INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'07
28 - 31 AUGUST 2007, CITE DES SCIENCES ET DE L'INDUSTRIE, PARIS, FRANCE

USING EVOLUTIONARY ALGORITHMS IN THE CONCEPTUAL DESIGN OF SELFOPTIMIZING SYSTEMS

Dr.-Ing. R. Radkowski[1], Prof. Dr.-Ing. J. Gausemeier[1]
[1]Heinz Nixdorf Institute, University of Paderborn, Fürstenallee 11, 33102 Paderborn

ABSTRACT

The machines of tomorrow will be capable of self-optimization. This means that they are able to adjust to changing surroundings and conditions on their own. New systems require new developmental methods and strategies. The approach, which is presented here, is based on evolutionary algorithms in order to design the principle solution of a self-optimizing system. The evolutionary algorithm produces the active structure automatically. The partial model shape, behavior and function can be derived from it. The representation for these self-optimizing systems and the approach, how to design a concept with evolutionary algorithms is, what is described in this article. The functionality of this method will be verified by an example.

This contribution was developed in the course of the Collaborative Research Centre 614 "Self-Optimizing Concepts and Structures in Mechanical Engineering" (Speaker: Prof. Gausemeier) funded by the German Research Foundation (DFG) under grant number SFB 614 (see www.sfb614.de for further details).

Keywords: Self-optimizing systems, evolutionary algorithms, development methodology

1 INTRODUCTION

Products of mechanical engineering and related branches are more and more based on a close cooperation of mechanics, electronics, control engineering, and software engineering. It is the term mechatronics, which expresses this. Mechatronics optimizes the behavior of technical systems: Sensors gather information about their surroundings, which are then digitally processed and then actuators generate an optimal reaction for these informations. Future technical systems will consist of system-components with an inherent part-intelligence. The behavior of the whole system will be affected by the communication and interaction of their intelligent components. This opens up a new perspective for mechanical engineering. The term "self-optimizing" characterizes this new perspective significantly. A special challenge comes from the fact, that it is no longer possible to anticipate all possible system states.
Extensive instruments for the development of such systems are being developed in the context of the Collaborative Research Centre 614 " Self-optimizing systems in mechanical engineering". Two approaches are being followed in the Collaborative Research Centre. One approach follows the construction methods by PAHL/BEITZ [1], [2], [3]. The second approach is based on the use of an evolutionary algorithm (EA). This is the subject of the article at hand. Central to both approaches is the so-called principle solution, which is the result of the conceptual design. The principle solution defines the physical and logical mode of operation of the system, as well as the structure of the system elements. In our approach, the principle solution is build by an evolutionary algorithm.
Evolutionary algorithms were already used in a number of other projects for the development of technical systems. Hence, we can only mention a few of these projects at this place. A classification of already existing work on the topic has shown that EA's are mainly used for the development and optimization of two aspects of technical systems: Namely for developing and optimizing the design of a technical system [4],[5],[6] and for developing and optimizing the motion of a technical system

[7],[8]. VANJA has already found that the use of evolutionary algorithms for a developmental process is not opposed to the prevalent design theory [5], because developers also make use of evolutionary approaches. Impressive results have been presented by Lipson, Polack and others, too. They developed methods for autonomous development of static structures (Legobots) and dynamic robots. The design and the controller of the robot are built automatically, and they are able to manufacture themselves (partly). [9], [10]

Most of the related work, which takes into account real systems of mechanical engineering or mechatronic systems, uses evolutionary algorithms to optimize these systems. To optimize means for example: Found the right shape and diameter for a tube to maximize the flow rate. Or find the best parameters for a controller to minimize the reaction time of the system.

Our aim is to support the engineers in the conceptual design during develop of the principle solution. The presented approach should support the engineers with new design ideas or show possible solutions. This principle solution is not concrete enough, that one ore more single parameter of a special aspect should be optimized. In the Collaborative Research Centre 614, a specification technique for the principle solution has been developed. It is necessary to take account of a total of eight aspects to specify a self-optimizing system: Requirements, environment, functions, scenarios, shape, system of objectives, active structure, and behavior. The presented approach can support the design of self-optimizing systems, because the needed partial models are take into account. Four of these eight aspects will be taken into account in this article: functions, active structure, shape, and behavior.

Furthermore, we use solution elements to describe the single components of a system. A solution element is a well-known and tested component, which you can buy or which has been developed in a earlier project. For example, it can be an engine or a microprocessor. Every solution element is described by the eight mentioned aspects. However, other projects, which try to found ideas for products, etc. only take one or two aspects into account: The mechanical design (structre) or/and the movement.

The structure of the article is as follows: The concept of self-optimizing systems will be introduced in the upcoming paragraph. The computer-internal representation will be illustrated in the next paragraph. The approach to a design with evolutionary algorithms will be presented in chapter four. Chapter five consists of an example. The article closes with an outlook on future development.

