From Domain-Spanning Conceptual Design to Domain-Specific Controller Design of Self-Optimizing Systems

Prof. Dr.-Ing. Jürgen Gausemeier
Dr.-Ing. Ursula Frank
M. Sc. Cheng Yee Low
Heinz Nixdorf Institute, University of Paderborn
Fuerstenallee 11, 33102 Paderborn, Germany
Tel.: +49-5251-606267, Fax: +49-5251-606268
E-mail: {juergen.gausemeier@hni.uni-paderborn.de}

Dipl.-Ing. Christian Henke
Institute of Power Electronics and Electrical Drives, University of Paderborn
Warburger Str. 100, 33098 Paderborn, Germany
Tel.: +49-5251-605482, Fax: +49-5251-605483
E-Mail: {henke@lea.upb.de}

Summary:
Self-optimizing systems are technical systems with the ability to learn, communicate with each other, and optimize their behaviour autonomously in response to environmental changes. The operation of such systems requires close interaction between mechanics, electronics, control engineering and software engineering. This paper addresses the conceptual design of controllers based on the principle solution of self-optimizing systems. Controllers required by self-optimizing systems consist of multiple control algorithms. The control algorithms are designed to implement the control functionalities formulated during the conceptual design phase. The development of controllers based on the principle solution requires engineering skills not covered by control theory. These skills are extracting the control functionalities from the principle solution, decomposing the overall control problem and organizing the individual solutions to constitute a complete solution. A procedural model that integrates these skills is thus presented to link the domain-spanning principle solution phase with the domain-specific controller design phase.

Keywords:
mechatronics, development methodology, principle solution, specification technique, control.

1 Introduction

Self-optimizing systems are developed in interdisciplinary teams. The development process starts with domain-spanning conceptual design and is followed by domain-specific concretization. During the phase of conceptual design, fundamental decisions concerning the structure and the operation of the system are made by specialists from different
domains. The result of the conceptual design phase is the principle solution. The cross-domain specification of the principle solution lays the foundation for effective communication and cooperation between the specialists who are engaged in developing the self-optimizing system. With the information extracted from the principle solution, specialists will concretize their design using domain-specific methods and techniques. For instance, mechanical engineers develop the optimum geometry, software engineers analyze the state transitions, while control engineers are responsible for the controlled behaviour of selected system variables.

This paper focuses on the transition from the principle solution phase to the controller design phase. System developers normally conceptualize the principle solution of self-optimizing systems following a top-down approach. On the contrary, control engineers prefer to design the required controllers following a bottom-up approach. A method is therefore needed to avoid any development gap between domain-crossing conceptual design and domain-specific controller design.

This paper is organized as follows. Section 2 describes the concepts and structures of self-optimizing systems. In Section 3, various specification approaches in the early development phases of technical systems are described. The cross-domain specification for the principle solution of self-optimizing systems is explained in Section 4. Section 5 describes the procedural model for controller design based on the principle solution. The procedural model is then exemplified by the development of convoy strategy for autonomous shuttles under research initiative “Neue Bahntechnik Paderborn / RailCab”.

2 Self-Optimizing Concepts and Structures

Nowadays, most mechanical engineering products are relying on the close interaction of mechanics, electronics, control engineering and software engineering which is aptly expressed by the term mechatronics. The ambition of mechatronics is to optimize the behaviour of a technical system. Sensors collect information about the environment and the system itself. The system utilizes this information to derive optimal reactions. Mechanical engineering systems in the near future will consist of configurations of system elements with inherent partial intelligence. The behaviour of the overall system is characterized by the communication and cooperation between these intelligent system elements. From the viewpoint of information technology, these distributed systems are considered as cooperative agents.

Progressing from mechatronics to self-optimization, the hierarchical structuring of complex mechatronic systems suggested by Joachim Lückel is extended to include the aspect of self-optimization [LHL01]. This hierarchy is illustrated in Figure 1. The basis of this is provided by “mechatronic function modules” (MFMs), consisting of a basic mechanical structure, sensors, actuators and a local information processor 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 systems 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. The controller of a NMS carries out higher-level functions in the same way as that of the AMS. In the context of railway technology, a
spring and tilt module would be an MFM, a shuttle would be an AMS, and a convoy would be a NMS. On each level the controller is enhanced by the functionality of self-optimization. Thus the previously mentioned system elements (MFM, AMS, NMS) receive an inherent partial intelligence.

