14

Architecture and Design Methodology of Self-Optimizing Mechatronic Systems

Prof. Dr.-Ing. Jürgen Gausemeier
Dipl.-Wirt.-Ing. Sascha Kahl
Heinz Nixdorf Institute
Fürstenallee 11, D-33102 Paderborn

Abstract
The conceivable development of information and communication technology will enable mechatronic systems with inherent partial intelligence. We refer to this by using the term "self-optimization". Self-Optimizing systems react autonomously and flexibly on changing operation conditions. They are able to learn and optimize their behavior at runtime. The development of mechatronic and especially self-optimizing systems is still a challenge. A significant milestone within the development is the principle solution. It determines the basic structure as well as the operation mode of the system and is the result of the conceptual design. Additionally it is the basis for the concretization of the system which involves experts from several domains, such as mechanics, electrical engineering/electronics, control engineering and software engineering. This contribution presents a new specification technique for the conceptual design of mechatronic and self-optimizing systems. It also uses the railway technology as a complex example, to demonstrate how to use this specification technique and in which way it profits for the development of future mechanical engineering systems.

Keywords
Design Methodology, Mechatronics, Self-Optimization, Principle Solution, Conceptual Design, Domain-Spanning Specification

1. Introduction

The products of mechanical engineering and related industrial sectors, such as the automobile industry, are increasingly based on the close interaction of mechanics, electronics and software engineering, which is aptly expressed by the term mechatronics. The conceivable development of communication and information technology opens up more and more fascinating perspectives, which move far beyond current standards of mechatronics: mechatronic systems having an inherent partial intelligence. We call these systems "self-optimizing systems". Self-Optimization of a technical system is the endogenous adaptation of the systems' objectives as a reaction to changing influences and the resulting autonomous adjustment of parameters or structure and consequently of the systems' behavior [ADG+08].

According to this self-optimizing systems have the ability to react autonomously and flexibly on changing operation conditions. Thereby self-optimization goes far beyond conventional control and adaptation strategies.
To develop mechatronic and especially self-optimizing systems, still is a challenge. The established design methodologies of the conventional engineering domains are no longer adequate. This particularly applies to the early design phase "conceptual design" which results in the so-called "principle solution". The principle solution represents a significant milestone because it determines the basic structure and the operation mode of the systems and, subsequently, it is the basis for further concretization. This need for action was the starting point for the collaborative research centre (CRC) 614 "Self-Optimizing Concepts and Structures in Mechanical Engineering" at the University of Paderborn funded by the German Research Foundation (DFG).
This contribution presents the essential results of the Collaborative Research Center 614. It first explains the paradigm of self-optimization and the key aspects of such systems. Afterwards it describes in detail the three actions of self-optimization, the so called Self-Optimization Process. For the realization of complex, mechatronic systems with inherent partial intelligence an adequate concept of structure as well as architecture for the information processing is needed. Hence the new concept of the Operator-Controller-Module (OCM) has been developed. This concept is also presented in detail. A new and powerful paradigm, such as self-optimization, naturally calls for new development methods as well as development tools. Therefore a new design methodology for self-optimizing and thus for mechatronic systems is introduced. It divides the development process into two main phases – the "conceptual design" and the "concretization". The main emphasis is on a holistic integrative specification of the principle solution. Therefore a new domain-spanning specification technique is presented. Within the "conceptual design" the specification of the principle solution forms the basis for all the experts' communication and cooperation. It will be described in which way the development activities of the subsequent "concretization", that take place in parallel, are going to be structured, coordinated and how the consistency of these activities is ensured on the basis of the principle solution.
All the works by the CRC 614 use the "Neue Bahntechnik Paderborn/RailCab" as a demonstrator. All examples throughout this contribution refer to that project. RailCab is an innovative railway system which is realized on a test track at a scale of 1:2.5. Autonomous vehicles (RailCabs) that supply transport for both passengers and cargo, establish the core of the system (figure 1). They drive on demand and not by schedule. The RailCabs act in a pro-active way, e.g. in order to reduce the required energy by forming convoys. The actuation is realized by a contact-free dual-feed electromagnetic linear drive [ZS05], [ZBS+05]. The stator of the linear drive is situated between the track and the rotor within the shuttle. The three-phase winding in the stator forms a magnetic field which moves asynchronous along the tracks and propels the vehicle with it. By the magnetic dynamic effects between the stator and rotor magnetic fields´, the vehicle is accelerated and also slowed down. The dual feed allows variable adjustment of the vehicle's magnetic field. Consequently, several RailCabs can be operated on the same stator section with different velocities. The linear drive allows power to be supplied to the vehicle without overhead lines or contact rails. The supporting and guiding of the shuttle take place by using wheel/track contact that allows the usage of already existing railway tracks. With an active tracking module, based on an independent axle chassis with loose wheels, the choice of direction by passing over a switch can now take

place vehicle-sided. In that case, the switches work in a passive way, in contrast to the conventional rail. An active spring technology with an additional tilt technology results in a high travelling comfort. The RailCab's basic technology is placed in the plain-built undercarriage on which the chassis for passengers or cargo will be set upon.

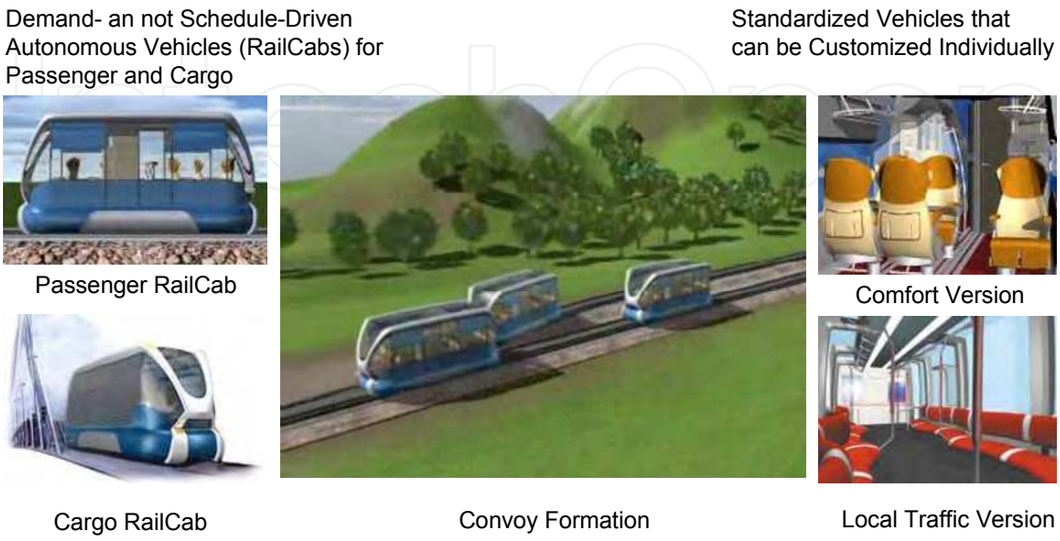


Fig. 1. RailCabs of the project „Neue Bahntechnik Paderborn/RailCab"

2. Self-Optimization

All future intelligent systems of mechanical engineering, regarded in this contribution, rely on mechatronics. The CRC 614 took up the hierarchical structure of complex mechatronic systems suggested by LÜCKEL and added the aspect of self-optimization (figure 2) [LHL01]. The basis consists of so-called mechatronic function modules (MFM) that comprise a mechanical basic structure, sensors, actors and a local information processing, which contains the controller. MFMs that are connected by information technology and/or mechanical elements result in autonomous mechatronic systems (AMS). They also feature information processing. Within this information processing, superior tasks are being realized, such as monitoring, fault diagnosis and maintenance decisions. Additionally targets for the local information processing of the MFM are generated. AMS form the so-called networked mechatronic systems (NMS). NMS are produced just by connecting the AMS parts via information processing. Similar to the AMS, the information processing of the NMS is realizing superior tasks. If the terms are to be transferred in the vehicle engineering, the spring and tilt module would be considered to be a MFM, the RailCab itself with an active chassis an AMS and a convoy a NMS.

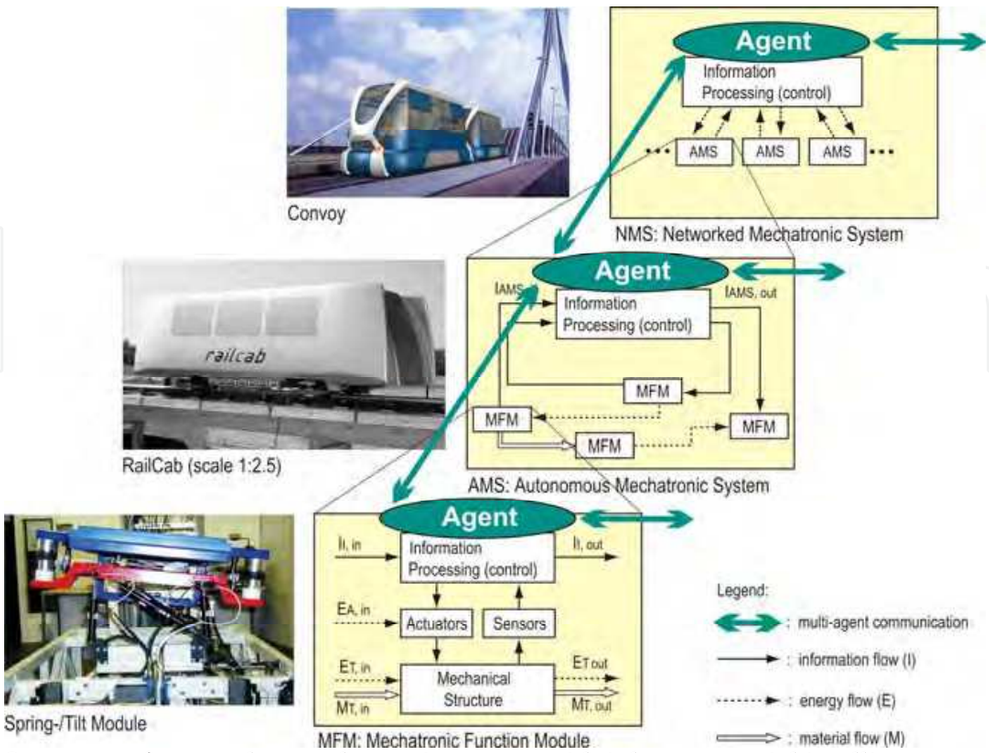


Fig. 2. Structure of a complex mechatronic system with inherent partial intelligence

On every level of this structure it is possible, to add the functionality of self-optimization to the controller. By this, the regarded system's elements (MFM, AMS, NMS) gain inherent partial intelligence. The behavior of the whole system is formed by the communication and cooperation of the intelligent system's elements. From an information processing point of view we consider these distributed systems to be multi-agent-systems.

Against this backdrop, we understand a system's self-optimization as endogenous adaptation on changing operating conditions, as well as a resulting objective-oriented adaptation of the parameters and, if necessary, of the structure and therefore the behavior of the system [ADG+08]. Thus self-optimization enables systems that have inherent "intelligence". They have the ability to react autonomously and flexibly on changing operating conditions.

The key aspects and the operation mode of a self-optimizing system are depicted in figure 3. Using the influences as a basis, the self-optimizing system determines the internal objectives that have to be pursued actively. These internal objectives are based on external ones, whereas those are set from the outside, e.g. by the user or other systems, and also on inherent objectives that reflect the design purpose of the system. Inherent objectives of a driving module can be for example: saving of the driving functions and a high efficiency. If we below talk about objectives, we refer to the internal ones, because those are part of the optimization. Low energy demand, high travelling comfort and low noise emission belong to internal objectives of a RailCab. The adaptation of objectives means, for instance, that the relative weighting of the objectives is modified, new objectives are added or existing objectives are discarded and no longer pursued.