2 SELF-OPTIMIZING SYSTEMS

Future systems in mechanical engineering consist of configurations of intelligent system-components. The behavior of the whole system will be affected by the communication and cooperation of these intelligent system-components. On the software side, these are distributed systems of cooperating agents, by which a self-optimizing behavior is possible.

The term self-optimizing is defined as follows:

> *„Self-optimization of a technical system is defined as the endogenous adjustment of the aims of the system to changed influences and the resultant autonomous purposive adaption of the parameters and, if necessary, the structure and thus of the behavior of the system. So self-optimization goes well beyond the known rule- and adaptation strategies; Self-optimization enables systems with inherent intelligence, that are able to take action and which are capable of reacting independently on changing conditions.“*

The following characteristics can be derived: Self-optimizing systems are capable of recognizing influences on the system. These are influences originating from the surroundings of the system (environment, user, etc.) or the system itself. The self-optimizing system determines its aims according to the recognized influences. This means for example, that the emphasis of the aims can be altered, new aims can be added or existing aims can be neglected. The adjustment of the aims redefines the system-behavior. The adjustment itself happens via altering the parameters of the system and if this is not sufficient, via altering the structure. Altering the parameters means, that one or more numerical values change, e.g. the parameters of a controller. Structural adjustment means that the system alters the current configuration of the system components and their relation among each other. With this, there is a distinction between reconfiguration, which means that the relations between the system components change, and adaptation, which means that new system components are integrated into the system.

Self-optimization itself is understood as a process, which consists of three steps: “Analysis of the current situation”, “determination of the system objectives”, and “adaptation of the system behavior”. These three steps of the self-optimization process will be explicated in the following: By “analysis of the current situation”, the current situation includes the state of the system itself and all the observa-

tions that have been made about its environment. Such observations may also have been made indirectly by communicating with other systems. By "determination of the system objectives", the current objectives of the self-optimizing system have been determined anew. At least "adaptation of the system behavior" leads to the new objectives.
The specific feature of self-optimizing systems is the endogenous alteration of objectives and based on this, the alteration of behavior. A state transition occurs in the context of the self-optimizing process. [2]

3 REPRESENTATION OF SELF-OPTIMIZING SYSTEMS

In order to describe the eight aspects or views of the principal solution of a self-optimizing system, a set of specification techniques will be used. Every view is represented by a partial model on the computer. The principal solution is described through eight partial models. The partial models are interconnected and can affect each other. These relations result in an integrated system of partial models, by which the principal solution is represented on the computer [3].
For a design using evolutionary algorithms, the representation of the principal solution is split up into two sections (Figure 1): One representation is automatically generated by the evolutionary algorithm. This is, what is called the active structure. The other representation shows the objectives of the design and is used in order to evaluate it. In this case, they are the three partial models "shape", "function", and "behavior", in which the "behavior" is modelled by a multi-body system. The representation, which is generated by the evolutionary algorithm, is called genotypic representation (genotype) (The terms genotype and phenotype are used to describe this two view, because to get the phenotype, a decoding of the genotypes formal description is necessary) [10]. The representation, which represents the objective of the draft, is shown as phenotype in Figure 1.

