Active Patterns for Self-Optimization

Schemes for the Design of Intelligent Mechatronic Systems

Andreas Schmidt
UNITY AG, Lindberghring 1, D-33142 Büren, Germany,
Andreas.Schmidt@unity.de, http://www.unity.de

Abstract. Self-optimizing mechatronic systems react autonomously and flexibly to changing conditions. They are capable of learning and optimize their behavior throughout their life cycle. The paradigm of self-optimization is originally inspired by the behavior of biological systems. The key to the successful development of self-optimizing systems is a conceptual design process that precisely describes the desired system behavior. In the area of mechanical engineering, active principles based on physical effects such as friction or lever are widely used to concretize the construction structure and the behavior. The same approach can be found in the domain of software-engineering with software patterns such as the broker-pattern or the strategy pattern. However there is no appropriate design schema for the development of intelligent mechatronic systems covering the needs to fulfill the paradigm of self-optimization. This article proposes such a schema called Active Patterns for Self-Optimization. It is shown how a catalogue of active patterns can be derived from a set of four basic active patterns. This design approach is validated for a networked mechatronic system in a multiagent setting where the behavior is implemented according to a biologically inspired technique – the neuro-fuzzy learning method.

1 Introduction – Self-Optimization in Mechatronic Systems

Future systems in the area of mechanical engineering will comprise configurations of intelligent system elements, where the communication and cooperation between these elements shape the behavior of the overall system. In terms of software engineering these are distributed systems of interacting agents. Agents are autonomous and adaptive function modules which can themselves initiate actions. These function modules are heterogeneous subsystems with mechanical,

electronic and information technology components. The agents' behavior can be modified while the system is in operation – this is expressed by the term "adaptive".

A self-optimizing system is characterized by four fundamental aspects (Fig. 1): the target system, in the sense of a hierarchy of a number of targets; the structure, e. g. the topology of mechanical components, sensors and actuators; the behavior, which is the system's reaction to influences from its environment; and the parameters that characterize the system components [1].

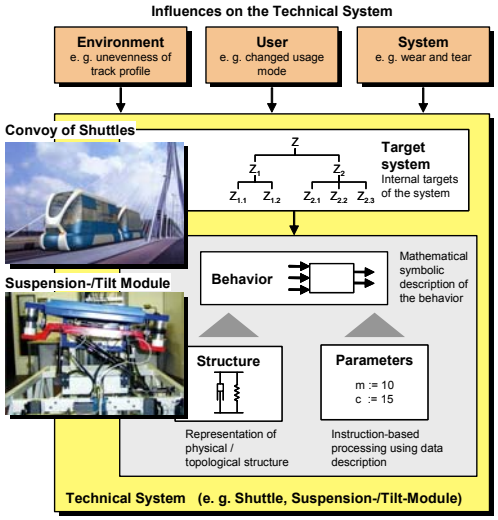

Fig. 1. Aspects of Self-Optimizing Systems

According to [2], intelligent mechatronic systems can be divided up into three layers: the *Multifunction-Module* layer (MFM) that is close to the sensor-/actuator, e. g. suspension-/tilt-modules. The *Autonomous Mechatronic System* (AMS) layer covers system elements that act autonomously in its environment such as single shuttles. The *Networked Mechatronic System* (NMS) layer represents unions of AMS, e. g. convoys that pursue common goals such as crossing a switch[1].

The aim is to carry out self-optimization on the basis of mathematical models, e. g. using a realistic physical model of the controlled system supplemented by excitation and evaluation models. Frequently, it will not be practicable to use models for reasons of cost, so model-based self-optimization is combined with what is called "behavior-based self-optimization" which acts quasi-nondeterministic. This means that changes occurring during operation are sensed and analyzed, and then, depending on the results of this analysis, either another appropriate mathematical optimization model is loaded, or, if the limitations of available models are exceeded, the system reverts to using past experience in the form of learned structures or

[1] The sample mechatronic system originates from the New Railway Technology project Paderborn (NBP) [3]. NBP has set-up a test-track where railway shuttles autonomously drive on an innovative magnetic track system.

parameter settings from its knowledge base. The self-optimization process proceeds continuously and repeatedly according to the subsequent three actions:

1. **Analysis of current situation:** The system records its own state and the state of its environment. The necessary information may be obtained by communicating directly with other systems or by accessing previously recorded observations.
2. **Determination of targets:** The system determines its current target system in view of the current situation, and, if necessary, also adapts it.
3. **Adaptation of the system behavior:** The adaptation itself is carried out by modifying the parameters, the structure, and/or the behavior of individual system elements.