Thus, the adaptation of the objectives leads to an adaptation of the system's behavior. The behavior's adaptation is achieved by an adaptation of the parameters and, if necessary, of

the structure. An adaptation of the parameters means an adaptation of the system's parameters, e.g. the adaptation of a controller parameter.
Adapting the structure, concerns the arrangement and relations of the system's elements. We differentiate between reconfiguration and compositional adaptation. Reconfiguration is the change of the relations of a fixed quantity of elements. Compositional adaptation means the integration of new elements in the already existing structure or the subtraction of elements from the structure. Self-Optimization takes place as a process that consists of the three following actions, called the **Self-Optimization Process**:
1. Analyzing the current situation: The regarded current situation includes the current state of the system as well as all observations of the environment that have been carried out. Observations can also be made indirectly by communication with other systems. Furthermore, a system's state contains possible previous observations that are saved. One basic aspect of this first step is the analysis of the fulfillment of the objectives.
2. Determining the system's objectives: The system's objectives can be extracted from choice, adjustment and generation. By choice we understand the selection of one alternative out of predetermined, discrete, finite quantity of possible objectives; whereas the adjustment of objectives means the gradual modification of existing objectives respectively of their relative weighting. We talk about generation, if new objectives are being created that are independent from the existing ones.

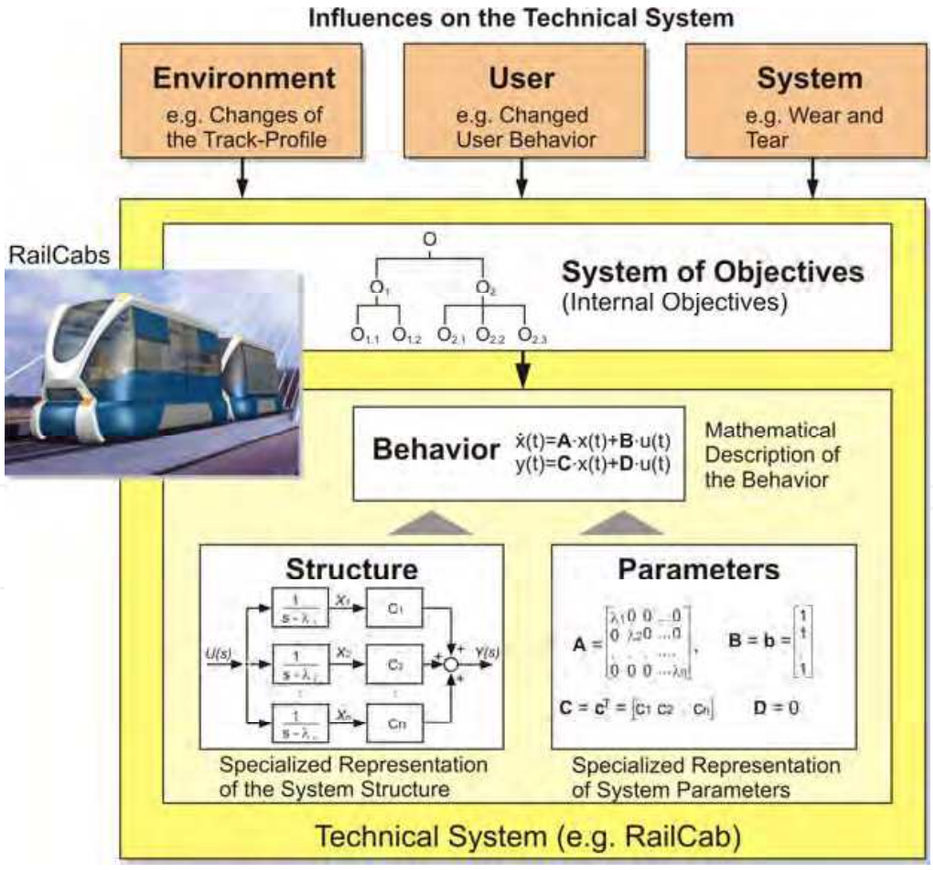


Fig. 3. Aspects of a self-optimizing system – influences on the system result in an adaptation of the objectives and an according adaptation of the system's behaviour

3. Adapting the system's behavior: The changed system of objectives demands an adaptation of the behavior of the system. As mentioned before this can be realized by adapting the parameters and, if required, by adapting the structure of the system. This action finally closes the loop of the self-optimization by adapting the system's behavior.

The self-optimizing process leads, according to changing influences, to a new state. Thus a state transition takes place. The Self-Optimizing Process describes the system's adaptation behavior. This can occur on every hierarchy level of a self-optimizing system shown in figure 2. The realization of mechatronic systems with inherent partial intelligence requires an adequate concept of structure as well as an architecture for the information processing. To make this possible, the new concept of the **Operator-Controller-Module (OCM)** has been developed: [ADG+08]. From an information processing point of view, it corresponds to an agent. Figure 4 shows its architecture. As a result, an OCM can be structured into three levels (Controller, Reflective Operator and Cognitive Operator) which will be examined in detail below.

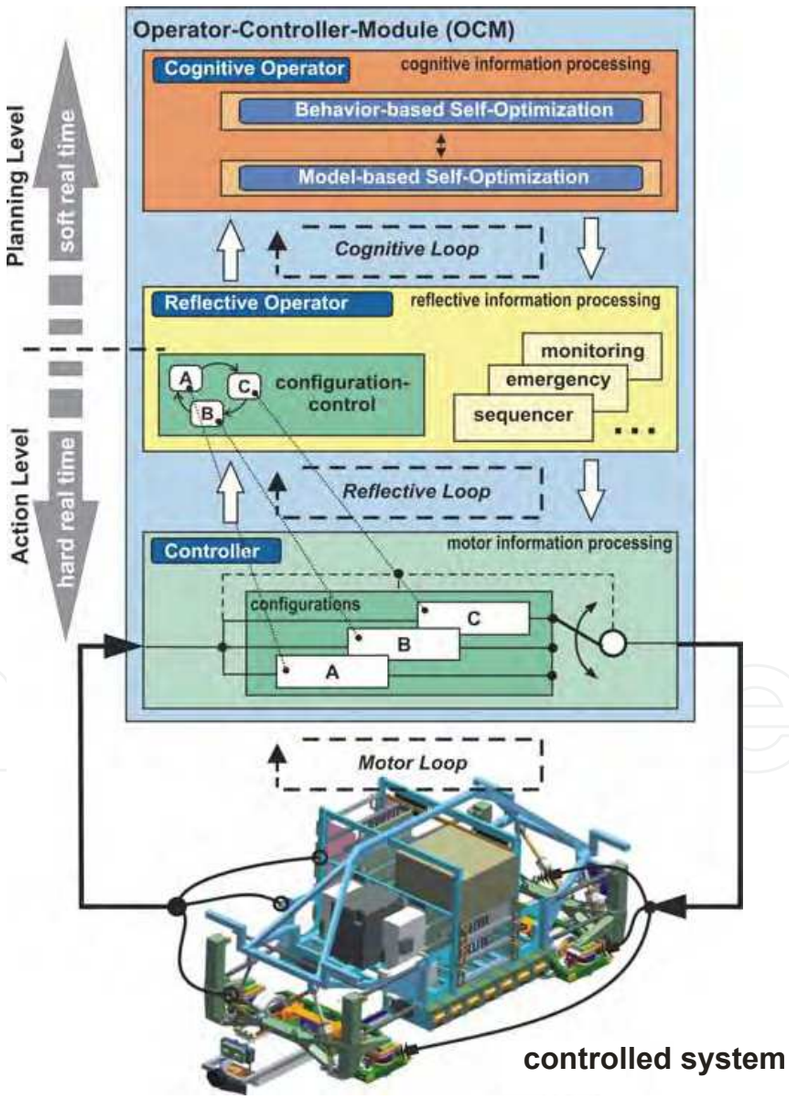


Fig. 4. Architecture of the Operator-Controller-Module (OCM)

- The **Controller** level stands for the control loop with direct access to the technical system. The software at this level operates continuously under hard real-time conditions. The controller itself can be made up of a number of controller configurations with the possibility of switching between them. The changeover takes one step; necessary cross-fading mechanisms and the like are summarized, in turn, into one controlling element.
- The **Reflective Operator** supervises and regulates the controller. It does not access directly the actuators of the system but it modifies the controller by initiating parameter changes or changes of the structure. If changes of the structure do appear (e.g. as in reconfigurations), not just the controllers will be replaced but also corresponding control and signal flows will be switched within the controller itself. Combinations that consist of controllers, circuit elements and corresponding control or signal flows are described as controller-configurations. Figure 4 shows possible controller-configurations, exemplified by the blocks A, B and C. The controlling of the configurations, realized by a state machine, defines which state of the system uses which kind of configuration. It also determines under which circumstances it is necessary to switch between the configurations. The reflecting operator mostly works event-oriented. The close connection with the controller requires a mode of operation in so-called hard real-time. The reflecting operator offers an interface (working as a conjunctional element to the cognitive level of the OCM) between the elements operating not in real-time or soft real-time and the controller. It filters the arriving signals and takes them to the levels underneath. Moreover, the reflecting operator is responsible for the real-time communication between the OCM, which together constitute a composed self-optimizing system.
- The **Cognitive Operator** is the highest level of the OCM-architecture. Here the system uses knowledge on itself as well as on its environment in order to improve its own behavior by using varied methods (such as learning methods and model-based optimization). The main emphasis is put on the cognitive abilities for carrying out of the self-optimizing process. Model-based processes allow a predictable optimization that is, to a large extent, decoupled from the underlying levels while the system is in operation.

Based on the OCM-architecture, the actions of the Self-Optimizing Process (1. analyzing the current situation, 2. determining the system's objectives and 3. adapting the system's behavior) can be realized in various ways. When the self-optimizing adaptation needs to fulfill real-time requirements all three actions are carried out in the reflective operator. Systems that do not have to run the self-optimization in real time can use more elaborate methods, which are settled within the cognitive operator. In that case, the behavior's adaptation is carried out indirectly, relayed by the reflective operator, which needs to synchronize the instructions of the behavior's adaptation with the controller's real-time course. In addition, there might occur mixtures within one single OCM. There are also hybrid forms that occur within a single OCM, when the two described forms of self-optimization take place simultaneously and asynchronously.

3. Challenges during the development of self-optimizing systems

A new and powerful paradigm, such as self-optimization, naturally calls for new development methods as well as development tools. Because of the high complexity and the participation of a multitude of different engineering domains the development of self-optimizing systems is still a challenge.

By the activities of the Heinz Nixdorf Institute of the University of Paderborn a practical guideline for the development of mechatronic systems was established – the VDI-guideline 2206 „Design methodology for mechatronic systems“ (figure 5). The guideline is based on the so called V model of the domain software engineering.