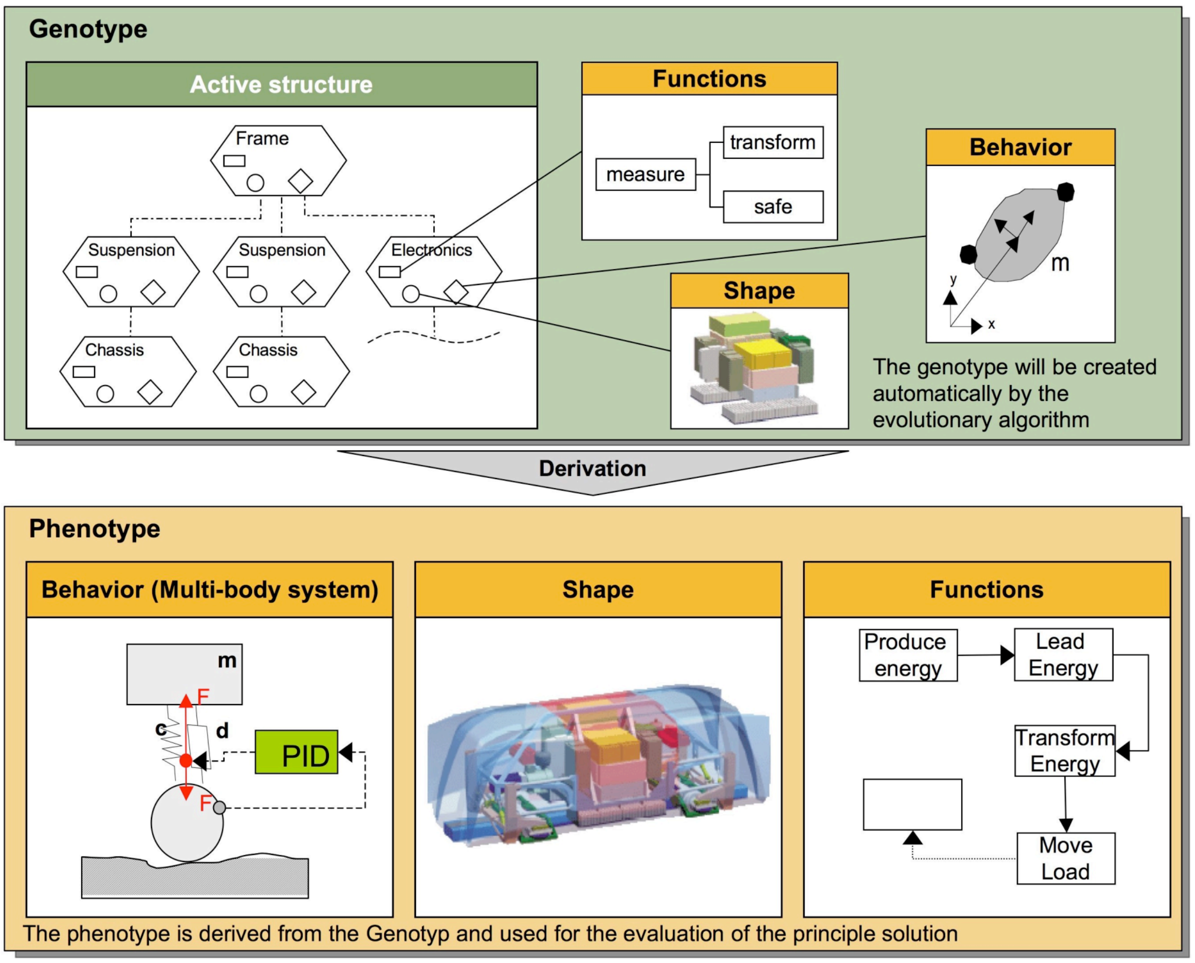


Figure 1: Derivation of the behavioral model (multi-body system), the building structure and the functions from the active structure

The active structure is automatically generated by the evolutionary algorithm. Methods of natural evolution are used in order to achieve this. The three partial models “shape”, “function”, and “behaviour” are derived from the active structure. This derivation also happens automatically. The results are four partial models of the principal solution. In the following, when mentioning the principal solution, exactly the four models “active structure”, “shape”, “function”, and “behaviour” are meant. Only some of these models need to be modelled, in order to generate the principal solution, but in this way, the comprehensibility of the text shall be increased.
The genotypic and phenotypic representation will be introduced in the following. Further, the way of deriving the phenotype from the genotype will be discussed.

3.1 Genotypical representation

As mentioned before, the active structure is used as a genotypic representation. The active structure is assembled from system elements and their relations among each other. These relations form flows of information, matter, and energy. At the beginning of the design process, the system elements are still abstract and act as placeholders for the software components or the assembly group. They are substantiated in the course of the developmental process. In order to model relations between the system elements, every system element is provided with an interface. The values of the system elements (e.g. forces, breaker signals…) are transmitted through this interface. Four attributes are used in order to provide a more detailed description of the interfaces: These are the class of the interface (information, energy, or flow of material), the type of the interface (e.g. mechanical), the unit of the interface (e.g. Newton), and two boundary values (min / max) The interfaces can permit for directed and undirected connections and they can also serve as entries and exits. The computer-internal representation of the system elements contains references to partial models. The partial models describe the system elements more accurately (Figure 1). These are the partial models of “shape”, “function”, and “behaviour”.

3.2 Phenotypical representation

The partial models “shape”, “function”, and “behaviour” of the whole principal solution represents the phenotype. The partial model “shape” contains information about the rough shape of the principle solution and of the active surface, as well as, information about the position of active points. Active Points indicate the interface of a logical function. The functions describe the basic functionality of the system. Therefore, *special functions* as defined by ROTH are used [12]. In order to describe the “behavior”, which in this case is the movement, a multi-body system and a block diagram are used. The block diagram describes a controller.

3.3 Derivation of the phenotype from the genotype

The phenotypic representation can be derived from the genotypic representation: Every system element is specified by the three partial models “shape”, “function”, and “behavior”. The system elements are connected by their relations among each other. Because of these interconnections, relations between the partial models can be derived. In this way, the two associated partial models and the two similar associated system elements are connected respectively. This applies to the whole active structure. This is how an assembly structure is derived from the singular shape models, as well as, a function model from the singular functions and a multi-body system for the whole principal solution from the singular multi-body systems.

4 EVOLUTIONARY DESIGN

The evolutionary design is split up into two phases (Figure 2), the pre-process and the runtime-process. During the pre-process, the requirements to the principal solution and its behaviour of movement are calculated. The requirements describe the objective of the design. The automatic generation of the four partial models, namely the active structure, “shape”, “function”, and “behaviour” follow. The runtime-process can for instance operate during the running time of the self-optimizing system: The self-optimizing system can evaluate which system elements are adequate in order to meet the requirements of the functions and it can then adjust itself according to these requirements.
Next up, this procedure will be introduced, followed by the evolutionary algorithm which is used for the design.