2 Current Situation – Design of Intelligent Mechatronic Systems

The design of self-optimizing systems is based on systems engineering [4], design methods of conventional mechanical engineering [5] and the design methodology of mechatronics [6] and extends those methods with essential aspects of the self-optimization paradigm. The conception phase constitutes one of the most decisive stages within the design of self-optimizing systems (Fig. 2). This is when fundamental functionalities (Function Hierarchy) and the structure (Construction Structure and Component Structure) of the system are determined.

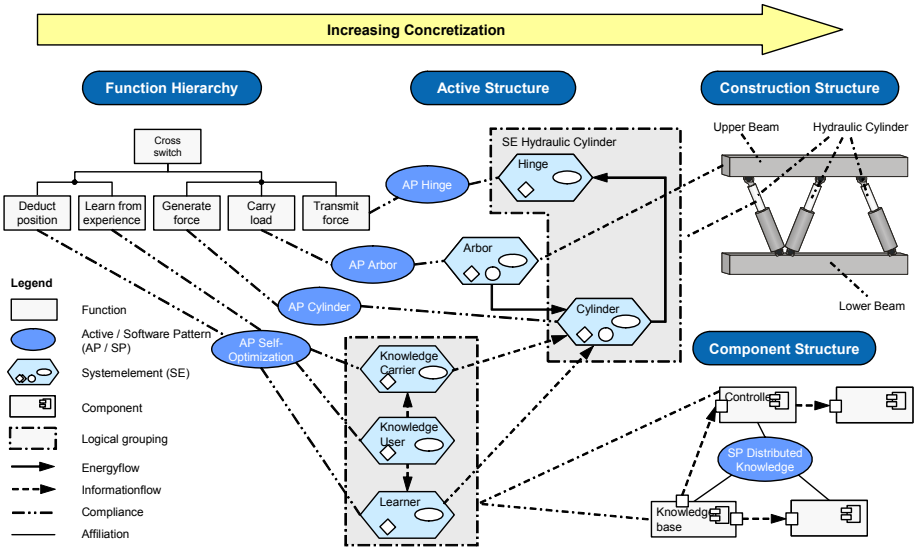

Fig. 2. Core steps of the early stages of system conception

In order to reuse successfully proven previous system engineering knowledge, active patterns are utilized. Active patterns contain template system elements and behavior to realize functions that are concretized in an active structure. Fig. 3 depicts

a categorization of domain-specific patterns [1], e. g. active principles AP of mechanical engineering according to [5] such as *AP Cylinder* in Fig. 2 or software patterns SP such as the *broker pattern* according to [7] or *SP Distributed Knowledge* in Fig. 2. However, those patterns do not address the specific needs for the superior paradigm of self-optimization, namely specifying intelligent and autonomous behavior in an unknown or partially known environment by analysis of the current situation, determination of targets and adaptation of the system behavior. This is where the demand for active patterns for self-optimization comes into play shown as AP Self-Optimization (Fig. 2) and categorized as a pattern of information processing (Fig. 3).