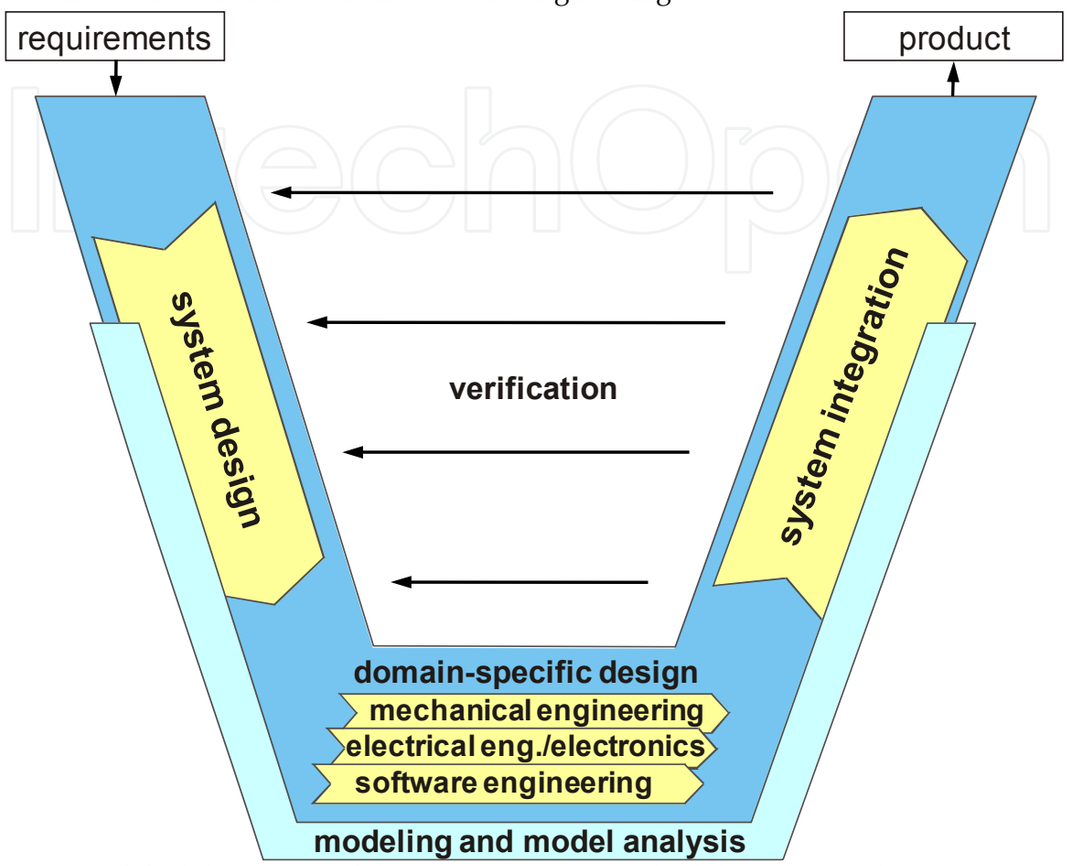


Fig. 5. V model of the VDI-guideline 2206 „Design methodology for mechatronic systems” [VDI04]

Starting point for the system design are the requirements. During the system design the basic structure and the main physical and logical operating characteristics of the system are described in a domain-spanning product concept (synonymous: principle solution). The principle solution forms the basis for the subsequent domain-specific concretization. The term “concretization” describes thereby the domain-specific design of a technical system. During the concretization the participating domains specify the system by using domain-specific procedure models, methodologies and tools. Afterwards the integration of the results from the individual domains into an overall system allows to investigate the characteristics of the system. During the different development steps the reliability and safety of the characteristics is analyzed by computer models. The result of one cycle of the V model is a product of a specific stage of maturity (e.g. functional model, prototype, pre-production model and repetition part). For the development of a product of a high stage of maturity a number of cycles are required. The presented model is a first step on the way to a holistic design methodology for mechatronic systems.

The holistic domain-spanning system design – thus the left bough of the V model – still bears the major challenge in the development of mechatronic systems. There is a gap between the normally used list of requirements, which is more or less a rough specification of the total system and, hence, leaves much space for interpretation, and well-established specification techniques of the involved domains used within the domain-specific concretization. This gap is the reason, why the engineers run in deep trouble when they have to integrate their results in the system integration in the right bough of the V model. This applies to mechatronic as well as to self-optimizing systems. It is the question, how established design methodologies can be extended to face these challenge. For the system design we became aware, that the basic structure of the conceptual design in classical mechanical engineering design methodology (formulation of the requirements, definition of the functions and searching for active principles for the realization of the tasks [PBF+07]) is also valid for mechatronic and self-optimizing systems. By looking into this more deeply, it became clear that an extension of the design methodology was urgently necessary. This appears, for instance, in the use of solution patterns as well as in the necessity of modeling the environment, application scenarios and the system of objectives. Therewith a holistic approach for the conceptual design of self-optimizing systems, comprising of an integrative specification of the description of the principle solution and a holistic procedure model, is the main need for action on the way to a design methodology for self-optimizing systems (compare figure 6). Both elements are presented in the following chapters.

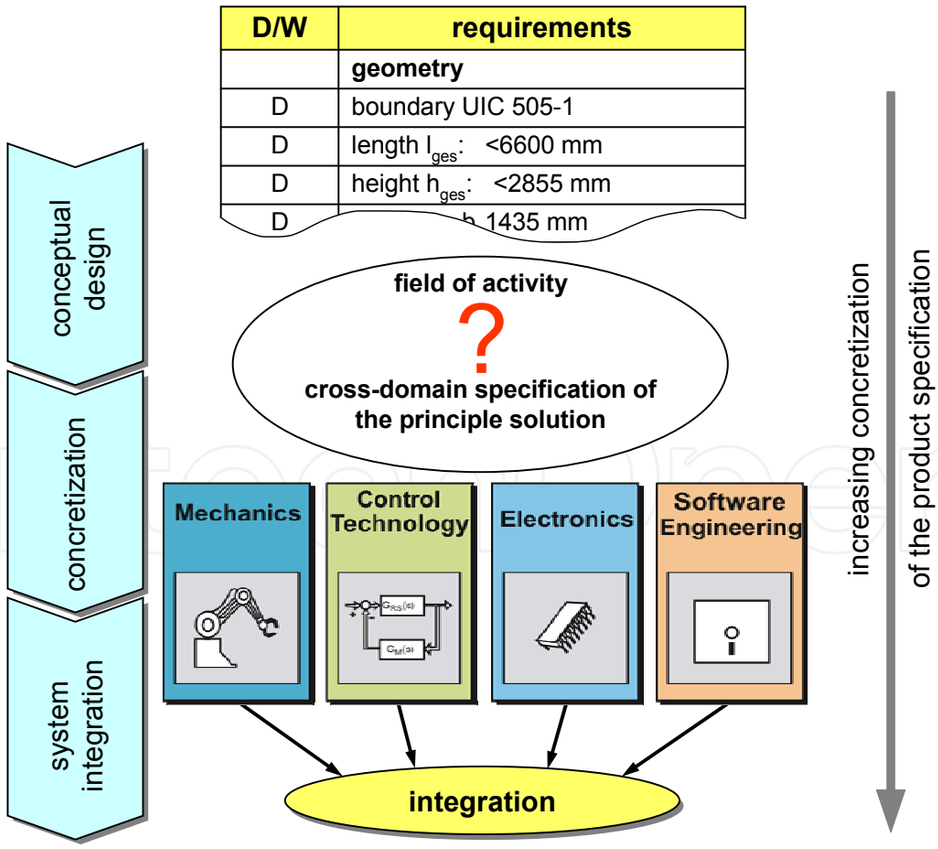


Fig. 6. Central challenge: a new specification technique for the description of the principle solution of a mechatronic respectively self-optimizing system

4. Specification of the principle solution

In the following, we present a specification technique for the description of the principle solution of a self-optimizing system. It is also applicable for mechatronic systems. The specification technique is based on the research of FRANK, GAUSEMEIER and KALLMEYER [GFD+08], [GEK01]. It became clear that a comprehensive description of the principle solution of a highly complex system needs to be divided into aspects. Those aspects are, according to figure 7: requirements, environment, system of objectives, application scenarios, functions, active structure, shape and behavior. The behavior consists of a whole group because there are different kinds of behavior, e.g. the logic behavior, the dynamic behavior of multi-body systems, the cooperative behavior of system components etc. The mentioned aspects are represented by partial models. The principle solution consists of a coherent system of partial models because the aspects are in relationship with each other and ought to form a coherent system. It is necessary to work alternately on the aspects and the according partial models. Nevertheles there is a certain order.

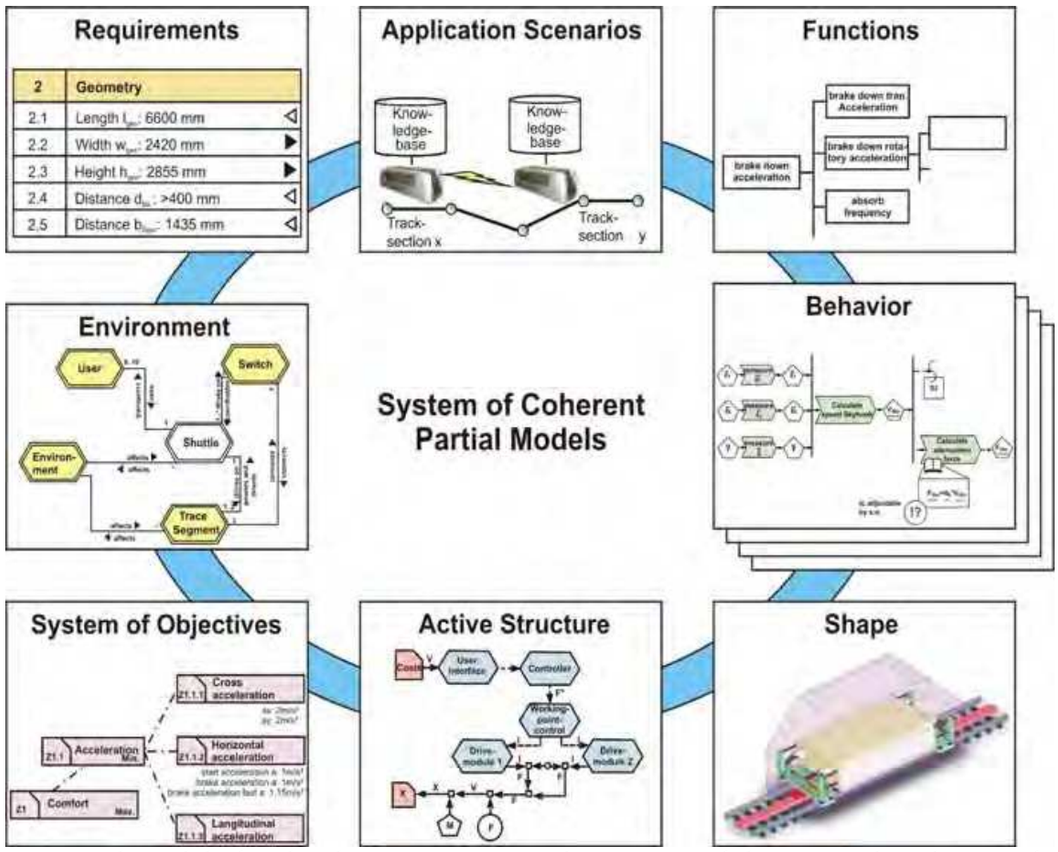


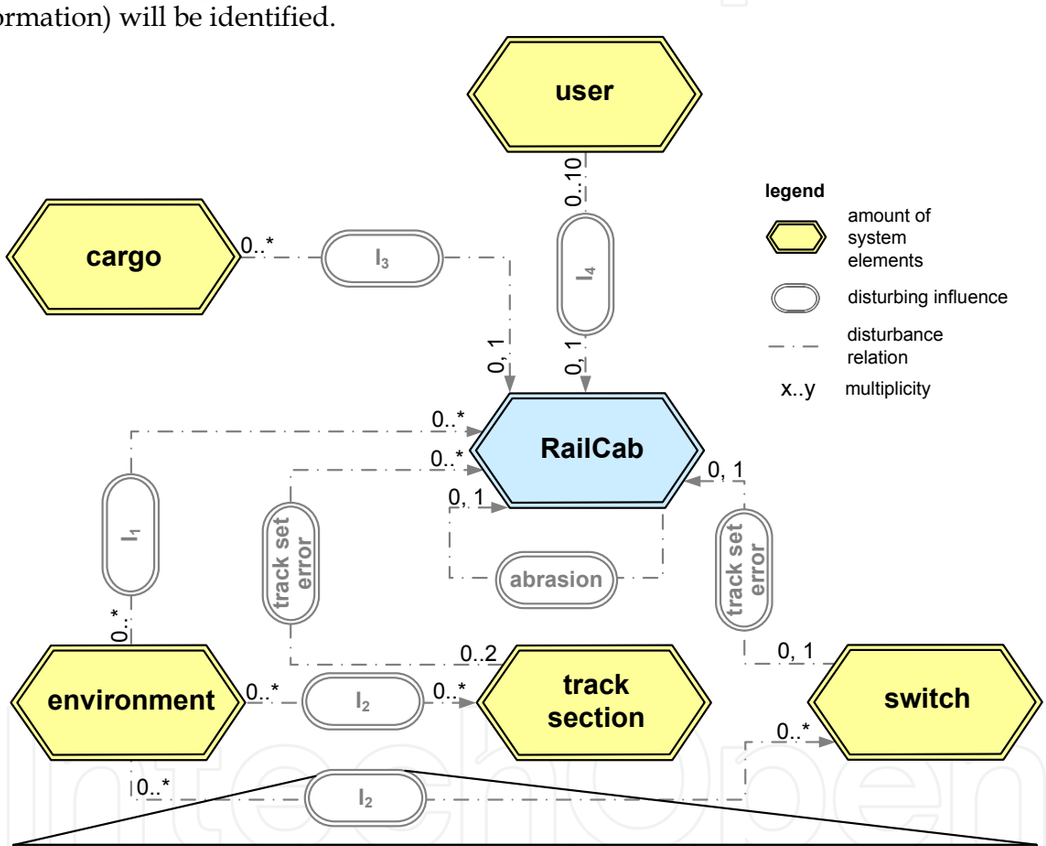
Fig. 7. Partial models for the domain-spanning description of the principle solution of self-optimizing systems

The description of the environment, the application scenarios and requirement serve as the starting point. They are usually followed by the system of objectives, the function hierarchy and the active structure. The active structure represents the core of the principle solution in

conventional mechanical engineering. The modeling of states and the state transitions as well as the impacts on the active structure play a decisive role in the specification of a self-optimizing system. This kind of modeling takes place within the group of behavior models. An example is given in subchapter 3.3. The following subchapters explain the partial models, the relationships between the partial models and the specific characteristic of the specification of self-optimizing systems.