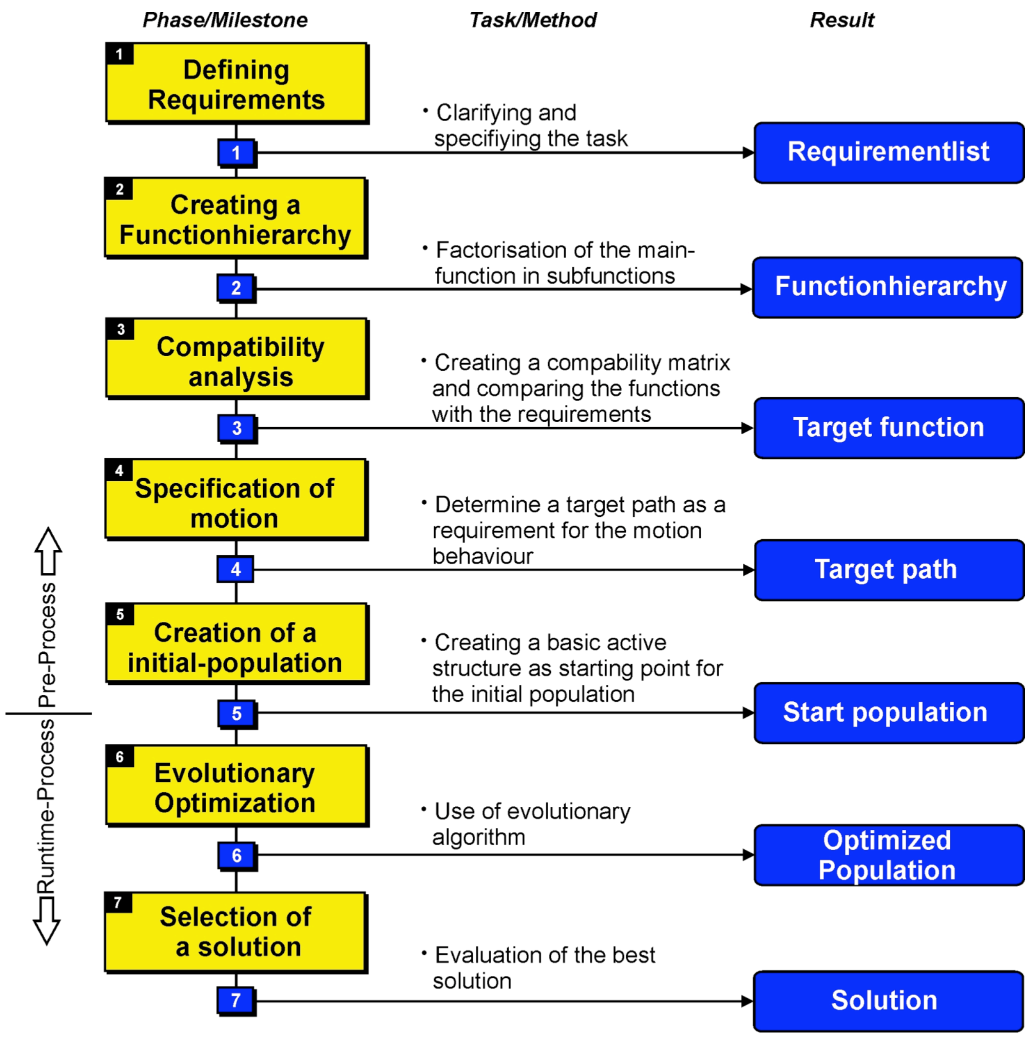
4.1 Procedure

The procedure consists of seven steps, as shown in figure 2. The pre-process consists of the five following steps: Definition of requirements, arrangement of a function-hierarchy, analysis of compatibility, definition of the movement, and generation of an initial population. The post-process consists of two steps: Evolutionary optimization and choice of a principal solution. The steps are described more closely in the following. However, the first two steps will be skipped because they have been standardized by PAHL/BEITZ.

The compatibility analysis is carried out with the help of a compatibility-matrix. The requirements are listed in the columns of the matrix, whereas the functions are listed in the rows. The fields of the matrix evaluate how good or how bad the functions meet the requirements. The scale goes from 1 (=completely incompatible) to 5 (=consistent). The compatibility analysis provides a relation between functions and requirements.

In the next step, the pre-defined movement is specified by a path. Hereby, two different types of paths are distinguished: The first path is the so-called system path. The principal solution has to be able to follow this path. The second path is the so-called load-path. This path describes the way of a load. The designed system has to be able to transport a load along this path.

In the next step, an initial population is generated. This population consists of a limited number of active structures. The initial population is generated at random. The origin of it is an initial system. This is the first active structure. It is then multiplied and single system-elements are replaced randomly. The system can either be defined by the user or also be generated at random. If the user already has a good idea about the system, its active structure can be specified. However, if there is no basic idea, the initial system is generated randomly.



