INTERNATIONAL DESIGN CONFERENCE - DESIGN 2010
Dubrovnik - Croatia, May 17 - 20, 2010.

DESIGN 2010

VIRTUAL AND AUGMENTED REALITY FOR SYSTEMATIC TESTING OF SELF-OPTIMIZING SYSTEMS

J. Gausemeier, F. Rammig, R. Radkowski, A. Krupp and W. Müller

Keywords: self-optimzing systems, systematic testing, virtual & augmented reality

1. From Mechatronic to Self-Optimizing

The products of mechanical engineering and related industrial sectors, such as the automotive industry, are often 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 enables advanced mechatronic systems to react autonomously and flexibly on changing operation conditions. Aim of the Collaborative Research Center (CRC) 614 called: "Self-optimizing Systems and Structures in Mechanical Engineering" is the realization of self-optimizing systems which develop procedure models, development methods, and software tools.
Here self-optimization connotes the endogenous adaptation of a system's optimization objectives on changing operating conditions and the subsequent adaptation of the parameters and, if necessary, of the structure and thereby of the behavior of a technical system [Adelt et al. 2008]. Thus, self-optimization operates enormously beyond the familiar rule-based and adaptive control strategies.
The CRC 614 uses the project "Neue Bahntechnik Padernborn/RailCab" as a demonstrator. RailCab is an innovative railway system, which is prototypically realized as a comprehensive test track at a scale of 1:2.5 [http://nbp-www.upb.de]. 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 [Zimmer et al. 2005]. Carrying and leading functions are realized by a wheel-rail contact. Therefore, already existing tracks can be used. With an active tracking module, based on an independent axle chassis with loose wheels, the choice of direction while passing over a switch, takes place vehicle-sided. 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 which the passengers or cargo chassis will be set upon.
The key aspects and the mode of operation of a self-optimizing system are depicted in figure 2: 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, which 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: to ensure the driving functions and a high efficiency. If we (see 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 are examples of internal objectives. The internal objectives are derivated from the above mentioned objectives and they are in relationship with each other. Altogether, they form a system of objectives - the internal system of objectives. 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.

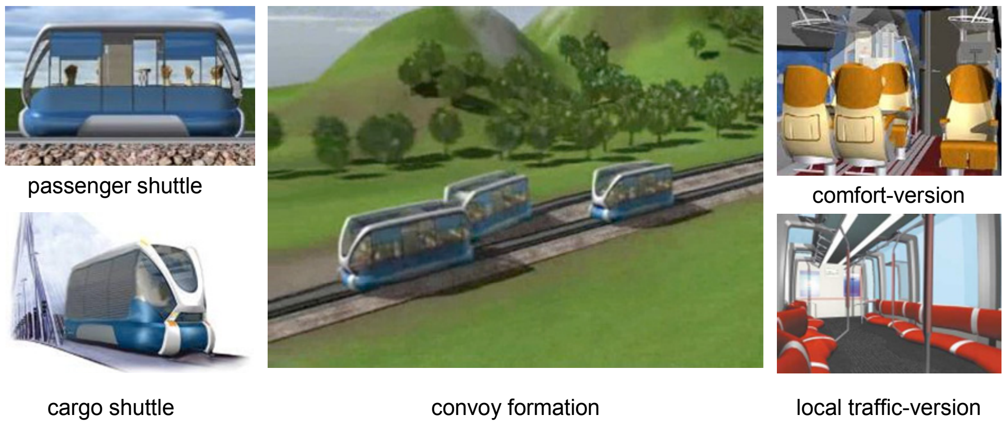

Figure 1. Shuttles of the project „Neue Bahntechnik Paderborn/RailCab“

The adaptation of the objectives leads to an adaptation of the system's behavior. Adapting the objectives in this way leads to an adaptation of the system behavior. This is achieved by adapting the parameters and, where necessary, the structure of the system. The term parameter adaptation means adapting system parameters, for instance, adjusting the controller gains.
Structural adaptations involve reconfigurations and compositional adaptations. It affects the arrangement of system elements and their relationships. Self-optimization takes place as a process that consists of the three following actions, called the Self-Optimization Process:

1. Analysis of the current situation: Here the current situation includes the state of the system itself and all the observations that have been carried out about its environment. Such observations may also be made indirectly by communicating with other systems. The current state of the system also includes any records of previously made observations. One essential aspect of this analysis is the examining of the degree up to which the pursued objectives have been fulfilled.
2. Determination of the system objectives: The current situation includes the current state of the system including all observations of the environment that have been made. Observations can also be made indirectly by communication with other systems. Furthermore, a system's state contains possible previous observations that have been recorded. One basic aspect of this first step is the analysis of the objectives'fulfillment.
3. Adaptation of the system behaviour: The system's objectives can be generated by choice, adjustment and generation. By saying "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 will be created which act out indepently from the existing ones.