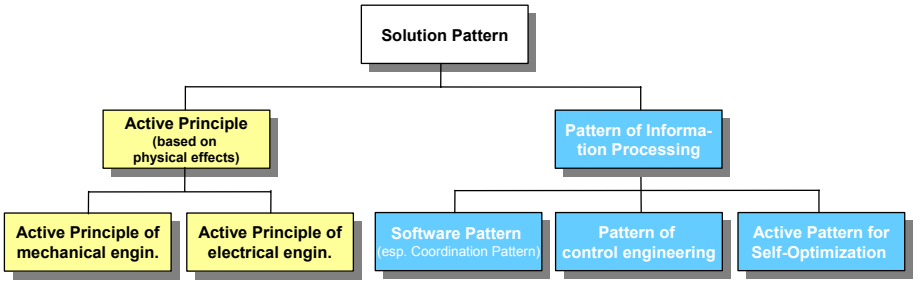

Fig. 3. Category of Patterns for the Design of Intelligent Mechatronic Systems

3 Approach – Design with Active Patterns for Self-Optimization

Active patterns for self-optimization (AP_{SO}) realize functions for self-optimizing systems such as autonomous planning, cooperation, and learning. AP_{SO} constitute templates which specify generally accepted, autonomous and intelligent behavior by using principle-models, application-scenarios, structure-models, behavior-models and method-models (Fig. 4). The principle-concept characterizes the basic idea of the AP_{SO}. It is used to allow the designer an intuitive access to the AP_{SO}. Application-scenarios depict situations in which the AP_{SO} have already been applied successfully in the past. Those scenarios shall help the designer to select an appropriate AP_{SO} for the task at hand. The structure-model specifies necessary participating system-elements and their relations among each other. One or more behavior-models describe adaptation-processes as a kind of state changes. The focus is on the modeling of autonomous intelligent behavior, which activates, supports and/or executes these state changes. This way a system is transformed from a given initial state to a desired target-state by the use of specific methods. Method-models specify those methods in detail.

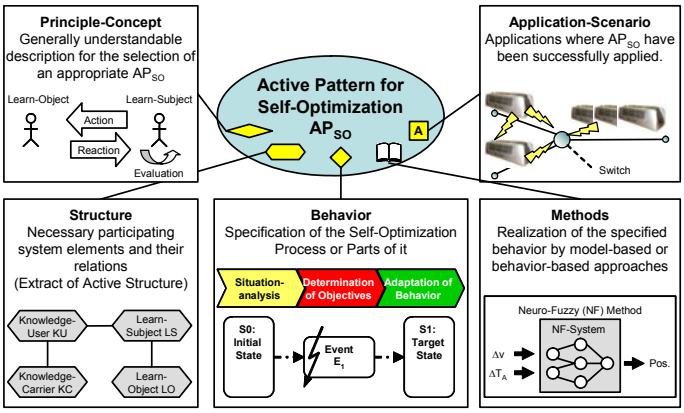

Fig. 4. Components of Active Patterns for Self-Optimization

We structure active patterns according to the *House of Active Pattern for Self-Optimization* (Fig. 5).

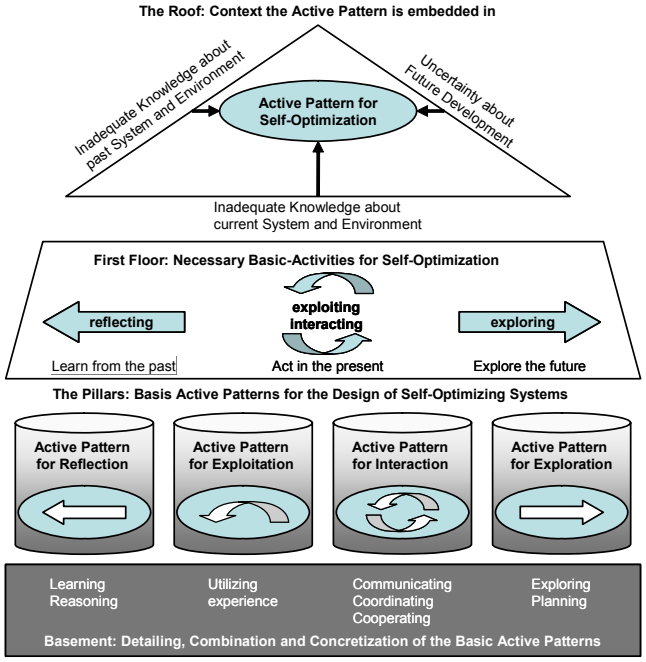

Fig. 5. The House of Active Patterns for Self-Optimization

Four basic active patterns can be differentiated which are derived from the pattern context and necessary activities for fulfilling the self-optimization process. The context may involve problem areas such as inadequate knowledge about past system and environment behavior, inadequate knowledge about the current system