Phase/Milestone
Task/Method
Result
1 Defining Requirements
• Clarifying and specifiying the task
1
Requirementlist
2 Creating a Functionhierarchy
• Factorisation of the main-function in subfunctions
2
Functionhierarchy
3 Compatibility analysis
• Creating a compability matrix and comparing the functions with the requirements
3
Target function
4 Specification of motion
• Determine a target path as a requirement for the motion behaviour
4
Target path
5 Creation of a initial-population
• Creating a basic active structure as starting point for the initial population
5
Start population
6 Evolutionary Optimization
• Use of evolutionary algorithm
6
Optimized Population
7 Selection of a solution
• Evaluation of the best solution
7
Solution
Pre-Process
Runtime-Process

Figure 2: Methodology for the evolutionary design

During the runtime-process, the initial population is optimized by an evolutionary algorithm. The single steps of the algorithm are described in the following. This optimization process can for instance happen during the running time of the self-optimizing system. In this way, the steps one to five can be processed during the conceptualization. During the running time, the system can evaluate whether, for instance another configuration of the system elements allows for a better adaptation to the environment. However the system can not reconstruct itself, but it can recommend reconstruction during maintenance. Because the optimization-algorithm allows for more than one potential solution, it is up to the user, to decide, which solution to choose.

4.2 Evolutionary Algorithm, Operators, and Evaluation

The evolutionary algorithm optimizes the initial population, which was generated during the fifth step. It is based on the kind of genetic programming, which was developed by KOZA [13]. The operators however were altered for this application.
The workflow of the algorithm, as well as the operators and the evaluation function will be introduced in the following.

4.2.1 Workflow of the algorithm

Figure 3 describes the workflow of the algorithm schematically. This shows step number 6, as displayed in Figure 2. The initial population serves as starting point for the optimization process. During the first step, the principal solutions are evaluated. Therefore the phenotypical representation is derived from the genotypical representation. A value of fitness is calculated for every phenotype (s. 4.2.3). This is a numerical value, which describes how good or how bad the principal solution matches the requirements. Then the recombination follows. Two new active structures are generated from two active structures of the current population respectively. Then the active structures of the populations mutate. In order to achieve this, singular elements of the system are exchanged. Next up is the evaluation of the population where, again, the fitness value for the population is calculated. Then comes the selection: Promising active structures – those with a high fitness value – remain in the population, whereas bad ones are deleted. The method, which is called "ranged selection", is used to achieve this goal [14].

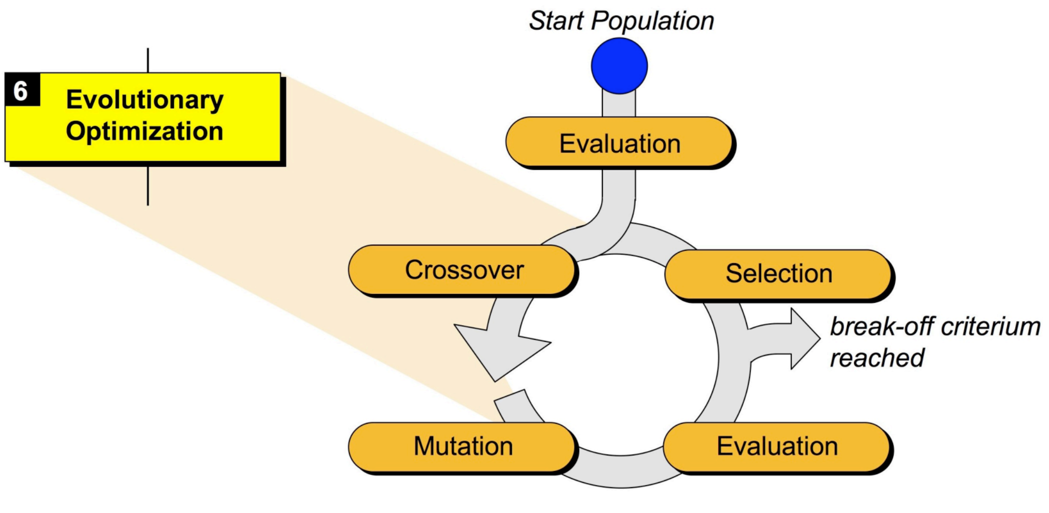


Figure 3: Course of the evolutionary algorithm

Finally the cycle starts over again. The algorithm continues to run until one of the stop criteria is met.

4.2.2 Evolutionary operators

Two evolutionary operators are used: The recombination-operator and the mutation-operator.
The recombination-operator generates a new active structure from two existing active structures. In order to achieve this, two active structures are chosen from the population. One active structure is split up at random connection points. (flow of matter, energy or material). The second active structure is split up in a way, so that the number of separated connections agrees with the number the separated connections of the first active structure. In this way, there are always four halves of active structures. Next, any of the halves is connected, each which a half of the opposing active structure. The muta-

tions-operator alters an active structure by exchanging some system elements at random and integrating new ones in the active structure. Connections between the system elements are deleted or new connections between two system elements are generated. Which one of these two operators is used, is also left to chance.