Starting from a certain initial state, a self-optimizing system progresses into a new state on the basis of specific influences, i.e. the system undergoes a state transition. The influences that trigger a state transition are referred as events. The self-optimization process defines the activities that effect this state transition, and thereby describes the system's adaptive behavior. It can take place on every hierarchical level of an optimizing system explained in figure 2.
The self-optimizing process demands a controlling and self-optimizing information processing, which must execute multiple complex optimization processes and adaption tasks. New methods are required for their development which allow to manage the system's complexity. The CRC 614 has developed new methods for the model-driven development, by which the code can be generated and verified automatically [Adelt et al. 2008]. The increase of productivity, that attends with model-based methods,

facilitates the integration of complex functions into the information processing of self-optimizing systems.

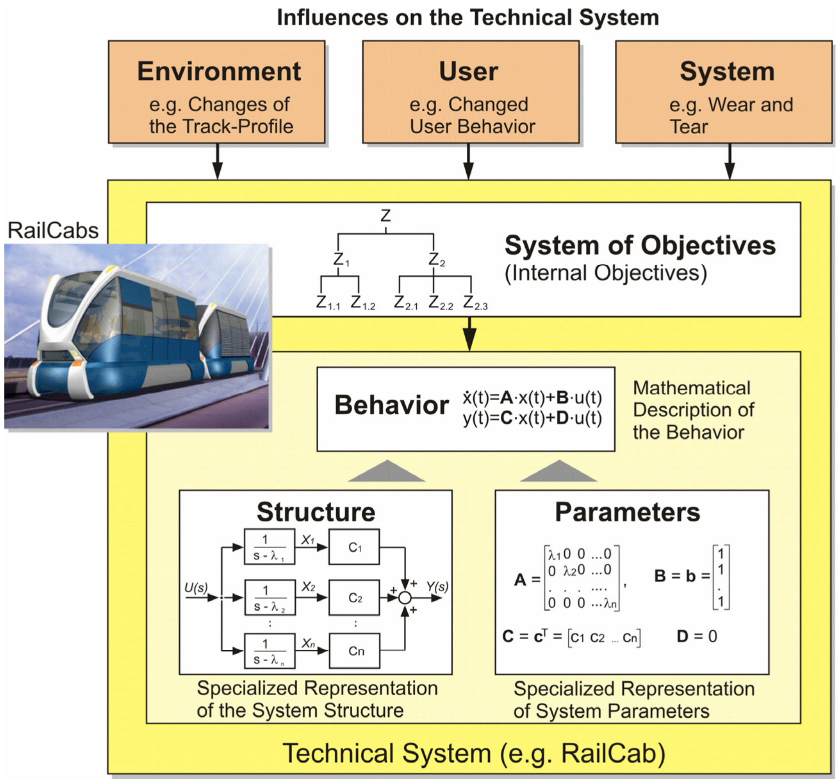
Influences on the Technical System
Environment
e.g. Changes of the Track-Profile
User
e.g. Changed User Behavior
System
e.g. Wear and Tear
RailCabs
Z
Z1
Z2
Z1.1 Z1.2 Z2.1 Z2.2 Z2.3
System of Objectives
(Internal Objectives)
Behavior
ẋ(t)=A·x(t)+B·u(t)
y(t)=C·x(t)+D·u(t)
Mathematical Description of the Behavior
Structure
Parameters
Specialized Representation of the System Structure
Specialized Representation of System Parameters
Technical System (e.g. RailCab)

Figure 2. 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 behavior

2. Systematic Testing

The automatic generation of code using model-based methods bypasses many sources of errors of manual programming. Of course, it is not able to detect design errors; therefor tests are necessary. However, model-based methods open up varied options for formal tests. A formal test method is the so-called model-based test. During the work at the CRC 614, we extended this method in order to validate the functions of self-optimizing systems. Need for action existed within the automatic and formal derivation of test cases. Our method systematically generates a set of test cases, which ensures that all relevant system states will be tested. We express this by the term "systematic testing". For the derivation of the test cases, the classification tree method is used. Also, it has been extended by aspects of self-optimizing systems. Furthermore, an approach has been developed for the validation of the test results by well-considered experiments at a real test-bench.

