Determination of Active Pattern during the Conceptual Design of Self-Optimizing Systems demonstrated by an Air Gap Adjustment System

Jürgen Gausemeier, Herbert Podlogar, Jörg Donoth
Heinz Nixdorf Institute, University of Paderborn
Fürstenallee 11, 33102 Paderborn, Germany

Detmar Zimmer, Alexander Schmidt
Institute for Mechatronics and Design Engineering, University of Paderborn
Pohlweg 47-49, 33098 Paderborn, Germany

Abstract

The conceivable development of information 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 environmental conditions. They learn and optimize their behavior during operation. The design of self-optimizing systems is an interdisciplinary task. Mechanical, electrical, control, and software engineers are involved as well as experts from mathematical optimization and artificial intelligence. During the phase "conceptual design" developers have to choose solution patterns for functions from various domains. Different terminologies complicate the search for appropiate solution patterns, espacially in case of solution patterns used for self-optimization. In this contribution we introduce the active pattern for self-optimization as a new type of solution patterns. We present a domain spanning search for active patterns for self-optimization and demonstrate the search at a self-optimizing air gap adjustment system of a linear drive.

1. Introduction

Information technology is increasingly permeating the field of conventional mechanical engineering and bringing with it a considerable potential for innovation. Most modern mechanical engineering products already rely on the close interaction between mechanics, electronics, control engineering and software, which is known as "mechatronics". The aim of mechatronics is to improve the behavior of technical systems by using sensors to obtain information about the surroundings and the system itself. The processing of this information enables the system to react optimally to its current situation. Given the tremendous pace of development in information technology we can recognize further possibilities that go far beyond mechatronics: Systems with inherent intelligence. Therefore we use the term "self-optimization" [5]. Self-optimization enables mechanical engineering systems that have the ability to react autonomously and flexibly on changing operation conditions. The design of such systems is a challenge. Established development methodologies in the areas of classical mechanical engineering, e.g. the engineering design by PAHL/ BEITZ [8], and also methodologies of mechatronics, e.g. the VDI-guideline 2206 "Design methodology for mechatronic systems" [11], are not sufficient here. Within the Collaborative Research Centre (CRC) 614 "Self-Optimizing Systems and Structures in Mechanical Engineering" of the University of Paderborn, a set of specification techniques in order to describe the principle solution of self-optimizing systems has been worked out.

In the beginning, this contribution defines the paradigm of "self-optimization". It continues on analyzing the challenges of the development of self-optimizing systems and points out the usage of solution pattern and active pattern for self-optimization in the early phases of the development process. The center stage in this contribution takes the suggested search process for active pattern for self-optimization. Finally, the search process is exemplifyed on the task of implementing self-optimizing behavior to an air gap adjustment system during the phase "conceptual design".

2. Self-Optimizing Systems

The self-optimizing system determines its currently active objectives on basis of encountered influences. The self-optimizing system is able to adapt its objectives autonomously. This means, for instance, new objectives are

978-0-7695-3331-5/08 $25.00 © 2008 IEEE
DOI 10.1109/ICSEng.2008.60

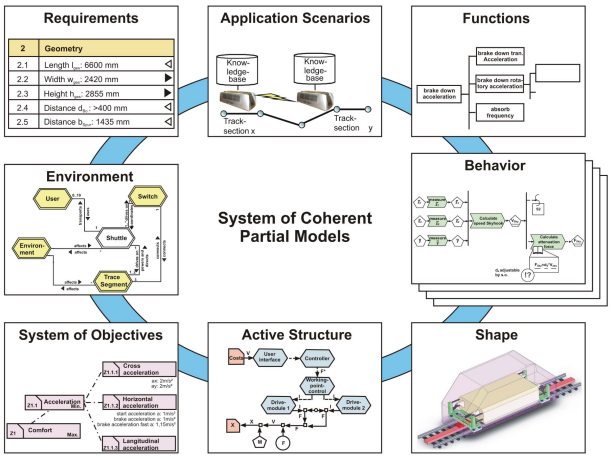

Figure 1. Partial models for the domain-spanning description of the principle solution

added or existing objectives are discarded and no longer pursued. Adapting the objectives in this way leads to an adjustment of the system behavior. This is achieved by adapting the parameters and where required the structure of the system. The term parameter adaptation means adapting a system parameter, for instance, changing a control parameter. Structure adaptations affect the arrangement of the system elements and thus their relationships.