The key aspects and the mode of operation of a self-optimizing system are illustrated in Figure 2. The self-optimizing system detects factors that influence the system. The factors may originate from its surroundings (environment, users, etc.) or from the system itself. The self-optimizing system determines its currently active objectives on the basis of the encountered influences. Objectives formulate the behaviour that is required by the system, desired, or to be avoided [FGK+04].
Figure 2: Aspects of self-optimizing systems in terms of objectives, behaviour, structure, and parameters

Self-optimizing systems are able to adapt their system of objectives autonomously. For instance, the relative weighting of the objectives is modified, new objectives are added or existing objectives are discarded. Adapting the objectives in this way leads to adaptation of the system behaviour. This is achieved by adapting the parameters and where necessary the structure of the system. The term parameter adaptation means adapting system parameters, for instance adjusting the controller gains. Structural adaptations involve reconfigurations and compositional adaptations. It affects the arrangement of system elements and their relationships. Reconfiguration changes the relationships between a fixed set of available elements, while compositional adaptation integrates new elements into the existing structure or removes existing elements out from it.

The self-optimization process is expressed as a series of three actions that are generally carried out repeatedly.

1. **Analysis of the current situation:** Here the current situation includes the state of the system itself and all the observations that have been made about its environment. Such observations may also be made indirectly by communicating with other systems. The current state of the system also includes any records of previously made observations. One essential aspect of this analysis is examining the degree to which the pursued objectives have been fulfilled.
2. **Determination of the system objectives:** The current system objectives may be determined by selection, adaptation or generation. A selection is here understood as choosing one alternative from a fixed discrete finite set of possible objectives, while the adaptation of objectives describes the gradual modification of existing objectives. We speak of generating objectives when new objectives are created independently of the existing ones.

3. **Adaptation of the system behaviour:** This is determined by the three aspects: parameters, structure and behaviour. In this action the reaction at the end of the self-optimization cycle is effected by adapting the system behaviour. The individual adaptation cases may be extremely diverse depending on which level of the mechatronic system (MFM, AMS, NMS) we are dealing with. The domain in which the adaptation takes place also plays a considerable role.

Starting from a certain initial state, a self-optimizing system progresses into a new state on the basis of specific influences, i.e. the system undergoes a state transition. The influences that trigger a state transition are referred as events. The self-optimization process defines the activities, that effect this state transition, and thereby describes the system’s adaptive behaviour.

3 **State-of-the-Art**

There are a wide variety of approaches for domain-spanning specification of technical systems in the early phases of product development. All of them fundamentally map the elements of the system and their interrelations, but no approach specifies the self-optimization process. ModCoDe – a system for the object-oriented modeling of mechatronic product concepts – models active elements and their interrelationships, including associated behaviour models, described by means of block diagrams, state charts and bond graphs [WLB01]. SchemeBuilder is a development tool for modeling and simulating conceptualized mechatronic systems by mapping and linking functions, active principles and components. Formulating behaviour models enables simulation models to be derived automatically [CPD+99]. The emphasis here is on mapping kinematic, dynamic and control behaviour.

Suh subdivides the description of technical systems into the domains: customer, function, physics and process. He formally models the domains and the relationships between them. He defines axioms that enable him to derive modules and determine the ideal solution concept. The formal specification can also be used as the basis for establishing the system's stability [Suh98], [Suh04].

In Buur’s work on the description of mechatronic concepts the emphasis is on modeling functions depending on system states and state transitions [Buu89], [Buu90]. The specification technique for describing mechatronic systems developed by the project iViP focused on mapping models to describe requirements, functions, structures, constraints and shape and the cross-links between them [KTA02].

The ordering concept CARTRONIC [BPF+00] and the specification technique for modeling mechatronic machine tool concepts that was formulated during the project “Planning mechatronic production systems” [ZLT02] both utilize software specification techniques such as UML. Here the focus is on communication relationships. An OMG consortium
formulated SysML™ (System Modeling Language), which is a standard based on UML for the specification, analysis, verification and validation of technical systems. Here the emphasis is on modeling system structures, parameters, requirements and behaviour (such as the system's activities, states and interactions).

Until now the authors do not know any method linking the domain-spanning principle solution and the domain specific concretization.

4 Specification of the Principle Solution

During the conceptual design phase, the development task of self-optimizing systems has to be attempted from multiple views. These views make a simple and intuitive specification possible so that a comprehensive understanding about the structure and the operation of the system can be achieved. Each view is represented as a partial model and interlinked internally in a computer. A complete description of the principle solution consists of the following partial models: 'requirements', 'environment', 'system of objectives', 'functions', 'active structure', 'shape', 'application scenarios' and various types of 'behaviour' of the system. These descriptions utilize a set of new uniform notation that integrates current domain-specific notations.