2.1 Model-based Testing

By operating model-based tests, test cases are derived from a model of the self-optimizing system. A test case specifies e.g. stimuli and a reference behavior of the system. Test cases are used in order to validate the functions of a self-optimizing system. Thereby it is verified, whether the shown behavior meets the desired behavior in all relevant situations. Figure 3 depicts the principle. The stimuli-generator creates stimuli signals, which effect to the system. The model reacts on the stimuli. The reaction is measured with and compared to the reference behavior by an evaluation component. If the shown behavior is equivalent to the reference behavior, then the test will be successful. The system is stimulated by different stimuli, until all relevant system states are tested. The concrete process of a test is specified in the so-called "verification plan". The technical part of the verification plan describes the execution of the test. For instance, the test object and its interfaces are defined, as well as, the basis of information for the systematic test execution.

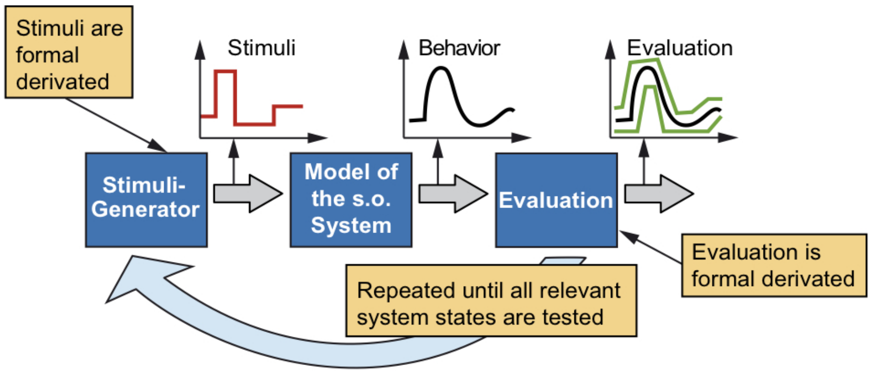
Stimuli are formal derivated
Stimuli
Behavior
Evaluation
Stimuli-Generator
Model of the s.o. System
Evaluation
Evaluation is formal derivated
Repeated until all relevant system states are tested

Figure 3. Principle of the model-based test – stimuli and evaluation are formal derivate out of a model and used in order to test a model of the self-optimizing system

In comparison to test the controlling and self-optimizing information processing, using a real system, the model-based test offers three essential advantages:

- The test, carried out with a model, facilitates the specification of test scenarios at an early stage in the development process.
- Model-based tests facilitate white-box testing methods, because normally, a complete observation of all internal system states of the model is possible.
- The model can be used as a reference model during the automatic test execution. On the one hand it serves as reference for verification of the system response; on the other hand it facilitates the automatic generation of stimuli.

2.2 Classification Tree Method

The classification tree method is used for the formal derivation of test cases out of the principle solution. It is guaranteed by the formal derivative that all relevant system states of an optimizing system are tested [Conrad 2004]. We enhanced the original classification tree method by formal approaches in order to derivate test quality criteria and acceptance criteria. In the following, the classification tree method is described as well as our extensions.
At the beginning of the 90's, the classification tree method was developed by GRIMM and GROCHTMANN, [Grimm 1995], [Grochtmann et al. 1993]. Basic elements of the method are the classification tree and the combination table. In the original notation the classification tree and combination table describe abstract (high-level) test cases. This is done in a graphical notation, as shown in figure 4, by an example of an ABS-braking system.
In the classification tree the entire system (ABS) is splitted into composition nodes. The classifications are specified below the composition, corresponding to single variables of a composition. The classifications are divided into classes, which usually describe exclusive states of a classification, for instance a class of velocity at *ABS.vehicle.speed*. Different combination of states represents different test cases, which are described in the combination table.
CONRAD und FEY have extended this notation by test sequences, which facilitate a specification of time-continuous test cases, [Conrad et al. 1999], [Conrad 2004]. The extensions are well known as classification tree method for embedded systems (CTM/ES). This has already been used successfully in different development projects [Lamberg et al., 2005]. Main field of application is the model-based test of software for automotive controlling devices [Rau 2002]. The advantage of the CTM/ES notation for the generation of test values is the combination of discreet and continuous elements by interpolation.
We have enhanced the classification tree method; we have included new aspects, which are necessary for the systematic testing of self-optimizing systems [Conrad et al. 2006]. These extensions facilitate a formal derivation of major parts or the verification plan, which have been unconsidered so far. Starting point for the specification of the verification plan is the principle solution of a self-optimizing system (figure 5). The principle solution describes all relevant information of the system. In order to specify the principle solution, the CRC 614 has developed a specification technique [Adelt et al. 2008]. These