Self-optimization takes place as a process that consists of the three following actions, called the *Self-Optimization Process* [5]:

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.

2. *Determining the system's objectives*: The system's objectives can be extracted from choice, adjustment and generation.

3. *Adapting the system's behavior*: The changed system of objectives demands an adaptation of the behavior of the system. This can be realized by adapting the parameters and by adapting the structure of the system.

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 adaptive behavior. Self-optimizing systems perform additionally to mechatronic systems functions like communicate information, share knowledge, use information, determine objectives, and change structures. Several combinations of these functions constitute self-optimizing processes.

2.1. Development of Self-Optimizing Systems

The development process for self-optimizing systems is devided into three phases: It starts with the domain-spanning conceptual design, followed by the domain-specific concretization and ends with the system integration. The result of the conceptual design phase is the principle solution. It describes the main physical and logical operating characteristics of the system in a domain-spanning way. On the basis of this jointly developed principle solution, further concretization will take place separately in the domains involved. Finally, during the system integration phase, the outcomes from the individual domains are integrated to form an overall system. Within the conceptual design phase, we use a set of semi-formal specification techniques to describe the principle solution of a self-optimizing system [4]. For a complete description several views on the self-optimizing system are needed. Each view is mapped by a computer onto a partial model. The principle solution is made up of the following views: requirements, environment, system of objectives, functions, active structure, shape, application scenarios and behavior. This last view is considered as a group because there are various types of behavior (e.g. the dynamic behavior of a multibody system, the cooperative behavior of system components etc.).

Of high importance are the interrelations between the partial models, which describe the coherence of the partial models. Those interrelations are built up between the constructs of the relating partial models (Figure 1). The partial models are not designed sequentially, but concurrently and interacting. Each development step undergoes numerous iterations, and the order, in which they are carried out, will depend upon the developed object, organizational constraints and on the preferred approach of the individual developer.

The function hierarchy is one milestone in the conceptual design phase. It primarily derives from the requirements. During the phase "conceptional design" solutions are established for each function (figure 2). These functions are represented by solution patterns [5]. The functions and their corresponding solution patterns may be conventional ones or those used for self-optimization.

2.2. Active Pattern for Self-Optimization

SCHMIDT [9] developed the concept of *active pattern for self-optimization* (AP_{SO}) as solution pattern, which facilitate developers to describe the self-optimization process. AP_{SO} fulfill self-optimization functions such as autonomous planning, cooperation, and learning. The scope of an AP_{SO} include the entire self-optimization process. The essential factor is that the system's state transitions