4.2.3 Evaluation

During the phase of evaluation, any principal solution of the population is assigned a value of fitness. The evaluation happens according to the three partial models "shape", "function" and "behaviour". The analysis of these three models results in a fitness value. Different aspects are revised in order to achieve this. A separate value of fitness is calculated for each partial model (f_1, f_2, f_3). These three values add up to an overall fitness value f_{ges}:

$$f_{ges} = g_1 f_1 + g_2 f_2 + g_3 f_3 \qquad (1)$$

The weighing of the singular partial models can be altered by altering any of the three parameters g_1 to g_3. This is more closely described in the following:

- **Evaluation of the multi-body system:** The multi-body system is simulated and the path of movement of the system is compared to the user's predefined path. In order to achieve this goal, the supporting points on the proposed path are compared to the respective supporting points on the user defined path. The more pairs of supporting points are equal or close to equal, the higher the fitness value. Furthermore, the existence of control loops will be analysed. A control loop is predefined by several patterns and consists of a circle of system elements, which are assembled like this or in a similar way: basic mechanical system – sensor – controller – actor.
- **Evaluation of the functions:** Two aspects are taken care of in the evaluation of the functions. First of all the functions themselves are checked. In step number three of the procedure displayed in Figure 2, the compatibility analysis was carried out, which creates a connection between the functions and the requirements. During the analysis, the number of existing functions is evaluated, which contribute to the compliance of the requirements. The more functions there are, the higher the fitness value will be. Further on the consistency among the connections between the functions is analysed. These connections match the connections between the system elements in the active structure. They too are defined by the four attributes described. During the analysis, it is checked, how many attributes of a connection are equal.
- **Evaluation of the assembly structure:** It is also two aspects, which are looked at in order to evaluate the assembly structure. First off, the respective shape models are tested for collisions. The lower the amount of collisions, the higher the fitness value. What is also evaluated are the pairs of active surfaces. It is tested whether the size and effect direction of two adjoining active surfaces fits.

All of the evaluations and fitness calculations are relative to the amount of existing components. The maximum value of fitness is 1.

5 RESULTS

To test the methodology, three software tool have been developed: These tools are the so called "Genetic Designer", the "Genetic Optimizer", and "VxDynamik". The tool "Genetic Designer" is used to specify the active structure. "VxDynamik" is used for the graphical representation of the assembly structure and for the simulation of the multi-body system. The tool "Genetic Optimizer" is the core part of the system. The evolutionary algorithm is implemented inside this tool as well as the whole process control. A XML-Interface is used for the communication between these tools.

For the verification of the method an example from the area of vehicle technique was chosen. 89 different system elements are modelled for testing. Aim was to create a vehicle: The vehicle should be able to follow a pre-defined path, the dimension and the weight of the vehicle was limited. Beside a minimum speed and acceleration limit was defined. From the 89 system elements, only 32 could be used to build up a vehicle.

The 89 system elements have been created for testing the algorithm, accordingly the optimal solution was known. This proceeding was chosen, because the performance of the of the algorithms should be checked this way. If an optima could be found can only be checked, if the optima is known. The system elements was prepared, that only a few solutions are possible. Thereby the performance of the evolutionary algorithm could be tested: If the algorithm is able to find the useable system elements and

connect them to a suitable solution or the optima, the algorithm works. Additionally not all system elements could be used to fulfil the requirements. For example, some vehicle parts was created, that are to heavy and to big. System elements was created, that are not strong enough to drive the vehicle.
To verify the method, an example is presented. As mentioned, objective was to create a vehicle. The starting system for the starting population was created by random. It is build up form 21 system elements. The system elements don't result in a suitable solution. A section of the active structure is shown in figure 4.