specifications describe seven aspects: requirements, application scenarios, environment, functions, a system of objectives, active structure, shape, as well as behavior. Computer-based, partial models represent the mentioned aspects. Because the aspects are in relationship to each other and should result in a consistent system, the principle solution consists of a coherent system of partial models. Mainly the partial models include all information, which are necessary to derivate the verification plan.

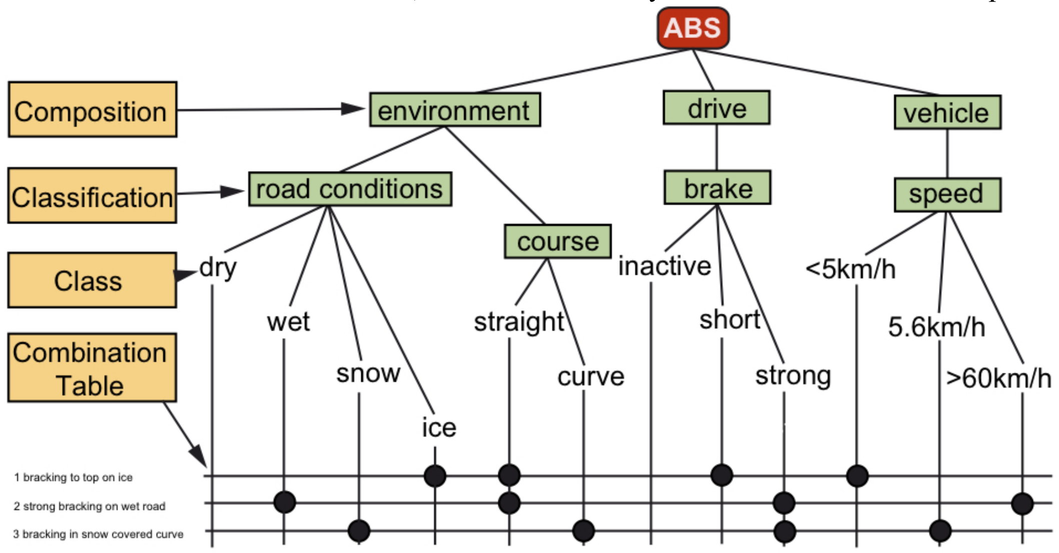

Figure 4. Simplified classification tree of an ABS-breaking system

To sum it all up, the verification plan describes following aspects: CTS-stimuli, constraints, test quality criteria and acceptance criteria. New are uniform notations for the representation of the aspects constraints, test quality criteria as functional coverage, and acceptance criteria.

Constraints are used for a flexible and precise description of stimuli patterns. They allow a declarative definition of stimuli pattern. This facilitates a requirements-based description of stimuli, which show a higher controllability in comparison to CTM/ES. On the one hand this allows the automatic generation of many stimuli. On the other hand existing constraints can be adapted easily during further development.

The new notation for acceptance criteria complete the definition of stimuli pattern and allow an automatic generation of acceptance criteria (tolerance limits) in one step together with the generation of stimuli. An extensive preparation of a reference model at the same level of abstraction like the model-under-test can be avoided.

Test quality criteria describe the objectives of a test. Normally, structure-based coverage criteria are used to define test quality criteria for the model-based test of a self-optimizing system. Indeed this is easy and automatically measurable. But they do not allow a mapping between the coverage metric and the requirements and functions, which should be validated by the test. This disadvantage can be compensated by a functional coverage. Therefore, the concept of functional coverage has been assigned to the classification tree method [Krupp et al. 2006]: a classification tree and a combination table form the basis for the functional coverage.

2.3 Real test at a self-optimizing system

Real test benches and real prototypes are used to validate the results of a model-based test. Even if their number is decreasing, a final validation of functionality can be achieved at a real test bench only. A reason for this is: a developer is able to anticipate, and consequently, to only model some parts of the operation conditions. In addition, some features, like automatic behavior of a self-optimizing system, causes decisions, which have to be made during operation time. These decisions cannot be anticipated, either.