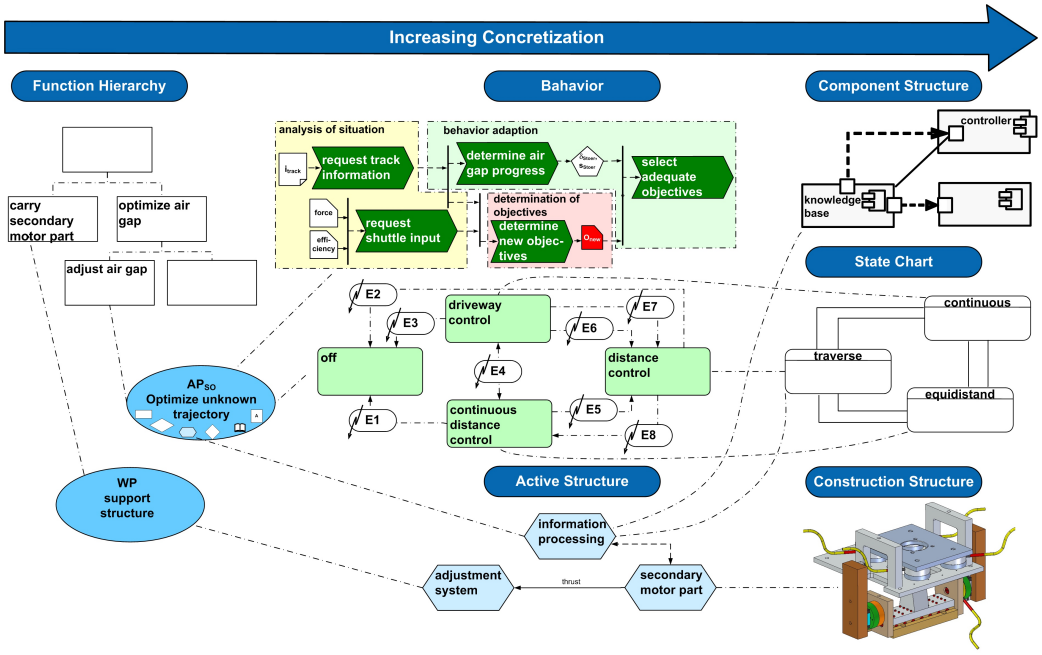

Figure 2. Developing self-optimizing systems

are initiated, supported, and effected by autonomous, intelligent behavior. AP_{SO} describe implementations of certain functions on such a level of abstraction, that they are generally applicable [5]. For the *phases situation analyses*, *determination of goals* and *adaption of the behavior* several functions of self-optimization like information obtaining, knowledge sharing, behavior coordination, information detection or optimization are implemented. Thus, an AP_{SO} describes not the implementation of a single function, but possible combinations of sub-functions which are integrated into the self-optimization process. The following paragraphs describe the overall structure of AP_{SO} and the information included. Figure 3 shows the aspects of an AP_{SO}: functions, principle concept, structure, behavior, methods, and application scenarios. Where suitable the AP_{SO} make use of the semiformal specification techniques developed by FRANK [4], which enables a domain-spanning specification of mechatronic systems. The aspects of an AP_{SO} are briefly described below:

The aspect *function* lists and describes the functions of self-optimization, which are implemented by the AP_{SO}. This aspect is necessary to integrate the AP_{SO} into the overall development process. With this aspect, the AP_{SO} are integrated in the development of principle solution after the design of partial model function.

The *principle concept* characterizes the underlying ideas behind the AP_{SO}. It enables developers to acquire an intuitive understanding of the AP_{SO} without any further formal details.

The aspect *structure* describes which system elements are fundamentally necessary in order to implement the AP_{SO}, and how those system elements are interrelated. The specification technique for describing the partial model active structure is used to describe the structure.

The aspect *behavior* describes the self-optimization process. Thus, we need to model the autonomous, intelligent behavior that initiates, supports and/or effects state transitions. The behavior aspect is split in two sub aspects. According to the corresponding partial model these sub aspects are *behavior state* and *behavior activity*. The behavior state aspect models possible sequences of states and state transitions. The behavior state aspect models the activities that are performed by system elements during the self-optimization process. Interactions between system elements are modeled in the behavior sequence aspect, using sequence diagrams.

Methods serve to implement the self-optimization processes, in particular to adapt objectives and behaviors (as a result of adaptations to parameters and possibly also structures). The aspect methods is a set or catalogue of methods which can be used to implement the self-optimizing process. Examples of such methods are fuzzy neural rule switching for adapting internal objectives, and case-based planning for experience-based behavior adaptation.

The aspect *application scenario* describes successful established applications of AP_{SO}. An application scenario encompasses the same aspects as the AP_{SO}. The aspects of application scenario are instances or concretization of the

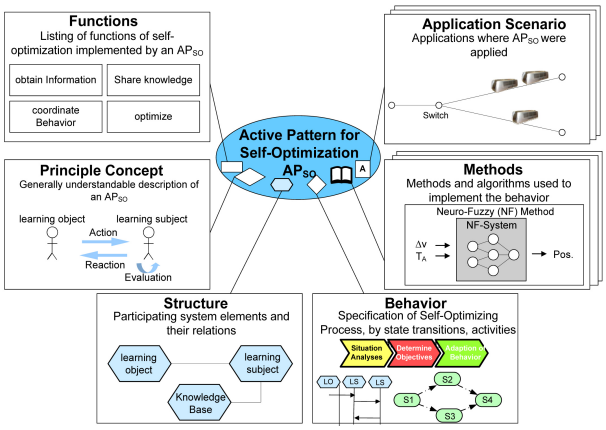

Figure 3. AP$_{SO}$: Description by partial models

aspects of the corresponding AP$_{SO}$.

