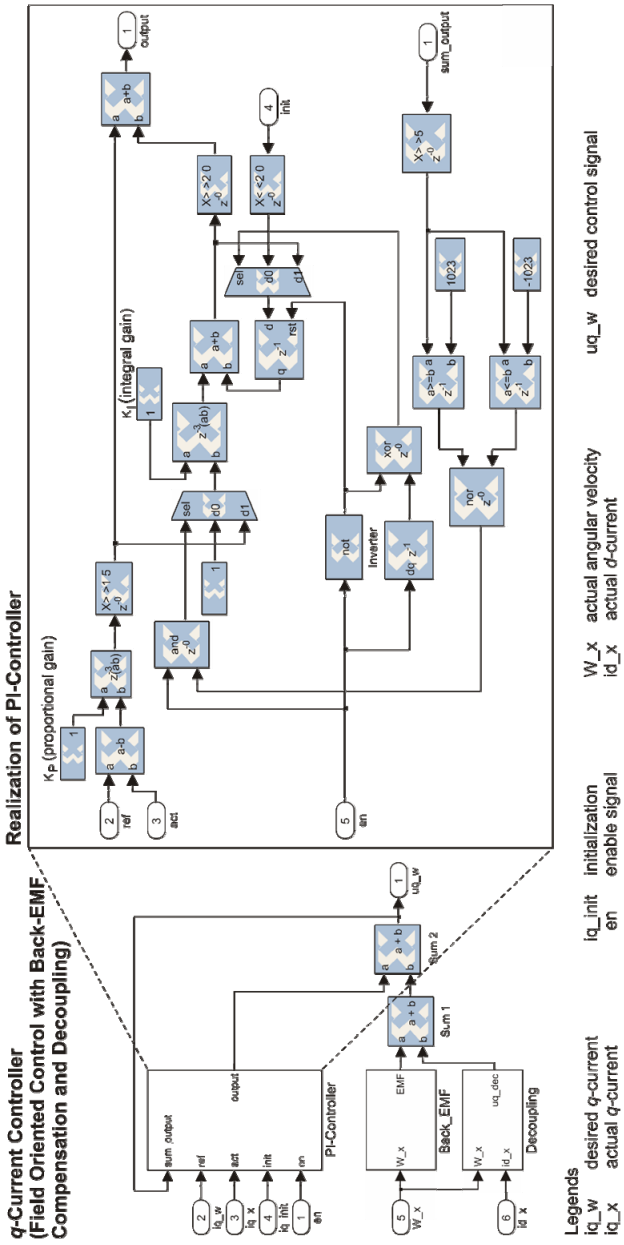

Figure 8. MATLAB-based implementation of the *q*-current controller (Type C)

The internal of the block 'PI-controller' is shown at the right of Figure 8. The internal parameters of the PI-controller, i.e. the proportional gain and the integral gain are labeled. The current controllers are implemented using MATLAB Simulink and Xilinx System Generator, which is directly the source for the FPGA-platform. For the current controller B (FOC with Back-EMF compensation), the adder 'Sum 1' and the block 'id_decoupling' are not implemented. For the current controller A (FOC without feedforward), only the block 'PI-controller' is implemented. Depending on the operational conditions of the motor drive, the feed forward parts of the current controller may not be required. If a feed forward for the Back-EMF and/or decoupling is dispensable, their associated reconfigurable resources on the FPGA can be released for other applications.

The current controller A, B, and C as well as the switching between the controllers are implemented and tested under different conditions. There are three operating conditions as specified in Section 3.1: constant load operation (constant motor speed), acceleration operation (speed up) and dynamic load operation (speed is varied by means of a load machine). The left column of Figure 9 compares the measurement results of the current controller A, B, and C under the acceleration operation. The reference value of the *q*-current are step inputs. The experimental results obtained were as expected. On the right column of figure 9, the switching from current controller A to current controller C is shown. The time of switching is marked by the switching signal in the first graph. It is shown that the switching does not have a negative influence on the control loop.