4.1 Partial models

Environment: This model describes the environment of the system that has to be developed and its embedding into the environment. Relevant spheres of influence (such as weather, mechanical load, superior systems) and influences (such as thermal radiation, wind energy, information) will be identified.



No.	influence (denotation and short description)	attribute	characteristic	range of tolerance	frequency of appearance	kind of influence: supporting, disturbing, neutral
1	terrain	inclination	> 5 %	0,30 %	medium	disturbing
2	wind	velocity	> 50 km/h	2 km/h	often	disturbing
3	rain	liters per minute	1 l/min	0,1 l/min	medium	disturbing
4	snow	depth of snow	20 cm	4 cm	rare	disturbing
5	ice	adhesion	<< 1	3	rare	disturbing
6	flood	level above track	> 10 cm	2 cm	rare	disturbing
7	leaves	thickness on rail	> 8 mm	2 mm	rare	disturbing

Fig. 8. Modeling of the RailCab's environment (cut-out)

Disturbing influences on the system's purpose will be marked as disturbance variables. Furthermore, the interplays between the influences will be examined. We consider a situation to be one consistent amount of collectively occurring influences, in which the system has to work properly. We mark influences that cause a state transition of the system as events. Catalogues, that imply spheres of influences and influences, support the creation of environment models. Figure 8 shows, in detail, the specification of the RailCab's environment. The users and the cargo affect the driving behavior of a RailCab, e.g. by their weight. Influences of the environment, such as wind, snow, ice and leaves, affect both, the RailCab's driving behavior as well as the state of the track sections and switches. Track set errors of track sections also influence the RailCab's driving behavior as well as the abrasion of the RailCab itself. The example concretizes the amount of influences I_2 in the influence table (figure 8).

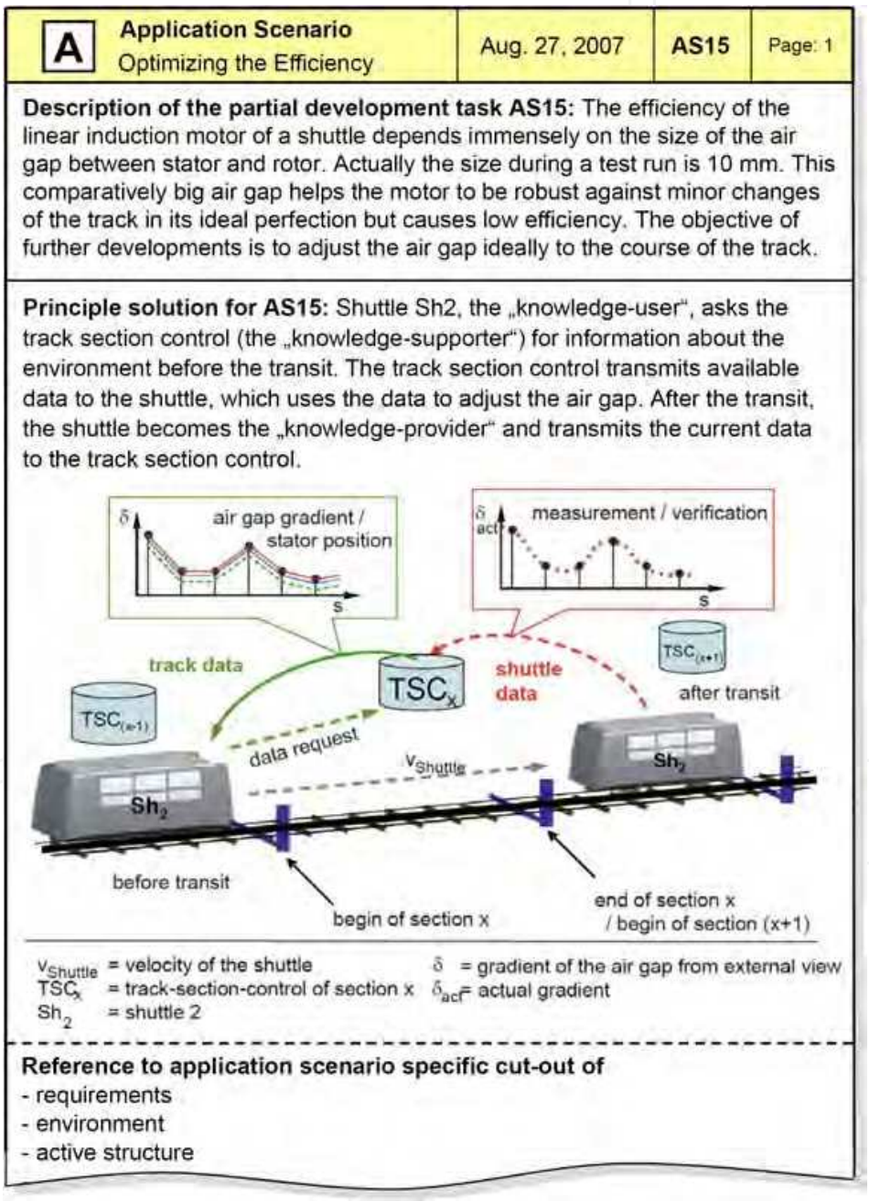


Fig. 9. Application scenario for the example "optimizing the efficiency" [GFP+08]

Application scenario: Application scenarios form first concretizations of the system. They concretize the system's behavior in a special state and a special situation and furthermore, what kinds of events initiate a state transitions. Application scenarios characterize a problem, which needs to be solved in special cases, and also roughly describe the possible solution. One application scenario of the RailCab "optimizing the efficiency" is shown in figure 9.

Requirements: This aspect considers the computer-intern representation of the requirements. The list of requirements sets up its basis. It presents an organized collection of requirements that need to be fulfilled during the product development (such as overall size, performance data). Requirements are separated into demands and wishes [PBF+07]. Every requirement is verbally described and, if possible, concretized by attributes and their characteristics. Several checklists assist the setting up of requirements; see for example [PBF+07], [Rot00], [Ehr03].

System of objectives: This aspect includes the representation of external, inherent and internal objectives and their connections (see chapter 2). A cut-out of the system of objectives of the RailCab is shown in figure 10. The external and inherent objectives are represented as a hierarchical tree. The hierarchy relations are specified by logical relations with declarations of the hierarchy criterion "is part-objective of". The potential internal objectives derive from the external and inherent objectives. The impact between the objectives will be expressed by an assisting influence matrix. The matrix shows if the objectives work in mutual support, if they influence each other negatively or if they are in a neutral relationship. The system is able to follow such systems simultaneously and without any problems, which work mutually and also those systems that have neutral relations. But if the systems influence each other in a negative way, this is an indication for the need of optimization. Instead of an influence matrix, graphs that model objectives and their interplays can be used.

Functions: This aspect concerns the hierarchical subdivision of the functionality. A function is the general and required coherence between input and output parameters, aiming at fulfilling a task. For the setting up of function hierarchies, there is a catalogue with functions which is based on BIRKHOFER [Bir80] and LANGLOTZ [Lan00]. This catalogue has been extended by functions, which especially describe self-optimizing functionality. Functions are realized by solution patterns and their concretizations. A subdivision into sub functions is taking place until useful solution patterns have been found for the functions.

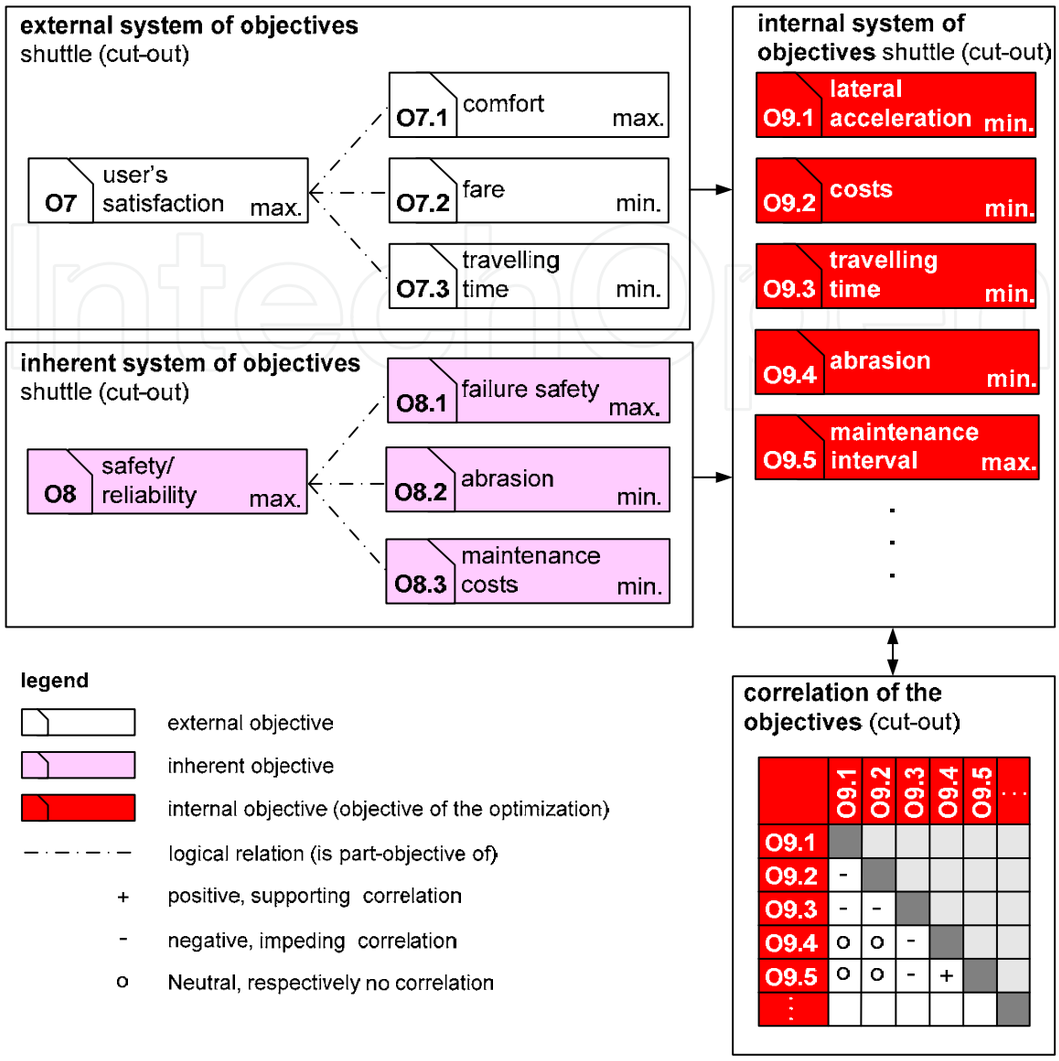


Fig. 10. The RailCab's system of objectives (cut-out)

Active structure: The active structure describes the system elements, their attributes as well as the relation of the system elements. It is the target to define the basic structure of a self-optimizing system, including all system configurations which can be thought anticipated. Figure 11 visualizes a cut-out of a shuttle's active structure. The active structure consists of system elements, such as drive and break modules, air gap adjustment, operating point control, tracking modules, spring- and tilt modules, energy management and their relations. Furthermore, incoming parameters are described, such as *comfort, costs* and *time*, which are external objectives of the user. The system is structured by logic groups in order to improve the necessary clearness. In this case, for example, the system elements of a driving module are combined. The system elements, which deal with the determination of system objectives of the self-optimization process, are marked by a slanting arrow (here: operating point control and air gap adjustment).