and environment behavior or uncertainty about the future behavior. The context specifies demands for necessary basic-activities, such as learning from the past which we call *reflecting*, acting in the present which we denominate *exploiting* knowledge as well as *interacting* with other system elements and finally *exploring* the future. This approach leads to the four basic *Active Patterns for Reflection*, *Exploitation*, *Interaction* and *Exploration*.

The basic active patterns can be detailed, combined and concretized (Fig. 6). Detailing an active pattern means to specialize the pattern structure and pattern behavior according to the method which shall execute the system behavior, e. g. detail *Reflection* towards *Reinforcing Reflection* in order to use the method reinforcement learning [8] where successful past behavior is rewarded. Basic patterns can be combined to form typical compound behavior, e. g. the combination of *Exploitation* and *Reflection* leads to a typical compound behavior in a multiagent setting of exploiting the knowledge of distributed system elements to direct the learning behavior of the whole system [9]. Eventually, the pattern structure and pattern behavior needs to be concretized towards the active structure and finally to the construction and component structure of the system.

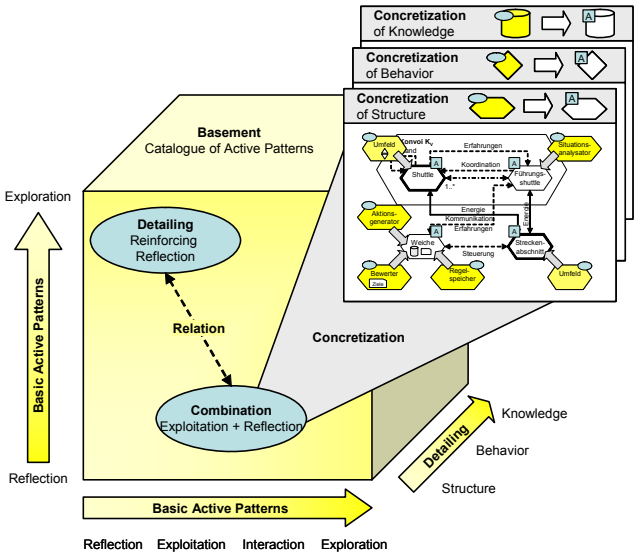

Fig. 6. Catalogue of Active Patterns

The catalogue of active patterns has been applied to several application scenarios of intelligent mechatronic systems, e. g. to shuttles driving on tracks by implementing the active patterns of *Exploration* and *Interaction* [10] or to the suspension-/tilt-module of a shuttle using the active patterns of *Interaction* and *Exploitation* [11]. The following chapter depicts the application of active patterns on the Networked Mechatronic System layer to design collaborative behavior of shuttles crossing a switch.

4 Validation – Collaborative Behavior of Shuttles Crossing a Switch

The application scenario of crossing a switch is as follows (Fig. 7): A Networked Mechatronic System of two convoys C_A and C_B, each consisting of several autonomous shuttles A_i und B_j, approach a switch. The passage of a single shuttle shall be designed such that the approaching convoys C_A and C_B are merged to a virtual convoy C_V. The shuttles shall optimize themselves autonomously and under restricted or no prior knowledge about an optimum behavior according to their own targets after each successfully completed crossing procedure. The whole scenario is split up into three zones – a decision, an execution and a learning zone.