AP$_{SO}$ are used when functions of self-optimizations occur in the partial model function hierarchy. For one or more functions of self-optimization a suitable AP$_{SO}$ has to be found and the product developer has to select an AP$_{SO}$ based on his experience. The application of an AP$_{SO}$ is demonstrated in the next section.

2.3. A Knowledge Base for Active Pattern Reuse

Once an AP$_{SO}$ has been recognized and used in a successful application, it is desirable to retain the information in order to enable the reuse in different applications as well. Apparently it is necessary to use a database, which stores the AP$_{SO}$ in such a manner, that engineers are able to recognize the required AP$_{SO}$. This approach demands a systematic towards the form, in which AP$_{SO}$ shall be stored. A convenient knowledge management systematic was developed by SCHMIDT in [9]. SCHMIDT introduced a scheme, which contains main characteristics of an AP$_{SO}$. For the storage a knowledge management system is proposed.

3. Problem Statement

Fulltext search is a mandatory feature in information retrieval applications such as a knowledge base for active pattern for self-opimization. Most users are familiar with this search method and it is usually their first choice to obtain information. Unfortunately, pure fulltext search is hardly helpful in the context of self-optimizing mechatronic systems. The reason is the involvement of various domains in the design process. Experts from mechanical, electrical, and software engineering contribute to the development process as well as experts for optimization and artificial intelligence. Each of these domains and the possible application domains – such as automotive, aerospace, or lokomotive – possess their own terminology.

Thus, a fulltext search with domain specific terminology will necessarily miss a considerable amout of relevant information stored in the active pattern knowledge base. DAVENPORT summarized this problem in one sentence: "People can't share knowledge, if they don't speak the same language" [3]. Even worse, application oriented users will only be able to identify active patterns already applied to their domain. But especially the transfer from other domains should be supported to facilitate innovations.

We suggest a more domain-spanning search instead of a pure text-based search. For the sake of simplicity and usability the search should still appear like the well known fulltext searches. The search process itself shall be transparent to the user. A domain-spanning search must take domain specific terms into account. It must be able to identify relevant information in the knowledge base, even if it is described in the terminoloy of an entirely different domain. Here, a translation between the domain-specific terminologies is required.

4. Domain Spanning Search

At first this section explain how ontologies can be used for translations between terminologies of different domains. Subsequently, we sketch the domain spanning search process. Two seperate processes can be identified regarding the domain spanning search. The first phase provides the information required for the actual search process. We will refer to it as the maintenance phase. The second process is the actual search process.

4.1. Mapping of Ontologies

Ontologies seem a promising approach to establish bridges between the terminologies of different domains. Ontologies provide theories about the kinds of objects, properties of objects and relation between objects that are relevant for a specific domain of knowlegde [2]. In their application to information systems, an ontology is regarded as *designed artefact* consisting of a specific *shared vocabulary* used to describe entities in the domain as well as a set of assumptions about the intended meaning of the terms in the vocabulary [6]. A formal definition of ontologies can be found in [7]. The formal definition of ontologies is important for automated reasoning in information systems. Domain ontologies represent the conceptualization and a framework for knowledge bases of certain domains (e.g. rail traffic, medical science).