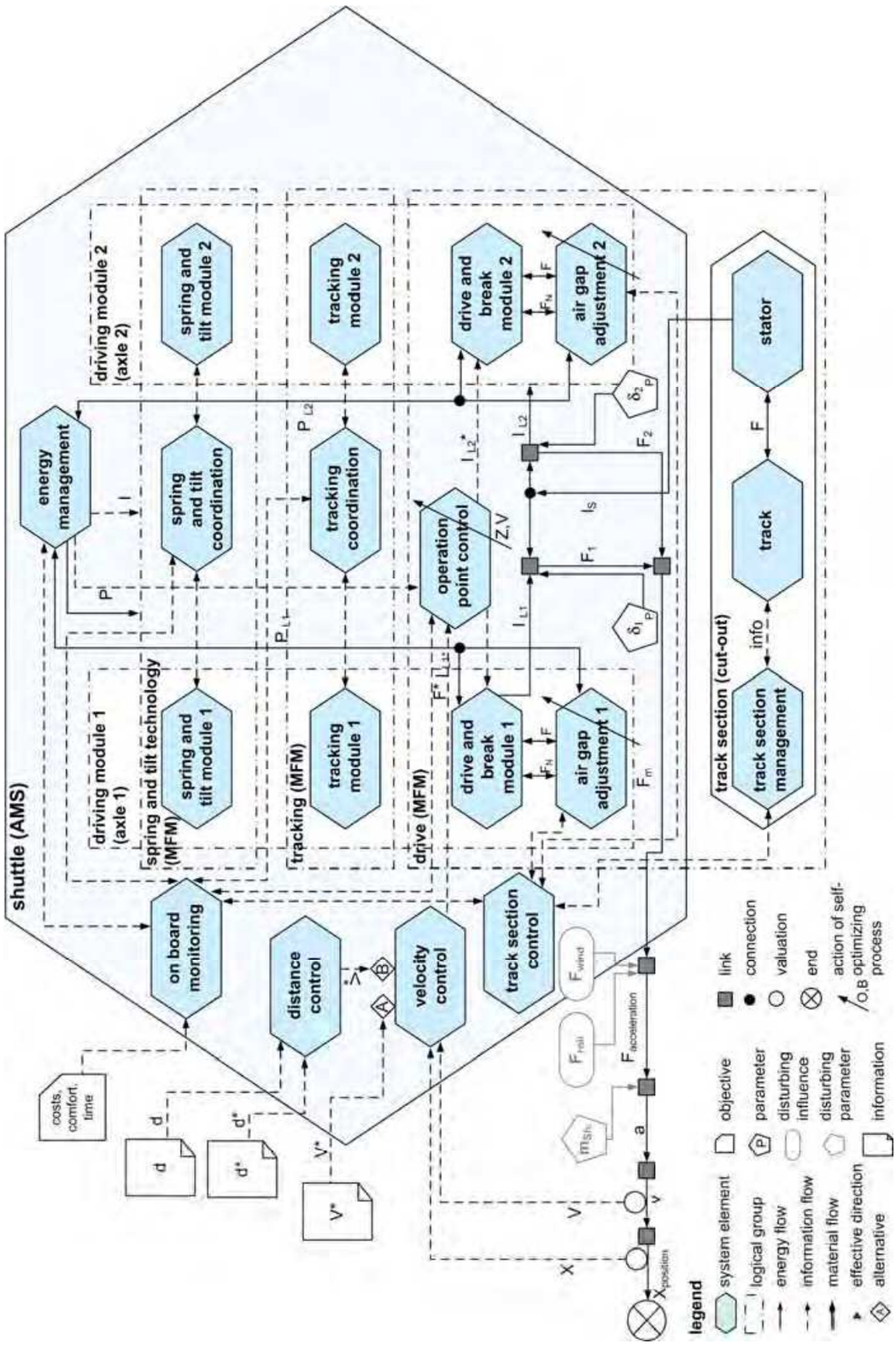


Fig. 11. Active structure of a shuttle (cut-out)

Shape: This aspect needs to be modeled because first definitions of the system's shape have to be carried out already in the phase of the conceptual design. This especially concerns working surfaces, working places, surfaces and frames. The computer-aided modeling takes place by using 3D CAD systems.
Behavior: This group of partial models comprises several kinds of behavior. Basically, what is needed to be modeled are the system's **states** with the connected **operation processes** and the **state transitions** with the **adaptation processes**. The adaptation processes represent the definite realization of the self-optimizing process. If there are several systems taking part in the self-optimizing process, the interplay of these systems needs to be described. Depending on the development task, more kinds of behavior, such as kinematics, dynamics or electromagnetic compatibility of the system's components need to be specified.

- The partial model **Behavior – States** defines the states and state transitions of a system. All of the system's states and state transitions, which can be thought ahead and thus, need to be considered, as well as the events initiating a state transition need to be described. Events can be characteristic influences on the system or already finished activities.
- The partial model **Behavior – Activities** describes the mentioned operation processes, that take place in a system's state, and the adaptation processes, which have the typical features of self-optimization. All in all, the processes are modeled by activities. "Determine fulfillment of current objectives" or "select adequate parameters and configuration" etc. are examples of such activities. Figure 12 introduces a cut-out of the self-optimizing process of the application scenario "optimizing the efficiency". The self-optimizing process is structured by logic groups in the analysis of the situation, the determination of objectives and the behavior adaptation. Within the analysis of the situation, the track data, the RailCab inputs (such as force, efficiency and safety) and information of the energy management module are being interrogated. Furthermore a continuous determination of the degree of fulfillment of the current objectives takes place. The analysis of the situation happens in soft real time. Based on certain information, the air gap adjustment forms a new system of objectives within the determination of the objectives themselves. Right at the beginning of the behavior's adaptation, the identification of the air gap's course concerning the track section takes place. Out of that air gap's course, useful parameters and accordingly configurations (for the self-optimizing air gap adjustment) are chosen in connection with the new system of objectives. Those parameters do not just find their use as an optimization result in the RailCab. Because they can be sent back to the RailCab's according track section control, they will be reused for future RailCabs when they pass over a track section.

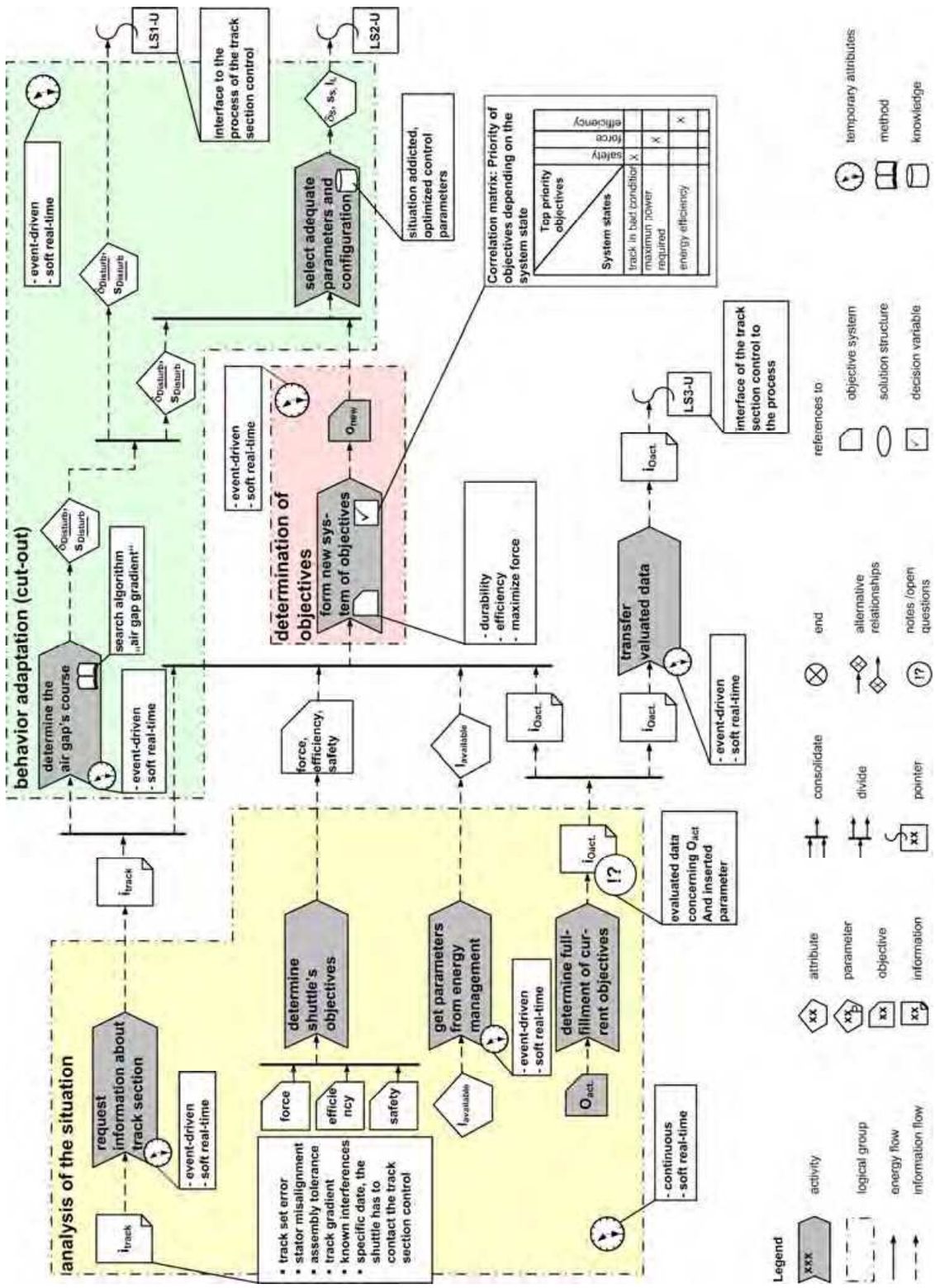


Fig. 12. Behavior – Activities for the application scenario „otimizing the efficiency (cut-out)“

- The partial model **Behavior – Sequence** describes the interaction of several system elements. The activities, being carried out during the interaction of the system elements, and the inter-changed information, are modeled in a chronological order.

4.2 Interrelations between the partial models

The partial models represent the different aspects of the principle solution of a self-optimizing system. The interrelations between the partial models which describe the coherence of the partial models are of high importance. Those interrelations are built up between the constructs of the relating partial models. There are, for example, functions (construct of the partial model *functions*) that are realized by system elements (construct of the partial model *active structure*). These system elements perform activities (construct of the partial model *behavior – activities*), whereas the activities might result out of the functions of the partial model *functions*. There could also be the achieving of a certain temperature (construct *influence* of the partial model *environment*) as an event (construct of the partial model *behavior – states*) that causes the activation of a new state (construct of the partial model *behavior – states*) and other activities. Table 1 shows a couple of interrelations between the partial models. The interrelations are shown directed to the right, i.e. in the table's left side there are the constructs which cause correlation, on the table's right side there are the constructs affecting the connections (an example in the 3rd line, table 1).

construct	partial model	kind of interrelation	construct	partial model
system element	active structure	realizes	function	functions
system element	active structure	performs	activity	behavior – activities
system element	active structure	takes	state	behavior – state
system element	active structure	persuades (opt.)	objective	system of objectives
system element	active structure	has (opt.)	volumes	shape
activity	behavior – activities	results from	function	functions
requirement	requirements	sets boundaries for	volumes	shape
requirement	requirements	decides	function	functions
function	functions	results from	requirement	requirements
influence/event	environment	activates	state	behavior – state
influence/event	environment	activates	activity	behavior – activities
•••				

Table 1. Interrelations between the partial models (cut-out)

A system element within the partial model *active structure* takes up a state in the partial model *behavior – states*. Optional interrelations are marked by *(opt.)*. Taking the information

in table 1 as a basis, a so-called integration model is created, which complements all the already described partial models.