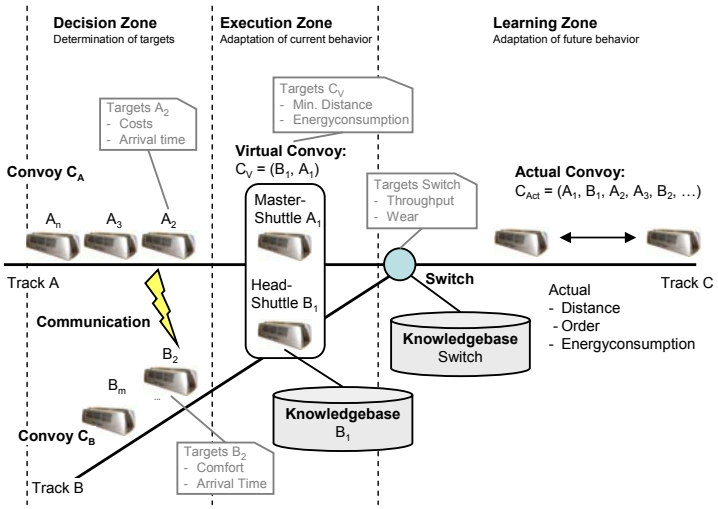

Fig. 7. The application-scenario of crossing a switch

In the course of the early conception stage, a function hierarchy is built up (Fig. 8). Let us illustrate the increasing concretization at the function of *Determination of Sequence* which shall define the passage sequence of shuttles. Based on prior design experience [11] this function can be realized by implementing the AP_{SO} *Exploitation*. The pattern structure of the AP_{SO} consists of two system-elements – the Knowledge-Carrier and the Knowledge-User. The pattern behavior can be specified by a statechart which specifies the adaptation process by a neuro-fuzzy method [12]. The AP_{SO} is concretized towards the active structure as follows. Every shuttle can be a Knowledge-Carrier because of its implicit experience about the determination of sequence generation with the help of neuro-fuzzy methods. The master-shuttle represents the Knowledge-User because it determines the passage-sequence for the remaining shuttles. Eventually, the adaptation of the behavior is detailed by a statechart which specifies possible adaptation processes for the generation of a virtual convoy C_v – here the adaptation process from an initial state S_0 – head-shuttles A_1 and B_1 right ahead of the switch – to the target-states $S_1 := C_v=(A_1,B_1)$

that is A_1 drives first, afterwards B_1 – as well as $S_2 := C_v=(B_1,A_1)$ that is B_1 drives first, then A_1.