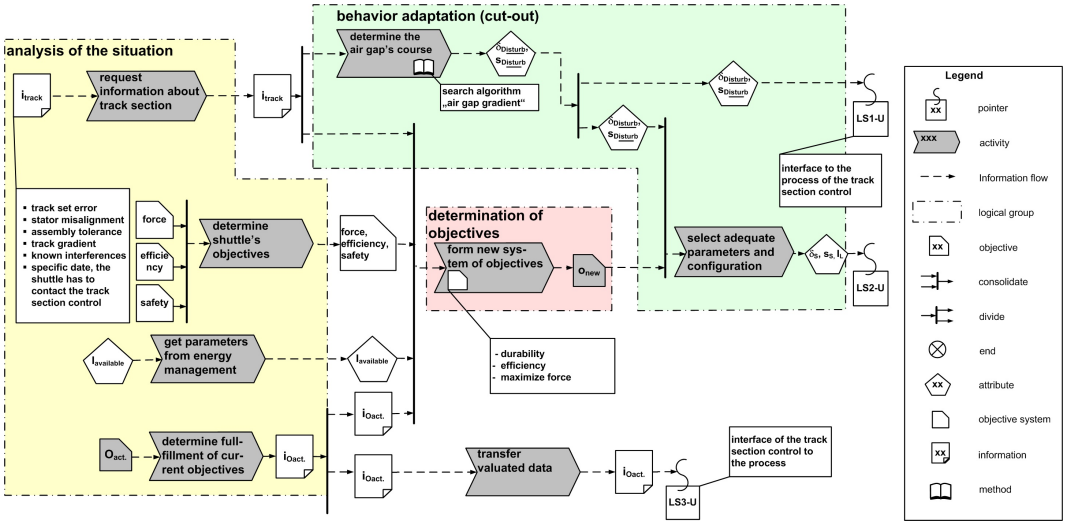

Figure 4. Application scenario: Behavior – activity

4.2. Maintenance Phase

In the maintenance phase the domain ontologies are provided to the knowlegde base and matched against each other. The definition of the domain ontologies is done in Protege. They are stored with the knowlegde base. New ontologies can be introduced anytime to the knowlegde base. The new ontology is matched against every ontology that is already in the knowledge base. Result of a matching process is – as described above – a number of matching classes from the two ontologies with rating denoting the reliabilty of the match. From this matching result a dictionary is created. A dictionary simply contains tuples for each class in an ontology with the class name, a related class in a target ontology and the reliability rating of the relation. There will be two dictionaries for each pair of ontologies, one for each translation direction.

4.3. Search Process

The user enters this input using the terminology related to his professional background. In order to identify relevant active patterns with information entered by experts from other domains, the search input has to be translated into the other domain terminology. The dictionaries generated in the maintenance phase are used to translate the search input.

In this paper we will restrict the search input to a set of words. The structure of the user input can be of course more complex, e.g. include phrases, field-restriction and boolean operators. The structure only depends on the used fulltext search engine and has no implication to the principle of domain spanning search. The general idea can be arbitrarily refined to exploit the features of a given search engine.

The words provided by the user can be translated into the other terminolgies by looking up the corresponding dictonaries. With the rating provided in the dictonaries, the quality of the translation can be estimated. Result of the translation process is a set of input strings, which can be used to invoke the search algorithm several times. Result of every invocation is a number of active pattern. The relevance of the active pattern can be calculated by a cummulative model: $\sum i \times rating_i$, where i is one, if the active pattern was found in search run i, and $rating_i$ is the reliability of the search string i. The identified active pattern are presented to the user ordered by their relevance.

5. Application Example

The research initiative "Neue Bahntechnik Paderborn (NBP)", founded at the University of Paderborn in 1997, deals with the development of an innovative railbound transport system for freight and passengers. Autonomous vehicles travel to their destination without any changeovers are the essential elements of the new transport system. The vehicles are no longer propelled via the wheel-rail-contact, but by a contact free dual-fed linear drive [1].

5.1. Air Gap Adjustment Systems

The primary motor part (PMP) of the linear drive is installed between the rails and mounted to the ties. Its secondary motor part (SMP) is fixed to the vehicles. The primary motor parts are combined to sections in order to reduce copper losses. Each section is separately powered from power supply units, which are distributed along the track. The whole track possesses four units. The SMPs

are powered by onboard batteries. The NBP is able to omit overhead lines or conductor rails because the double fed linear drive is able to transmit energy from the PMPs to the SMP. The propulsion force is generated in the air gap between both parts.

However, the innovative drive concept features a lower efficiency than conventional rotating drives with comparable installed engine power [10]. On the one side, the lower efficiency is based on the smaller covering-over of PMPs and SMP. The other reason is the wide air gap of the linear drive. This relatively wide air gap compared to conventional drives is necessary for the robust operation of the drive, which is influenced by production and assembling tolerances of the PMPs, the wheels, and the rails. The sum of the tolerances is not constant along the track which leads to a varying air gap. Additionally, temporal influences and thermal influences change the air gap along the track as well.