4.3 Particularities within the specification of self-optimizing systems

Chapter 1 already pointed out that the self-optimizing process initiates a new state of the system. The system is transformed from one configuration into another. The partial model *behavior – states* displays all relevant states of the system. It also contains all the events initiating a state transition. The configuration of a system in a specific state is described by its active structure. That means, the active structure can be differently shaped in different states, for example, if different elements of the system (controllers, sensors) are used for the execution of the self-optimizing process. A system's behavior in a certain state is described by its operation process. Operation processes are for example the acquisition of information about the environment, the derivation of adequate control interactions, and the controlling itself. State transitions are realized by adaptation processes, i.e. by self-optimizing processes. The operation and adaptation processes are modeled in the partial model *behavior – activities*. In order to describe the self-optimizing process, all of the three partial models need to be considered simultaneously (figure 13). Every state of the partial models *behavior – states* is assigned to an operation process of the partial model *behavior – activities*, which is operating actively in that state. Moreover, every state is related to a configuration of the active structure, which also actively operates. One example: The state *S5*, the respective operation process and the configuration of the active structure are emphasized by light grey colored, logical groups. The operation process takes place in a periodic way.

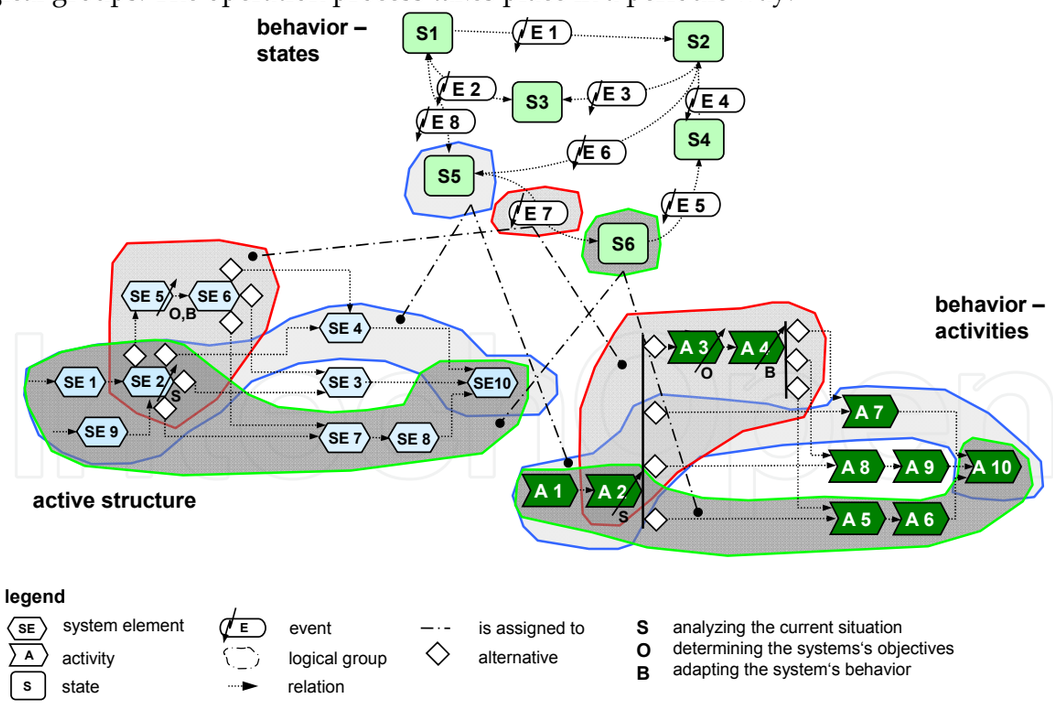


Fig. 13.Cooperation of the partial models active structure, behavior – states and behavior – activities in order to describe the self-optimization (simplified visualization of the principle)

Now – when event E7 appears, an adaptation process is triggered. Therefore, the necessary system elements are activated. Both, the adaptation process and the configuration of system elements, are assigned to the event E7 (see medium grey background in figure 13). After performing the adaptation process, the system takes over the new state S6. A new operation process and a new configuration of system elements are activated. They are colored in a dark grey within figure 13. The adaptation process and the used system elements are no longer activated.

5. Conceptual design of self-optimizing systems

As mentioned in chapter 2, the basic construction and the operation mode of the system are defined within the conceptual design phase. The basic procedure is divided into four sub-phases (figure 14), which are explained in detail below. [GFD+08]

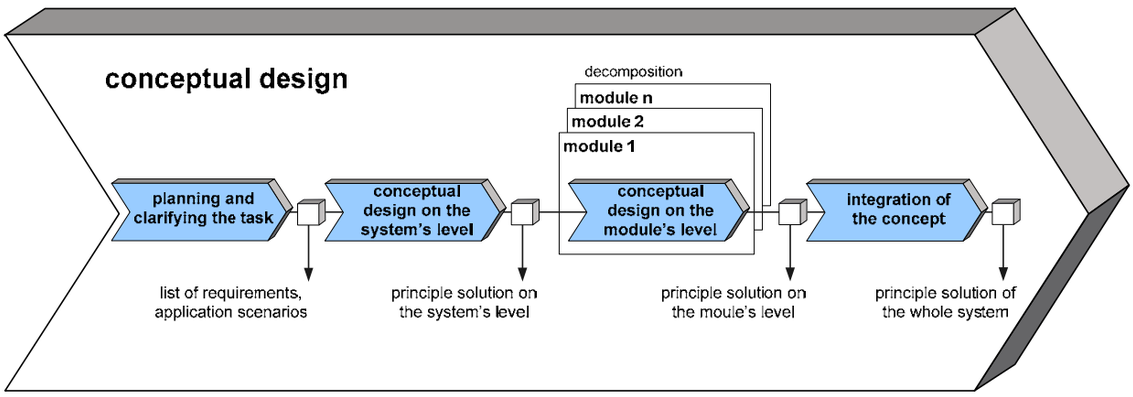


Fig. 14. Process of conceptual design of self-optimizing systems

Planning and clarifying the task

This sub-phase identifies the design task and the resulting requirements on the system is worked out in here (figure 15). At first the task is analyzed in detail. At this the predefined basic conditions for the product, the product program, and the product development are taken into account. This is followed by an analysis of the operational environment which investigates the most important boundary conditions and influences on the system. The external objectives emerge next to disturbances. Beyond that, consistent combinations of influences, so-called situations, are generated. By the combination of characteristic situations with a first discretion of the system's behavior, application scenarios occur. By using the structuring procedure by STEFFEN it is possible to identify a development-oriented product structure for the system and design rules, which guide the developers to realize this product structure type [Ste07]. The results of this sub-phase are the list of requirements, the environment model, the aspired product structure type and the assigned design rules as well as the application scenarios.

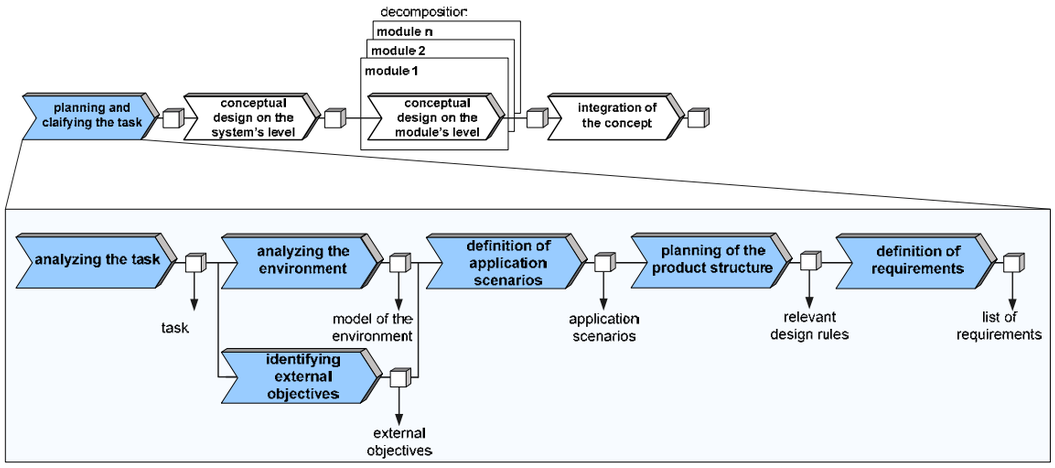


Fig. 15. Conceptual design phase "planning and clarifying the task"

Conceptual design on the system's level

Based on previously determined requirements of the system, solution variants are developed for each application scenario (figure 16). The main functions are derived from the requirements and set into a function hierarchy.

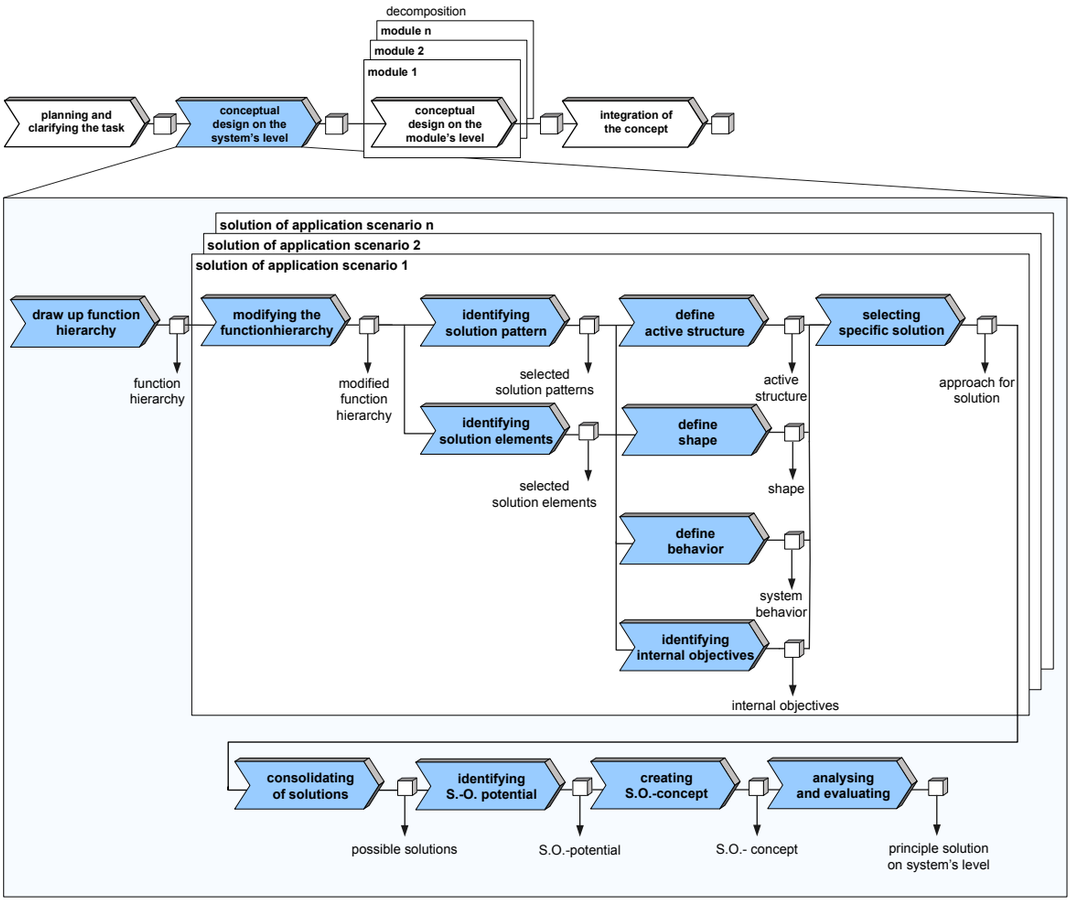


Fig. 16. Conceptual design phase "conceptual design on system's level"

The function hierarchy needs to be modified according to the specific application scenarios, e.g. irrelevant functions are removed and specific sub-functions are added. Then there is a search for "solution patterns" in order to realize the documented functions of the function hierarchy, which will be inserted into a morphologic box.
We use "solution pattern" as a general term. A pattern describes a reoccurring problem and also the solution's core of the problem [AIS+77]. Taking this as a starting point, it results in the classification shown in figure 17. We differentiate between solution patterns that rely on physical effects and between patterns exclusively serving the data processing. The design methodology of mechanical engineering describes the first group as active principles; they describe the principle solution for the realization of a function. The course of development concretizes active principles to material components and patterns of information processing to software components. The relations between active principles and components are of the type n:m; the characteristic depends on the basic method of embodiment design (differential construction method and integrated construction method). Within the integral construction, several active patterns are realized by one component; whereas in the differential construction several components fulfill one active pattern. This is exactly the same in the field of information processing. Basically, a definite modern mechanical engineering system consists of a construction structure that means an arrangement of shape-marked components within a space and their logic aggregation to assemblies and products, and a component structure that means the compound of software components.