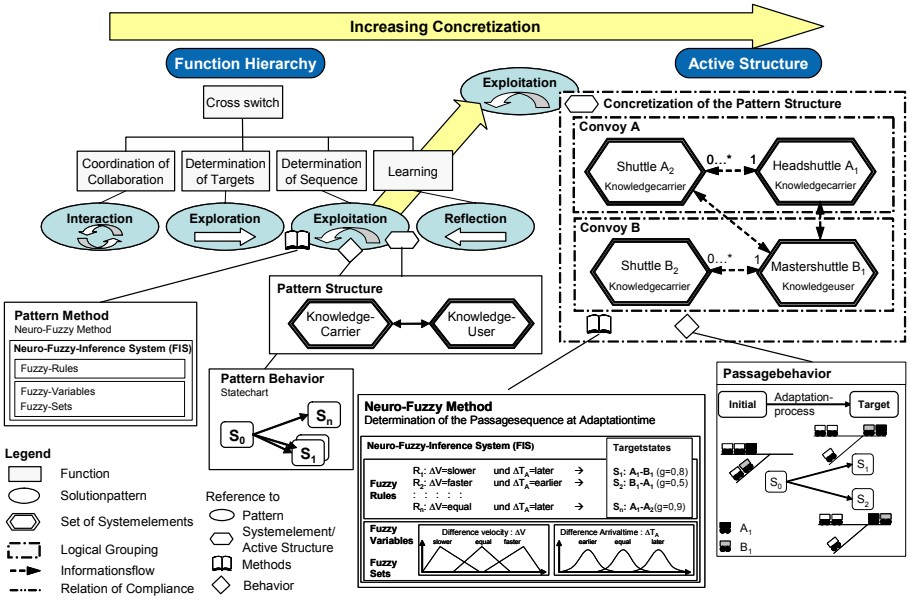

Fig. 8. Concretization in Design - Application Scenario of Crossing a Switch

In particular, the AP_{SO} *Interaction* specifies how the shuttles communicate with each other and determines the master-shuttle (Fig. 9). The AP_{SO} *Exploration* designs how target states such as S_1 or S_2 can be determined or newly created. A neuro-fuzzy system takes input-variables such as the velocity of shuttles $\Delta v = (v_{A1} - v_{B1})$ and arrival-time $\Delta T_A = (t_{A1} - t_{B1})$ at the switch to assign passage-classes such as $C_1 :=$ (A1-B1) = *Master-Shuttle drives first* and class $C_2 :=$ (B1-A1) = *Master-Shuttle drives second*. Because of the inherent uncertain and vague knowledge about environment- and system-states, fuzzy-variables are introduced, e. g. Δv = (slower, equal, faster) and ΔT_A = (earlier, equal, later). Fuzzy-rules realize the assignment of passage-classes, e. g. *If (Δv = slower and ΔT_A = later) Then (A1 – B1)*. The AP_{SO} *Exploitation* allows the system to start from initial knowledge and initial rules in system-state S_0 for setting up the neuro-fuzzy system. Once a target state such as S_1 is reached, AP_{SO} *Reflection* specifies, how the experience that was accumulated during the adaptation process can lead to adapted fuzzy-sets and new rules. This is done by evaluating the degree of fulfillment of committed targets such as minimum distance between shuttle Δd_{min} and maximum energy consumption E_{max} and consequently adapting the weights of the neural network of the neuro-fuzzy system leading to adapted fuzzy-sets and possibly to new fuzzy-rules.

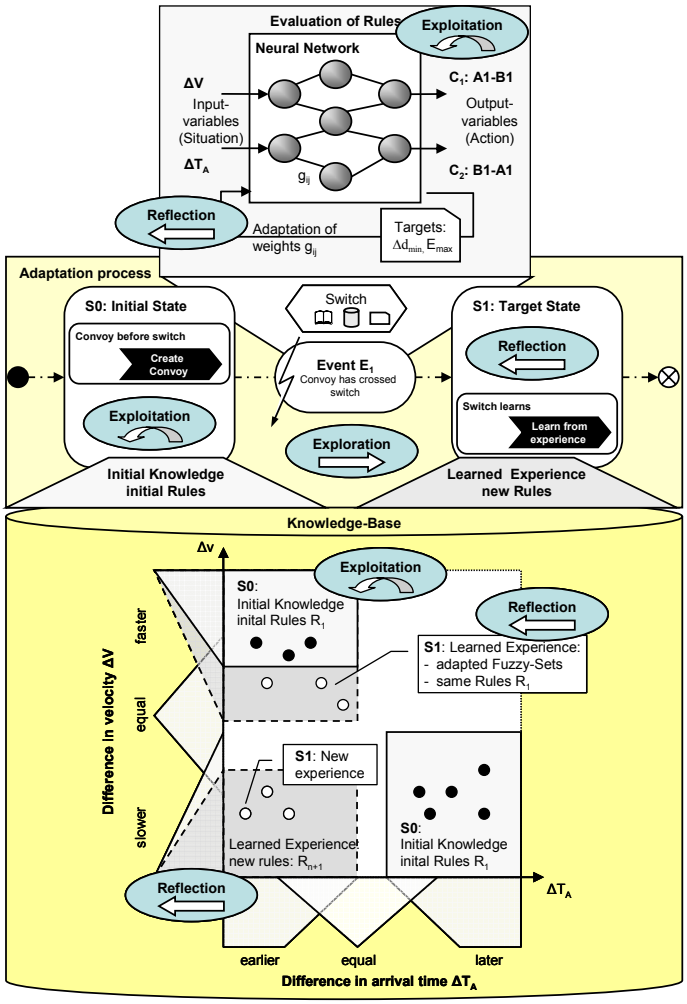

Fig. 9. Neuro-Fuzzy Learning with the exploitation of A-Priori Knowledge

5 Conclusion and future work

This article has proposed a schema called *Active Patterns for Self-Optimization* in order to design self-optimizing mechatronic systems in the early design stage. A set of four active patterns established the basis for the specification of a catalogue of patterns along the dimensions of detailing, combining and concretizing. The design approach was validated for a networked mechatronic system namely the crossing of a switch by convoys which consist of individually and autonomously acting shuttles. The pattern behavior was implemented according to the biologically inspired technique of neuro-fuzzy learning. Altogether it was shown, that active patterns for

self-optimization constitute an applicable approach for the design of intelligent mechatronic systems in the early design stages.