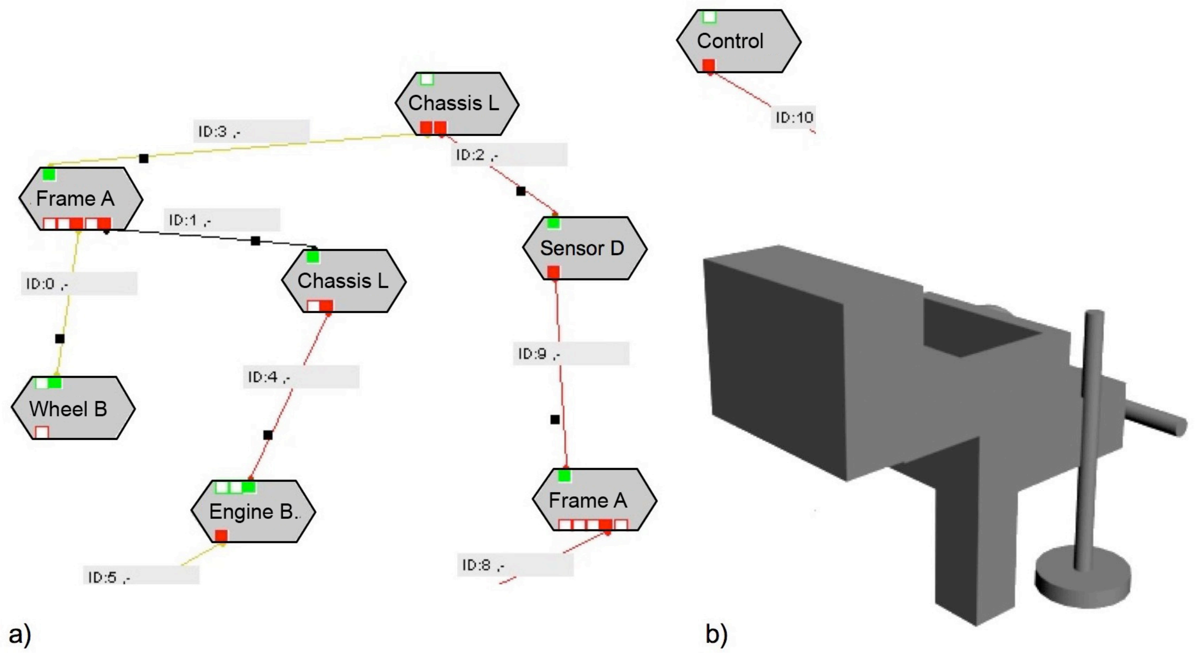


Figure 4: a) Section from the automatically produced active structure and b) representation of the assembly structure of the start system

After 250 iterations the optimization process was stopped. Figure 5a shows the final active structure. The vehicle is build of 14 system elements. All used system elements a part of these set of system elements, which was modelled to build this vehicle. The vehicle fulfils all requirements. In the simulation it is able to follow the predefined path. The achieved result correspond to one of the optimum solutions. Therefore the algorithm is able to find the useful system elements and to find the optima. Figure 5b shows the shape of the solution.
In summary, the algorithm is able to create the four partial models functions, shape, behavior and active structure. The four partial models support the engineers during the conceptual design and to create a principle solution. The algorithm is able to find the useful system elements in a set of system elements. In addition the algorithm is able to connect them into a suitable solutions.