Figure 3 shows the partial models used to specify the principle solution of self-optimizing systems. Previously, in mechatronics, the focus was placed on the system's active structure, but here the system's states and state transitions are in the foreground, i.e. the self-optimization process and its effects on the active structure and the processes taking place within the system.
**Figure 3: System of coherent partial models describing the principle solution of self-optimizing systems**

**Requirements:** This is the computer’s internal representation of the requirements, based on the requirements catalogue. It constitutes a structured collection of all the requirements for the product under development (e.g. dimensions, performance data). We distinguish three types of requirements, namely desired features, target features and required features.

**Environment:** This model describes the system’s environment and how it is embedded in that environment. It identifies the relevant areas of influence and possible disturbance variables (e.g. external temperature, mechanical stresses, higher-level systems). We also investigate any interactions or reciprocal effects between individual influences, and the possibility of their occurring concurrently. A consistent set of concurrent influences constitutes a “situation”, in which the technical system has to function.

**System of objectives:** This is the representation of the external, inherent and internal objectives and the links between them.

**Application scenarios:** Application scenarios offer a way of reducing the complexity of the development task; they focus on a subset of the system that is being developed together with its environment and one development task that has to be resolved for that subset. These specify how the system is to behave in a given state and a given situation, or how and on the basis of what influences state transitions should occur.
**Functions:** Here we are concerned with a hierarchical classification of the operating functions as a way of defining the system’s basic functionality. The functions dealt with here may be conventional functions, or functions used for self-optimization.

**Active structure:** Here we depict the system elements that represent solution patterns (active principles, software patterns), together with their characteristics and the interrelationships between those system elements. Our objective is to map the basic structure of the self-optimizing system together with all the envisaged system configurations. In this manner it is specified which variables can be detected and therefore also on which influences or events the system basically can react with behaviour adaptations.

**Shape:** This model contains information about quantities, shapes, positions and arrangements, plus the types of active surfaces and points of action of the self-optimizing system. We were already able to make initial rough statements about the system’s general shape during the conceptual design phase.

**Behaviour:** This group includes different types of behaviour. It will always be necessary to model the system states with the associated operative processes and the state transitions with the underlying adaptive processes. The adaptive processes bring out the concrete implementation of the self-optimizing process. Depending on the design task we will need to specify different types of behaviour, such as the system’s kinematics, dynamics or cooperation behaviour.

These partial models are developed interactively, but not sequentially. Each of the development steps undergoes numerous iterations, and their order depends upon the object being developed, the organizational constraints, and – especially – on the preferred approach of the individual developer and the use of a suitable methodology [HA02] [Lin05].

### 5 From Principle Solution to Controller Design

This section addresses the conceptual design of controllers based on the principle solution of self-optimizing systems. Controllers required by self-optimizing systems consists of multiple control algorithms, each designed to implement specific control functionalities. Designing such controllers based on the principle solution requires engineering skills not covered by control theory. These skills are extracting the control functionalities from the principle solution, decomposing the overall control problem and organizing the individual solutions to constitute a complete solution. A procedural model is therefore needed to integrate these skills, as illustrated in Figure 4 below. The procedural model presumes a well-developed system of partial models, as described previously in Section 4. With the partial models in hand, this section describes the transition from the phase of principle solution to the phase of controller design. In Section 6, the procedural model is further exemplified with the development of controllers required by convoy operation of autonomous railway shuttles.
Figure 4: Procedural model from domain-spanning conceptual design to domain-specific controller design

**Phase 1 - Extraction of Control Functionalities:**
Phase 1 involves the extraction of control functionalities from the partial models. Control functionalities refer to the functionalities of the controllers, by which their algorithm should realize the operating functions of the system. From one partial model to the other, all required control functionalities are listed for the system under development.

**Phase 2 - Outline of Control Hierarchy:**
This phase focuses on the partial model 'active structure', which describes solution elements and solution patterns to implement the control functionalities. During this phase, interdependencies among control functionalities are analysed. The types of interdependency determine partly how the control algorithm should be derived. Therefore, understanding on the interdependency is important here for effective coordination among the control algorithms. Besides that, from the hierarchization in the active structure, control engineers construct a simplified skeleton with only the solution elements necessary for controller design. The simplified skeleton gives an overview of system control and serves as the control hierarchy. The control hierarchy resembles the functional decomposition of the control problem. Each of the control functionality in the hierarchy refers to a particular task to be carried out by a control algorithm.
Phase 3 - Conception of Controller Design:
Each of the partial models plays its own role along the controller design process. Therefore, further control-specific information has to be identified from the partial models. Control-specific information includes, for instance, comparison elements, control elements, correction elements, process elements and measurement elements. These elements are organized together in control loops. The outcome is the 'preliminary block diagrams' representing the conceptual layout design of controllers. Phase 3 is repeated for each of the control functionalities. By applying formal design techniques, derivation of control algorithms can be done to implement the well-defined control functionalities.