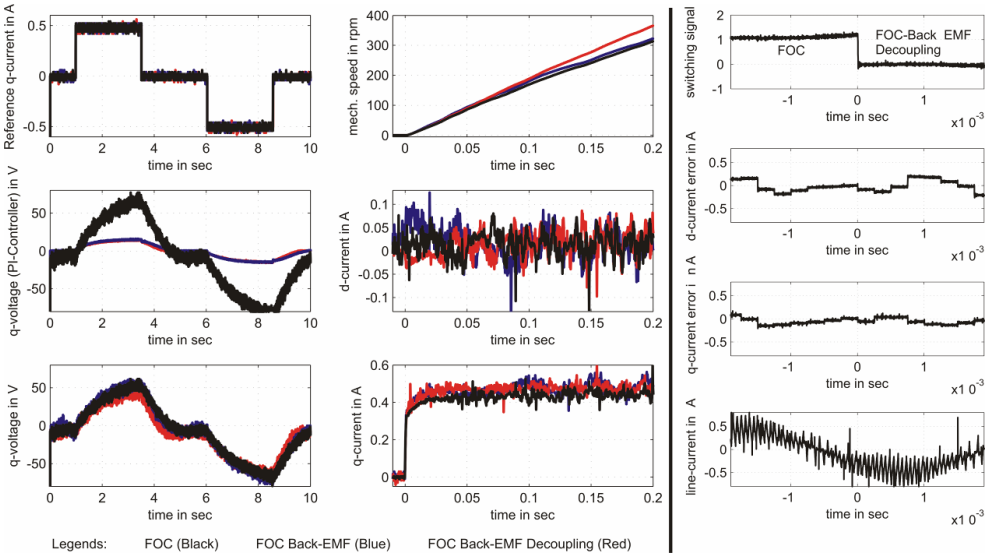

Figure 9. Measurement results of the current controller A, B, and C (left) and switching from current controller A to current controller C (right)

4. Conclusion

The method described here provides a practical guidance for the development of controllers for self-optimizing systems. From the domain-spanning principle solution towards the domain-specific controller design, the method delivers three concerns that we think are significant, i.e. easy handling of the design concepts, design at a high abstraction level from the beginning, and the re-use of the design concepts taken from the principle solution within the early design concretization phase. Using the holistic principle solution as the starting point for systematic development of controllers significantly reduce the late emerging design incompatibilities among various sections of the self-optimizing system, and consequently prevent major delays as well as cost overruns.

Acknowledgement

This contribution was developed in the course of the Collaborative Research Centre 614 "Self-Optimizing Concepts and Structures in Mechanical Engineering" (Speaker: Prof. J. Gausemeier) funded by the German Research Foundation (DFG) under grant number SFB 614.

References

Blaschke, F.: "The Principle of Field Orientation as Applied to the New Transvektor Closed-Loop Control System for Rotating-Field Machines", Siemens Review, Vol. 34, May 1972, pp. 217–220.

Böcker, J.; Schulz, B.; Knoke, T.; Fröhleke, N.: "Self-Optimization as a Framework for Advanced Control Systems", IEEE Industrial Electronics, IECON 2006 - 32nd Annual Conference of the IEEE Industrial Electronics Society, pp. 4671-4675, 2006.

Frank, U.: „Spezifikationstechnik zur Beschreibung der Prinziplösung Selbstoptimierender Systeme", Dissertation, Universität Paderborn, HNI-Verlagsschriften-Reihe Band 175, Paderborn, 2006.

Lückel, J.; Hestermeyer, T.; Liu-Henke, X.: "Generalization of the Cascade Principle in View af a Structured Form of Mechatronic Systems", IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2001), Como, Italy, 2001.

Oberschelp, O.; Hestermeyer, T.; Giese, H.: „Strukturierte Informationsverarbeitung für selbstoptimierende mechatronische Systeme". In: Gausemeier, J.; Wallaschek, J. (Hrsg.): 2. Paderborner Workshop Intelligente Mechatronische Systeme, 25.-26. März 2004, HNI-Verlagsschriftenreihe, Band 145, Paderborn, 2004.

Pahl, G.; Beitz, W.: "Engineering Design - A Systematic Approach", Second Edition, Springer Verlag, Berlin, Heidelberg, New York, 1996.

Schröder, D.: „Elektrische Antriebe - Regelung von Antriebssystemen", Springer Verlag Berlin Heidelberg New York, 2001.

Schulz, B.; Paiz, C.; Hagemeyer, J.; Mathapati, S.; Porrmann, M.; Böcker, J.: "Run-Time Reconfiguration of FPGA-Based Drive Controllers", 12th European Conference on Power Electronics and Applications, Aalborg, Denmark, 2007.

Umland, J.W.; Safiuddin, M.: "Magnitude and Symmetric Optimum Criterion for the Design of Linear Control Systems ─ What is it and how does it compare with the others?", Industry Applications Society Annual Meeting, Vol.2, pp. 1796-1802, Pittsburgh, PA, USA, 1988.

Verein Deutscher Ingenieure (VDI): "Design Methodology for Mechatronic Systems", VDI Guideline 2206, Beuth Verlag, Berlin, 2004.

Prof. Dr.-Ing. Jürgen Gausemeier
Heinz Nixdorf Institute, University of Paderborn
Fürstenallee 11, D-33102 Paderborn, Germany
Tel.: +49-5251-606267
Fax.: +49-5251-606268
Email: juergen.gausemeier@hni.upb.de