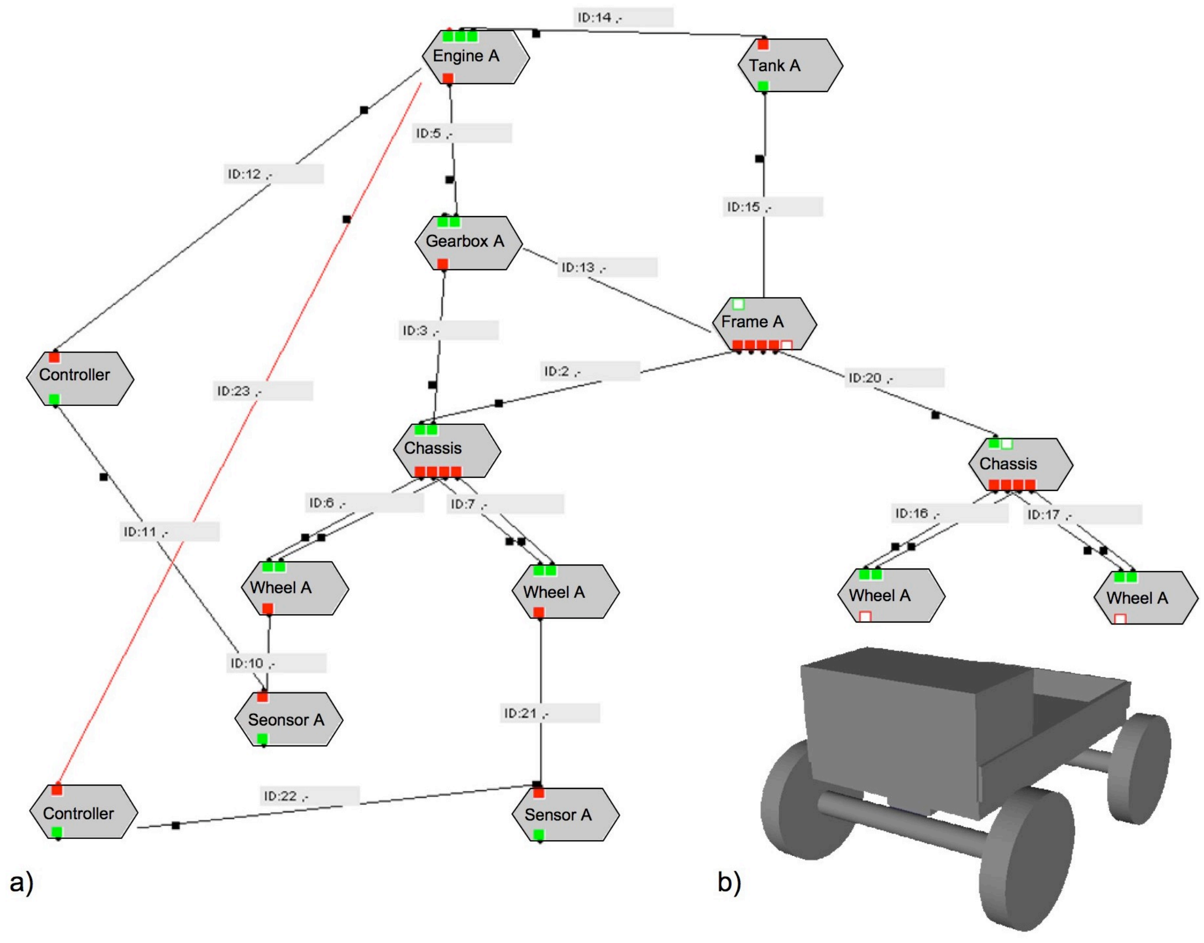


Figure 5: The result of the automatic design process , a) active structure of the sketched system and b) a simple shape model

6 OUTLOOK

This article provide a work, which is a first step into the direction of automatically creation of the principle solution. It could be shown, that the method work and that it can create a solution for a given problem.

The principle solution consists of seven partial models plus one group of behavior models. To create the whole principle solution, more partial models have to take into account by the evolutionary algorithm. These are the partial models environment, scenarios and objectives.

To get more practical solutions, the set of partial models, that are used, have to be extended. The system elements created until now could only be used for testing the performance of the algorithm. The system elements have to become more complex and have to be like real elements and not only test elements. Then a "real world application" will be created.

REFERENCES

[1] Pahl, G.; Beitz, W.: Konstruktionslehre – Methoden und Anwendungen. 5. Auflage, Springer Verlag, Berlin, 2003

[2] Gausemeier, J.; Frank, U.; Steffen, D.: Specifying The Principle Solution of Tomorrows Mechanical Engineering Products. In: Proceedings of International Design Conference, Design 2006, Dubrovnik, Croatia, 2006

[3] Frank, U.: Eine Spezifikationstechnik zur Beschreibung der Prinziplösung selbstoptimierender Systeme. Dissertation, Faculty of Mechanical Engineering, Universiy of Paderborn, HNI-Verlagsschriften-reihe, Band 175, Paderborn, 2005

[4] Chen, K.-Z.; Feng, X.-A.; Chen, X.-C.: Gene Engineering-based innovation of manufactured Products. In: Hoda: A. ElMaraghy; Waguih, H. ElMaraghy (Eds.): Advances in Design, S. 133-144, Springer-Verlag, London, 2006

[5] Vajna, S.; Bercesey, T.; Clement, A,; Jordan, A.; Mack, P.: Autogenetische Konstruktionstheorie – Ein Beitrag für eine erweiterte Konstruktionstheorie. In: Konstruktion, 3/2004

[6] Bentley, P. J.; Wakefield, J. P.: Conceptual Evolutionary Design by Genetic Algorithms. Engineering Design an Automation Journal, V3:2, John Wiley & Sons, p. 119-131, 1997

[7] Leger, C.: Automated Systhesis and Optimization of Robot Configuration: An Evolutionary Approach. Ph.-D. Thesis, Carnegie Mellon University, Pitsburgh, 1999

[8] Goodmann, E.; Seo, K.; Fan, Z.; Hu, J.; Rosenberg, R.: Automated Design of Mechatronic Systems: Novel Search Methods and Modular Primitives to enable Real-World Applications. In: NSF Design, Service and Manufacture Grantees and Research Conference, 7.-120. Jan. 2002, San Juan, p. 202-221, 2002

[9] Pollack, J.B.; Lipson, H.; Funes, P. Hornby, G.: Three Generation of Coevolutionary Robotics. Artificial Life. Vol. 7, pp 215-223, 2001

[10] Pollack, J.B., Lipson, H.: The GOLEM Project: Evolving Hardware, Bodies and Brains. In: the Second NASA/DoD Workshop on Evolvable Hardware. Palo Alto, Californien, 2000

[11] Banzhaf, W.; Nordin, P.; Keller, R.; Francone, F.: Genetic Programming – An Introduction. Morgan Kaufmann Publishers Inc. and dpunkt-Verlag. San Francisco, Heidelberg, 1998

[12] Roth, K.: Konstruieren mit Konstruktikonskatalogen. Band 1, 2. Auflage, Springer Verlag, Berlin, 1994

[13] Koza, J.: Genetic Programming, MIT Press, Cambridge, 1992

[14] Goldberg, D. E.; Deb, K.: A comperative analysis of selection shemes used in genetic algorithms. In: Rawlins, G.: Foundation of Genetic Algorithms. Morgan Kaufmann, San Mateo, 1991

Contact: Dr.-Ing. Rafael Radkowski
Heinz Nixdorf Institute
University of Paderborn
Fürstenalle 11
33102 Paderborn
Germany
Phone: 05251/606228
Fax: 05251/606268
e-mail: rafael.radkowski@hni.uni-paderborn.de