These three phases above show how information can be extracted from the principle solution, organised and then used in the controller design process. It allows incremental realization of individual controllers until the complete realization of the overall control system. Each controller can be designed, implemented and tested separately while ensuring its functional dependency with the other controllers of the higher, same, or lower hierarchical level. Going from the top to the bottom of the control hierarchy, the overall control problem is decomposed into partial and well defined control problems. On the contrary, going up from the bottom to the top of the hierarchy, individual solutions are combined into a coherent overall solution.

Solving the control problem of self-optimizing system is an iterative process. Some knowledge required to solve the problem is only discovered while solving the problem itself. Therefore, the procedural model supports both the top-down and bottom-up approaches with some iterations. Within the team of control engineers, some control problems specified by the principle solution are analyzed and decomposed, while some other parts are solved by combination of available solutions. Although this procedural model does not describe how to calculate control signals, it is an effective representation platform for the development of entire control system. Thus it is used as a guide for the controller design process of self-optimizing systems.

An important point here is the realization of the synergistic impact carried by the domain-spanning principle solution. From the principle solution, control engineers obtain the basic ideas about what functionality a controller has to implement and how to implement it. Different control structures may exist to implement each of the control functionalities, and therefore there are different realization possibilities. Implementation of a controller design may differ in terms of: quantity of controller, type of controller, controller layout, utilization of observers/filters, etc. The overview of control problem at system level stimulates the consideration of several alternative solutions.

6 Application Example: Development of Advanced Control Strategy for Autonomous Shuttles

The procedural model in Section 5 is demonstrated here using the shuttles developed under research initiative “Neue Bahntechnik Paderborn / RailCab” [http://nbp-www.upb.de]. The demonstration system has been realized on a scale of 1:2.5 at the University of Paderborn. The core of the system comprises autonomous shuttles for transporting passengers and goods according to individual demands rather than a fixed timetable. In this example, the development of control for longitudinal dynamics demanded
by convoy operation of the shuttle is illustrated. Figure 5 shows two shuttles during the
field test of convoy operation on the NBP test track.

![Field test of convoy operation on the NBP test track](image)

Figure 5: Field test of convoy operation on the NBP test track

Figure 6 shows an exemplary section from the active structure of a shuttle. As described in
Section 4, the active structure illustrates the system elements, their characteristics as well
as their relations among each other. Relations here refer to the flows of material,
information and energy, as well as their logical relations. The shuttle is represented here
as a system element on the hierarchy of AMS. The shuttles consist of two driving modules,
which are specified here as logical groups. Each driving module contains a spring and tilt
module, a guidance module and a drive and brake module. Besides that, the shuttle is
equipped with a power supply module coupled by energy management to fulfil the energy
demand of all the modules. All the four modules described here are at the MFM level.

This example focuses on the drive and brake module which enables shuttle motion and
convoy operation. The drive and brake module is driven by a linear motor and is steered
by means of self-optimizing operating point assignment. The linear motor consists of two
parts: the rotor on board the shuttle and the stator in the rail section. The propulsion force
$F_m$ is produced by the magnetic field generated between the rotor and the stator.
Influences such as rolling resistance, wind force, and shuttle weight resist the propulsion
force, and result in the actual velocity of the shuttle. The current position and velocity of
the shuttle are recorded and used as inputs to the controller of longitudinal dynamics. The
controller of longitudinal dynamics consists of cascaded control of position and velocity of
the shuttle, implemented together with current control of both the stator and rotor. The
behaviour adjustment as a result of the control of longitudinal dynamics leads to the
formation and separation of a convoy.
Figure 6: Active structure of a shuttle.

**Phase 1 - Extraction of Control Functionalities:**

This phase involves the extraction of all basic control functionalities from the principle solution. For instance, the drive and brake module requires control of forward and backward motion; the energy transfer from the stator to the on-board power supply module requires a controller which enables the control of bi-directional energy flow [PHG03]; the guidance module requires control in the lateral direction; while the spring and tilt module requires the damping of excessive vibrations. As the result, a list of all control functionalities envisaged for all system modules is obtained.