In order to cope with functional demands that arise from endogenous needs of agents as opposed to given external targets, future research will deal with the extension of active pattern schema towards cognitive behavior. Also, the pattern catalogue will be extended as new application scenarios of intelligent mechatronic systems demand a detailing and concretization of pattern structures and behavior.

6 References

1. Frank, U.; Giese, H.; Klein, F.; Oberschelp, O.; Schmidt, A.; Schulz, B.; Vöcking, H.; Witting, K.; Gausemeier, J. (Hrsg.): Selbstoptimierende Systeme des Maschinenbaus - Definitionen und Konzepte. HNI-Verlagsschriftenreihe Band 155, Paderborn, 2004
2. Lückel, J.; Hestermeyer, T.; Liu-Henke, X.: Generalization of the Cascade Principle in View of a Structured Form of Mechatronic Systems. IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2001), Villa Olmo ; Como, Italy, 2001
3. New Railway Technology Paderborn (NBP) – A Research Initiative for the Improvement of the Attraction of the Railway-System. http://nbp-www.upb.de, 2005
4. Daenzer, W. F.; Huber, F.: Systems Engineering – Methoden und Praxis. 8. verbesserte Auflage; Verlag Industrielle Organisation; Zürich, 1994
5. Pahl G.; Beitz W.; Feldhusen, J.: Engineering Design – A Systematic Approach. 3. ed., Springer-Verlag, Berlin 2006
6. Verein Deutscher Ingenieure (VDI): Design Methodology for Mechatronic Systems. VDI Guideline 2206, Beuth Verlag, Berlin, 2004
7. Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.: Design Patterns – Elements of Reusable Object-Oriented Software. Addison-Wesley, München, 1996
8. Berenji, N .R., Khedar, P.: Learning and Tuning Fuzzy Logic Controllers through Reinforcements. In: IEEE Trans. Neural Networks, No. 3, IEEE Press, Piscataway, NJ, USA 1992, pp. 724-740
9. Santamaría, J. C.: Learning Adaptive Reactive Agents. Doctoral Thesis, College of Computing, Georgia Institute of Technology, Atlanta, Georgia, 1997
10. Gausemeier, J.; Frank, U.; Giese, H.; Klein, F.; Schmidt, A.; Steffen, D.; Tichy, M.: A Design Methodology for Self-Optimizing Systems. In: Contributions to the 6th Braunschweig Conference of Automation, Assistance and Embedded Real Time Platforms for Transportation (AAET2005), Feb. 16th and 17th 2005, Technical University of Braunschweig, GZVB, 2005, Vol. II, pp. 456-479
11. Gausemeier, J.; Frank, U.; Schmidt, A.; Steffen, D.: Towards a Design Methodology for Self-Optimizing Systems. In: ElMaraghy, H.; ElMaraghy, W. (Hrsg.): Advances in Design, Springer Verlag, 2006, pp. 61-71
12. Koch, M; Kleinjohann, B.; Schmidt, A.; Scheideler, P.; Saskevic, A.; Gambuzza, A.; Oberschelp, O.; Hestermeyer, T.; Münch, E.: A Neuro-Fuzzy Approach for Self-Optimizing Concepts and Structures of Mechatronic Systems. In: Chu, H.-W.; Savoie, M.; Sanchez, B.: Proc. of the International Conference on Computing, Communications and Control Technologies (CCCT2004), Austin, USA, 14.- 17.08.2004, pp. 263-268