These circumstances motivated the development of a self-optimizing air gap adjustment system (AGAS) within the CRC 614. The target is the development of a system which is able to adjust the air gap to a situational optimum. The neccessary degree of freedom is implemented by additional actuators below the vehilce. They move the SMP vertically. Apart from a mechanical mechanism, an intelligent process is needed, which has to handle various situations: In situations such as *driving on an inclining track section* or the *start up within a station* the drive has to provide a maximal propulsion force. Therefore, the AGAS has to adjust the SMP to a minimal air gap. Less energy within the vehicle leads to a higher priority of the objective *maximize efficiency*. This means the AGAS has to minimize the air gap but also minimize the energy needed for the adjustment. The amount of the required energy for the adjustment has to be smaller than the saved up energy for the driving force. Travelling on unknown track sections, leads to an uncertainty of the stators' track displacement. Furthermore, it leads to a higher weighing of the objective *maximize safety*. Figure 4 illustrates the behavior aspect of the scenario.

5.2. Search for AP_{SO}

In order to meet these requirements, the user needs an AP_{SO} that enables the system to react autonomously to changing condition. Thereby, it has to calculate an optimal strategy for each recognized situation. The user enters several text phrases to the search engine, which characterize the application, e.g. *maximize safety*, *maximize efficiency*, and *air gap adjustment*. Using ontologies and automated mapping, the search process translate the terms to different terminologies. Comparing the translations with available active patterns the search suggest several active pattern as favorits. The developer can examine these active pattern and choose one for the considered function of self-optimization.

6. Conclusion

The design of self-optimizing systems is a challenge and involves various domains. We introduced active patterns for self-optimization. This new type of solution patterns helps to achieve a uniform formulation of proven solutions and enables developers a successful reuse, even if the specified method was never applied to the domain before. We introduced a concept of a domain spanning search process, driven by domain ontologies, automated mapping and translation. With the search, the identification of a suitable active pattern with terms of a specific domain is possible, even if the active pattern was never applied to the domain before.

Acknowledgment
This contribution was developed in the course of the Collaborative Research Centre 614 "Self-Optimizing Concepts and Structures in Mechanical Engineering" funded by the German Research Foundation (DFG).

References

[1] J. Böcker, A. Schmidt, B. Schulz, and D. Zimmer. Direktantriebe passend ausgewählt – Elektromagnetische Direktantriebe im Vergleich. *Antriebstechnik*, Nr. 2 (spezial):2–6, Februar 2005.

[2] B. Chandrasekaran, J. R. Josephson, and V. Benjamins. What are ontologies, and why do we need them? *EEE Intelligent Systems*, 14:29–26, 1999.

[3] T. Davenport and L. Prusak. *Working Knowledge: How Organizations manage what they know*. Harvard Business School Press, 1998.

[4] U. Frank. *Spezifikationstechnik zur Beschreibung der Prinziplösung selbstoptimierender Systeme*. PhD thesis, University of Paderborn, 2006.

[5] U. Frank, H. Giese, F. Klein, O. Oberschelp, A. Schmidt, B. Schulz, H. Vöcking, and K. Witting. *Selbstoptimierende Systeme des Maschinenbaus Definitionen und Konzepte*. Band 155. HNI-Verlagsschriftenreihe, 2004.

[6] N. Guarino. Formal ontology in information systems. In *Proceedings of FOIS'98, Formal Ontology in Information Systems*, 1998.

[7] A. Maedche. *Ontoloogy Learning for the Semantic Web*. Kuwer Academic Publishers, 2003.

[8] G. Pahl, W. Beitz, J. Feldhusen, and K.-H. Grote. *Enginieering Design - A Systematic Approach*. Springer Verlag, Berlin, 2007.

[9] A. Schmidt. *Wirkmuster zur Selbstoptimierung – Konstrukte für den Entwurf selbstoptimierender Systeme*. PhD thesis, University of Paderborn, 2006.

[10] A. Schmidt and D. Zimmer. Linear läuft es schneller – Der Luftspalt bei Linearmotor-getriebenen Schienenfahrzeugen. *Antriebstechnik*, Nr. 2 (spezial):6–10, Februar 2005.

[11] Verein Deutscher Ingenieure (VDI). Design methodology for mechatronic systems. VDI-Richtlinie 2206, Beuth Verlag, Berlin, 2004.