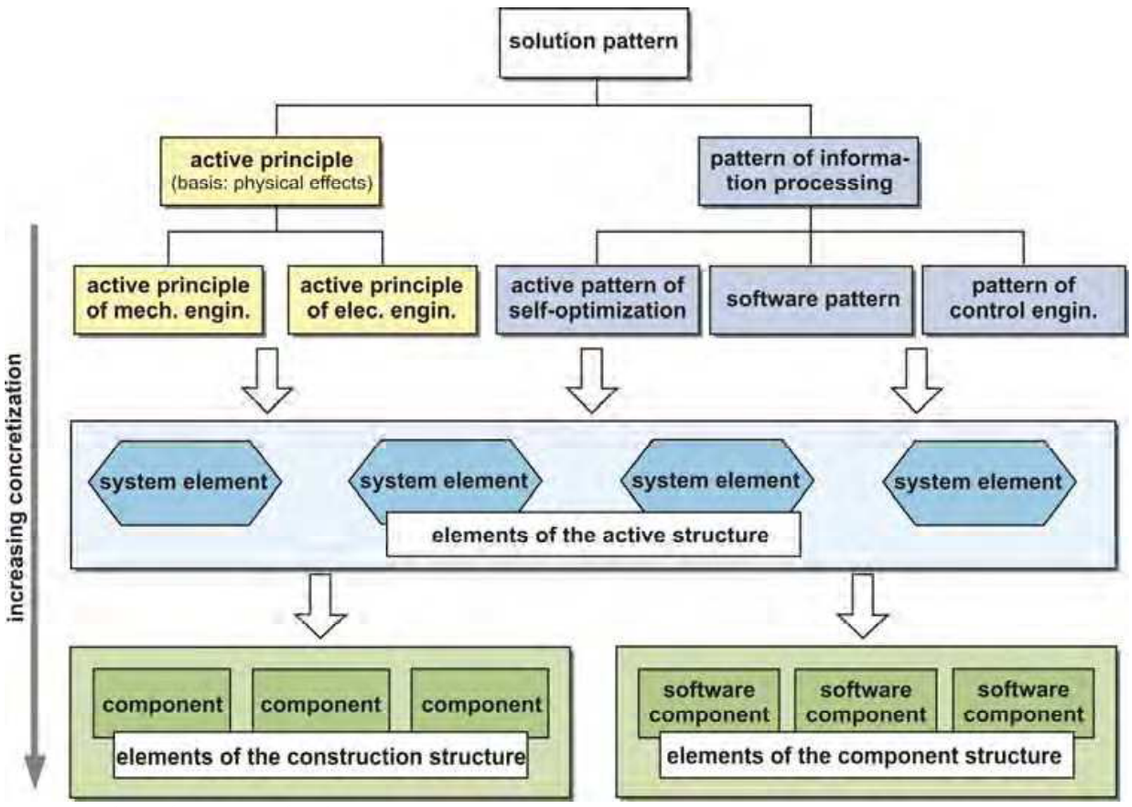


Fig. 17. classification of solution patterns

In some times, there are already existing, well-established solutions which we call "solution elements". If there are such solution elements, they will be chosen instead of the abstract solution patterns. The search for solution patterns is supported by a solution pattern catalogue. We use the consistency analysis in order to determine useful combinations of solution patterns of the morphologic box [Köc04]. As a result, there will be consistent bunches of solution patterns, with a solution pattern for each function.
The consistent bunches of solution patterns form the basis for the development of the active structure. In this step, the refinement of the solution patterns to system elements takes place as well. System elements form an intermediate step between solution patterns on one side and shape-marked components or rather software components on the other side. Based on the active structure, an initial construction structure can be developed because there are primal details on the shape within the system elements. In addition, the system's behavior is roughly modeled in this step. Basically, this concerns the activities, states and state transitions of the system as well as the communication and cooperation with other systems and subsystems. The analysis of the system's behavior produces an imagination of the optimizing processes, running within the system. The external, inherent and internal objectives can be defined.
The solutions for the application scenarios need to be combined. It is important that workable configurations are created which make a reconfiguration of the system possible. Keeping this information in mind, it is identified if there is a containing potential of self-optimization at all. There is a potential for self-optimization if the changing influences on the system require modifications of the pursued objectives and the system needs to adjust its behavior. If there is potential for self-optimization, the function hierarchy needs to be complemented by self-optimizing functions. In particular solution patterns of self-optimization are applied to enable self-optimizing behavior. The resulting changes and extensions of system structure and system behavior need to be included appropriately.
The best solution for each application scenario is chosen and these solutions are consolidated to a principle solution on the system's level. Afterwards, an analysis takes place which looks for contradictions within the principle solution of the system and which contradictions might be solved by self-optimization. Self-optimizing concepts for such contradictions are defined, which contain the three basic steps of self-optimization. The principle solution of a self-optimizing system on the system's level is the result of this phase.

Conceptual design on the module's level
The principle solution on the system's level describes the whole system. It is necessary to have a closer look at the solution, in order to give a statement on the technical and economical realization of the principle solution. For that purpose, the system is decomposed into modules by using the already mentioned structuring procedure by STEFFEN. The decomposition is based on the aspired product structure [Ste07], [GSD+09]. Afterwards a principle solution for each single module is developed. The development of a principle solution for each single module corresponds to the "conceptual design on the system's level", starting out with "planning and clarifying the task". This phase results in principle solutions on the module's level.

Integration of the concept
The module's principle solutions will be integrated into a detailed principle solution of the whole system. Again there is an analysis in order to find contradictions within the principle solutions of the modules and it is checked if these contradictions can be solved by self-optimization. Concluding, a technical-economical evaluation of the solution takes place. The result of this phase is a principle solution of the whole system that serves as a starting point for the subsequent concretization.

Integration of the concept: The module's principle solutions will be integrated into a detailed principle solution of the whole system. There is an analysis in order to find contradictions within the principle solutions of the modules. Again it will be checked if these contradictions can be solved by self-optimization. Concluding, a technical-economical evaluation of the solution is taking place. The result of that phase is a principle solution of the whole system that serves as a starting point for the subsequent concretization. This concretization is carried out parallel in the specific domains (mechanical engineering, electrical engineering, control engineering and software engineering). Chapter 7 gives further information on this.
On the basis of an example, the phases p*lanning and clarifying the task* as well as *conceptual design on the system's level* will be described into detail. There will not be any further consideration of the c*onceptual design on the module's level* because it operates by analogy with the c*onceptual design on the system's level*. The i*ntegration of the concept* has also been explained and is not being discussed anymore.

6. The role of the principle solution during the concretization

The communication and cooperation of the developers from the different domains throughout the whole development process is very important for a successful and efficient development of self-optimizing systems. The principle solution forms the basis for this communication and cooperation.
Within the conceptual design phase the domain-spanning development tasks are carried out in a cooperative way. Within the concretization the developers work on different modules and in different domains. Thus their specific development tasks in one domain of a module need to be synchronized with those of other domains respectively other modules. The development processes for the modules are synchronized by one superior process of the total system (figure 18). Within this process comprehensive aspects of the system like the shell or the dynamics of the whole system are developed in detail. [GRD+09]

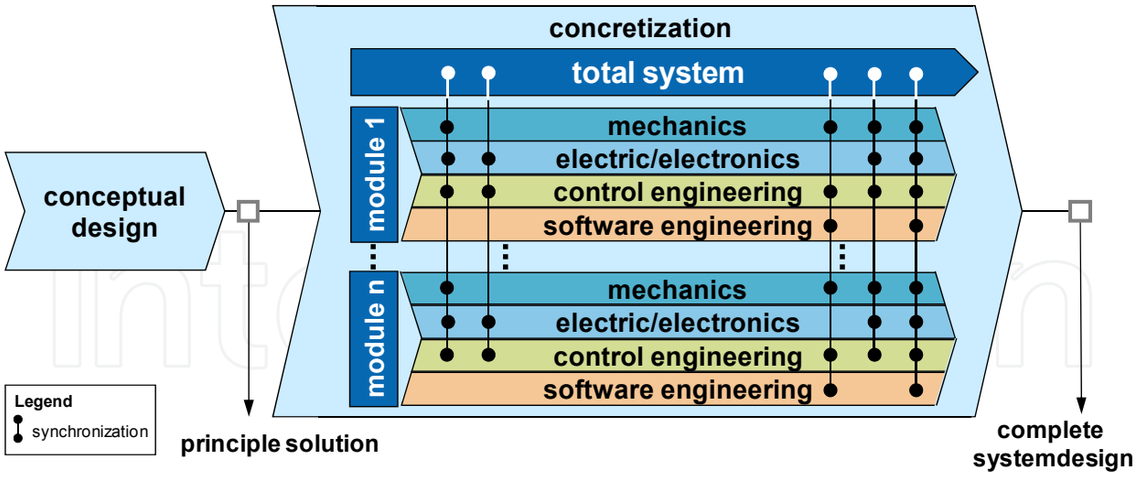


Fig. 18. Basic structure of the development process [GRD+09]

Furthermore, the information, based in the principle solution, serves as a fundament for deducing of domain-specific concretization tasks. In a first step, the system elements of a domain and their relations within the active structure will be identified. After that will be analyzed what kind of domain-specific functions are fulfilled by the system elements, which requirements they have to comply and which behavior is appropriate in certain situations. Following this, it will be checked if domain-specific requirements need to be added. In case of a software engineering, the necessary software components of the component structure, including the input- and output parameters, can be deduced by the system elements of the active structure (figure 18) [GSD+09].