Figure 7 shows the extraction of control functionalities from the function hierarchy of the drive and brake module. The function ‘to arrive at a specific point’ is implemented by the control of position; the function ‘to accelerate or decelerate’ is implement by the control of velocity; while the function ‘to avoid collision’ is implemented by the control of the distance between the leading shuttle and the following shuttle.
Figure 7: Extraction of control functionalities from the function hierarchy

Phase 2 - Analysis of Control Hierarchy:

A closer inspection of the active structure provides the interdependencies between the MFMs. Due to complex interactions among the four MFMs, only the drive and brake module is focused here. The drive and brake module contains the controllers for position and speed. Their references and operational profiles, however, result from the current status of the complete system. Furthermore, information of the actual position and speed is also demanded by the guidance module and the spring and tilt module. These interdependencies have to be considered during the design process. Figure 8 shows the interdependencies between the drive and brake module with the other MFMs in terms of information flow.
Figure 8: Active structure describing the interdependencies between the drive and brake module and the other MFM.s.

The drive and brake module allows energy transfer to the on-board power supply module depending on the operating point of the doubly-fed linear motor. Besides system status, the relation of the two MFM.s arises from the energy demand. Furthermore the drive and brake module will be affected by system status of the guidance module and the spring and tilt module. Figure 8 also shows the interdependencies between the drive and brake modules installed in two shuttles. The strategy for operating point assignment [PBS+04] is responsible for generating references for the linear drive. However, in case of convoy operation the operating point has to provide the maximum current amplitude for both shuttles. For adjusting the desired thrust, the controllers of the secondary motor parts have to readjust the motor currents [HVB05]. Therefore, the impacts between drive and brake modules of different shuttles have to be considered too.

Phase 3 - Conception of Controller Design:

This phase is demonstrated here only by the conception of controller design with reference to the partial model 'active structure'. The active structure is exemplified in Figure 9 with the conception of convoy control between two shuttles. The upper part shows the conception during the conceptual design phase and the lower part shows its concretization during the controller design phase. Resulting from the information processing in the NMS, references for the following shuttle are calculated by incoming information from the leading
shuttle. Similarly the leading shuttle needs system status from the follower, to avoid braking in an uncontrolled manner. Therefore, safe and smooth convoy formation and separation can be ensured by critical consideration during the conception phase. The convoy control is integrated onto the control of longitudinal dynamics. It constitutes an extension of the position control loop.

Figure 9: Conception for convoy operation of two shuttles

Figure 10 illustrates the conception of cascaded control layout based on the active structure of a shuttle. The upper part shows the active structure of the shuttle while the lower part shows the cascaded controller implemented. The two feedback loops for the position and the velocity are conceptualized in the active structure during the conceptual design phase. The design is then concretized that to control the speed at the inner loop and the position at the outer loop. The entire control design results in dynamic shuttle behaviour. Some designs cannot be made during the conceptual design phase. For
instance, the choice of the controller parameters, as it depends on the dead time of the converter and the time response of the linear motor. Basic design considerations include avoiding overshoot, maximum limit of acceleration as well as jerk in order to provide high riding comfort.

The other partial models also play a role in Phase 3. For instance, influences from the surrounding which carry an impact on the system behaviour is formulated in the partial model ‘Environment’. Details about these disturbance variables are then taken from the partial model ‘Environment’ and incorporated into the system equations. Besides that, the system’s behaviour and the adaptation of behaviour are described by the partial model “Behaviour”. Furthermore, the complexity of the development task can be reduced by referring to the partial model “Application Scenario” which focuses on a particular state and particular situation.
7 Conclusion

The cross-domain specification of the principle solution handles the complexity of self-optimizing systems in a systematic and manageable way. It provides an effective platform for control engineers to understand the requirements and constraints of the entire system. In such a way, qualitative requirements of the envisaged control functionalities as well as preliminary system structures and system boundaries can be laid out right from the conceptual design phase. With the domain-crossing principle solution, optimization of the performance of control systems with given resources is possible.

Acknowledgement

The research project was initiated in the course of the Collaborative Research Centre 614 “Self-Optimizing Concepts and Structures in Mechanical Engineering” funded by the German Research Foundation (DFG) under grant number SFB 614.

References


[GEK01] GAUSEMEIER, J.; EBBESMEYER, P.; KALLMEYER, F.: Produktinnovation; Carl Hanser Verlag, 2001