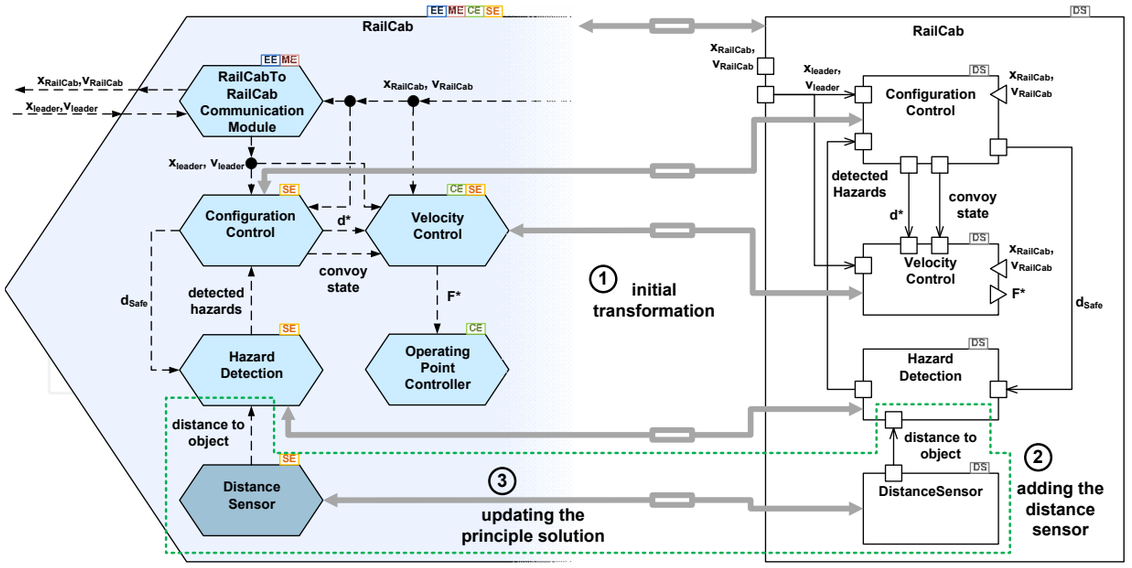


Fig. 19. The transformation from the active structure into a component diagram (software engineering) [GSD+09]

In case of changes occur during the domain-specific concretization, which affect other domains have to be transferred back into the principle solution. This happens for example if

there will be identified additional internal objectives during the course of concretization of a self-optimization process (in the frame of the determination of objectives). Thus the principle solution becomes a domain-spanning system model for the concretization. The aim is to keep this domain-spanning system model and the domain-specific models consistently. Figure 20 schematically shows the versions of the domain-spanning system specification and the different domain-specific models that are created in the course of the concretization. The shown change scenario can be realized by the use of automated model transformations [GSD+09].

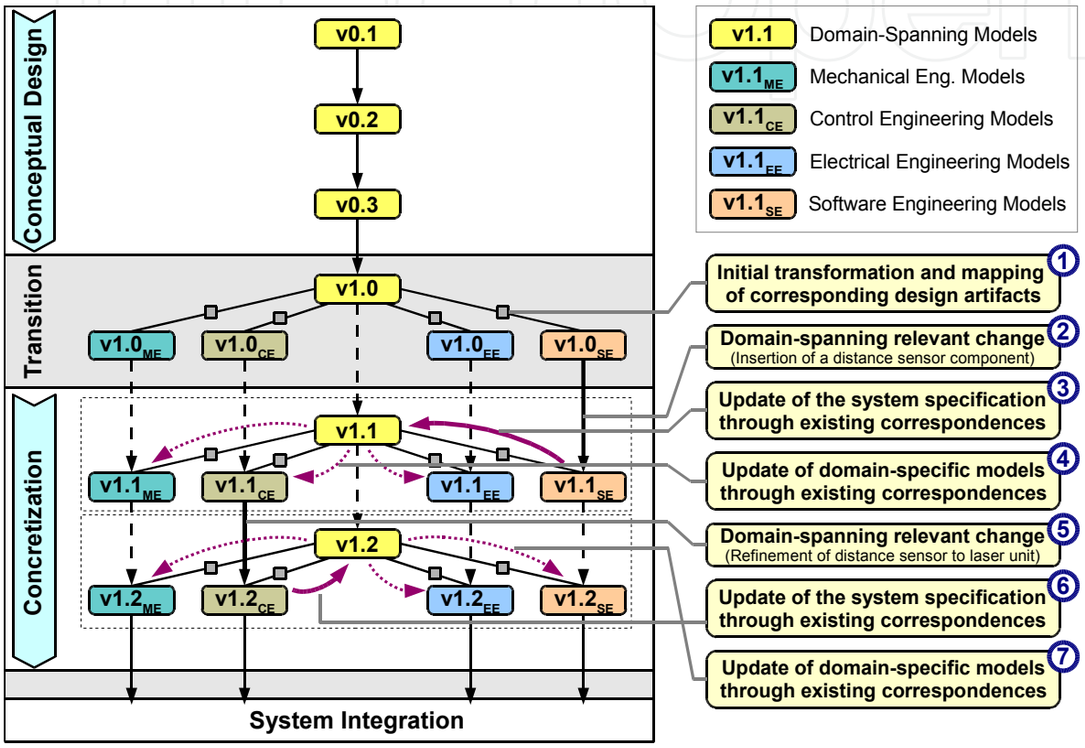


Fig. 20. Propagation of relevant changes between the domain-specific models and the domain-spanning system specification [GSD+09]

7. Conclusion

The paradigm of self-optimization will enable fascinating perspectives for the future development of mechanical engineering systems. These systems rely on the close interaction of mechanics, electrical engineering/electronics, control engineering and software engineering, which is aptly expressed by the term mechatronics. At present there is no established methodology for the conceptual design of mechatronic systems, let alone for self-optimizing systems. Concerning the conceptual design of such systems, the main challenge consists in the specification of a domain-spanning principle solution, which describes the basic construction as well as the mode of operation in a domain-spanning way. The presented specification technique offers the possibility to create a principle solution for advanced mechatronic systems, with regard to self-optimizing aspects, such as “application scenarios” and “system

of objectives". Simultaneously it outperforms classic specification techniques by appropriately encouraging the conceptual design process. It is fundamental to the communication and cooperation of the participating specialists and enables them to avoid design mistakes, which base on misunderstandings between them. It has been described in what way the according concretization, which takes place parallel to the participating domains, is going to be structured and coordinated on the basis of the principle solution. The practicability of the specification technique and the appropriate methodology was demonstrated by the example of a complex railway vehicle.

8. Acknowledgement

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

9. References

[ADG+08] ADELT, P.; DONOTH, J.; GAUSEMEIER, J.; GEISLER, J.; HENKLER, S.; KAHL, S.; KLÖPPER, B.; KRUPP, A.; MÜNCH, E.; OBERTHÜR, S.; PAIZ, C.; PODLOGAR, H.; PORRMANN, M.; RADKOWSKI, R.; ROMAUS, C.; SCHMIDT, A.; SCHULZ, B.; VÖCKING, H.; WITKOWSKI, U.; WITTING, K.; ZNAMENSHCHYKOV, O.: Selbstoptimierende Systeme des Maschinenbaus – Definitionen, Anwendungen, Konzepte. HNI-Verlagsschriftenreihe, Band 234, Paderborn, 2008

[AIS+77]Alexander, C.; Ishikawa, S.; Silverstein, M.; Jacobson, M.; Fiksdahl-King, I.; Angel, A.: A Pattern Language. Oxford University Press, New York, 1977

[Bir80]Birkhofer, H.: Analyse und Synthese der Funktionen technischer Produkte. Dissertation, Technische Universität Braunschweig, 1980

[Ehr03]Ehrlenspiel, K.: Integrierte Produktentwicklung. Carl Hanser Verlag, München, 2003

[GEK01]Gausemeier, J.; Ebbesmeyer, P.; Kallmeyer, F.: Produktinnovation - Strategische Planung und Entwicklung der Produkte von morgen. Carl Hanser Verlag, München, 2001

[GFD+08]Gausemeier J., Frank U., Donoth J. and Kahl S. Spezifikationstechnik zur Beschreibung der Prinziplösung selbstoptimierender Systeme des Maschinenbaus – Teil 1/2. Konstruktion, Vol. 7/8 and 9, July/August and September 2008, pp. 59-66/ pp. 91-108 (Springer-VDI-Verlag, Düsseldorf).

[GRD+09]Geiger, C.; Reckter, H.; Dumitrescu, R.; Kahl, S.; Berssenbrügge, J.: A Zoomable User Interface for Presenting Hierarchical Diagrams on Large Screens. In: 13th International Conference on Human-Computer Interaction (HCI International 2009), July 19-24, 2009, San Diego, CA, USA, 2009

[GSD+09]Gausemeier, J.; Steffen, D.; Donoth, J.; Kahl, S.: Conceptual Design of Modularized Advanced Mechatronic Systems. In: 17th International Conference on Engineering Design (ICED`09), August 24-27, 2009, Stanford, CA, USA, 2009

[GSG+09]Gausemeier, J.; Schäfer, W.; Greenyer, J.; Kahl, S.; Pook, S.; Rieke, J.: Management of Cross-Domain Model Consistency during the Development of Advanced Mechatronic Systems. In: 17th International Conference on Engineering Design (ICED`09), August 24-27, 2009, Stanford, CA, USA, 2009

[Köc04] Köckerling, M.: Methodische Entwicklung und Optimierung der Wirkstruktur mechatronischer Systeme. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, HNI-Verlagsschriftenreihe Band 143, Paderborn, 2004

[Lan00] Langlotz, G.: Ein Beitrag zur Funktionsstrukturentwicklung innovativer Produkte. Dissertation, Institut für Rechneranwendung in Planung und Konstruktion, Universität Karlsruhe, Shaker-Verlag, Band 2/2000, Aachen, 2000

[LHL01] Lückel, J.; Hestermeyer, T.; Liu-Henke, X.: Generalization of the Cascade Principle in View of a Structured Form of Mechatronic Systems. 2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2001), Villa Olmo; Como, Italy

[PBF+07]Pahl, G., Beitz, W., Feldhusen, J., Grote, K.-H.: Engineering Design – A Systematic Approach. ed. 3, 2007, Springer Verlag, London, 2007

[Rot00]Roth, K.-H.: Konstruieren mit Konstruktionskatalogen. Springer-Verlag, , Band 1 Konstruktionslehre, 3. Auflage, Berlin, 2000

[Ste07]Steffen, D.: Ein Verfahren zur Produktstrukturierung für fortgeschrittene mechatronische Systeme. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, HNI-Verlagsschriftenreihe, Paderborn, Band 207, 2007

[VDI04]Verein Deutscher Ingenieure (VDI): VDI-Richtlinie 2206 - Entwicklungsmethodik für mechatronische Systeme. Beuth-Verlag, Berlin, 2004

[ZBS+05] Zimmer, D.; Böcker, J.; Schmidt, A.; Schulz, B.: Elektromagnetische Direktantriebe im Vergleich. In: Antriebstechnik, no. 2/2005, Vereinigte Fachverlage GmbH, Mainz, 2005

[ZS05]Zimmer, D.; Schmidt, A.: Der Luftspalt bei Linearmotor-getriebenen Schienenfahrzeugen. In: Antriebstechnik, no. 2/2005, Vereinigte Fachverlage GmbH, Mainz, 2005

Mechatronic Systems Simulation Modeling and Control

Edited by Annalisa Milella Donato Di Paola and Grazia Cicirelli

ISBN 978-953-307-041-4
Hard cover, 298 pages
Publisher InTech
Published online 01, March, 2010
Published in print edition March, 2010

This book collects fifteen relevant papers in the field of mechatronic systems. Mechatronics, the synergistic blend of mechanics, electronics, and computer science, integrates the best design practices with the most advanced technologies to realize high-quality products, guaranteeing at the same time a substantial reduction in development time and cost. Topics covered in this book include simulation, modelling and control of electromechanical machines, machine components, and mechatronic vehicles. New software tools, integrated development environments, and systematic design methods are also introduced. The editors are extremely grateful to all the authors for their valuable contributions. The book begins with eight chapters related to modelling and control of electromechanical machines and machine components. Chapter 9 presents a nonlinear model for the control of a three-DOF helicopter. A helicopter model and a control method of the model are also presented and validated experimentally in Chapter 10. Chapter 11 introduces a planar laboratory testbed for the simulation of autonomous proximity manoeuvres of a uniquely control actuator configured spacecraft. Integrated methods of simulation and Real-Time control aiming at improving the efficiency of an iterative design process of control systems are presented in Chapter 12. Reliability analysis methods for an embedded Open Source Software (OSS) are discussed in Chapter 13. A new specification technique for the conceptual design of self-optimizing mechatronic systems is presented in Chapter 14. Chapter 15 provides a general overview of design specificities including mechanical and control considerations for micro-mechatronic structures. It also presents an example of a new optimal synthesis method to design topology and associated robust control methodologies for monolithic compliant microstructures.

Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen

IntechOpen