Therefore, a systematic test with a real system is performed (Figure 6). Tests on a real system and on a model are similarly executed. It is essential, that major components and aspects can be transferred from the model-based test to real prototypes: the stimuli-generator, the evaluation with all aspect models which are described by the verification plan. A renewed definition of these components and aspect models would exclude the validation of the systematic tests. Only a new system interface has to be defined; normally, the simulation and real systems using different target platforms. In addition, the evaluation needs to be adapted to the measuring system of the real test bench. In difference to simulations, real systems create noisy values, which need to be filtered mathematically. The necessary technical adaptions are described in [Adelt et al. 2008].

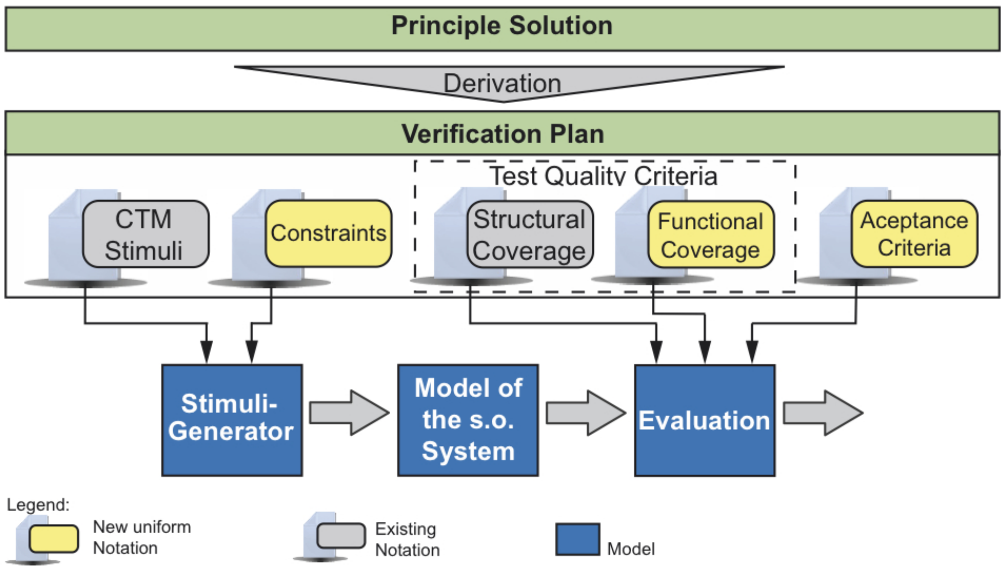

Figure 5. Verification plan for the systematic testing

Concluding, we assume that stimuli-generator and evaluation are able to validate the functions of a real system. Both components have been proven at a model before and have shown that they can test the system successfully. If the model meets the real system, this assumption can be made. Thus, we can validate the real system against the requirements.

3. Virtual and Augmented Reality for Systematic Testing

We use the technologies Virtual Reality (VR) and Augmented Reality (AR) for the analysis and evaluation of test results as well as to plan further tests on a real test bench (figure 6). Using VR and AR during the product development process is not unusual. Today, the technology VR is well established. One field of application is e.g. the design review of novel vehicle models [Katzenbach et al. 2008], [Spors et al. 2009]. VR is also applied in the field of mechanical engineering and plant engineering, in order to plan and evaluate technical systems [Ye et al. 2007]. In comparison to VR, the technology AR is still a novel technology. AR is a human-computer-interface, which superimposes the perception of reality with computer-generated information. Already today, first applications can be found in niches. Cutting-edge fields are automotive development and marketing. For instance, the automotive development uses AR for the evaluation of new car prototypes and for the preparation of experiments [Wittke 2007], [Klinker et al. 2002]. Marketing uses AR for product presentation. Recently, a manufacturer of toy building blocks tested AR, in order to present the final product on top of the package [Metaio 2008l].
During a detailed analysis of the existing approaches, we realized that AR applications normally concentrate on the presentation and manipulation of the shape of a product. This is not sufficient for the analysis of self-optimizing systems, because a main part of the added value comes from controlling and self-optimizing information processing. Besides, formal methods, such as systematic testing, are only supported insufficiently by visualizations. Our approach surpasses the common approaches. We use AR and VR to explain to developers, how information processing and self-optimizing works. Furthermore, we support the objective-oriented analysis of systematic tests. In the following, we describe our approach using to selected examples.

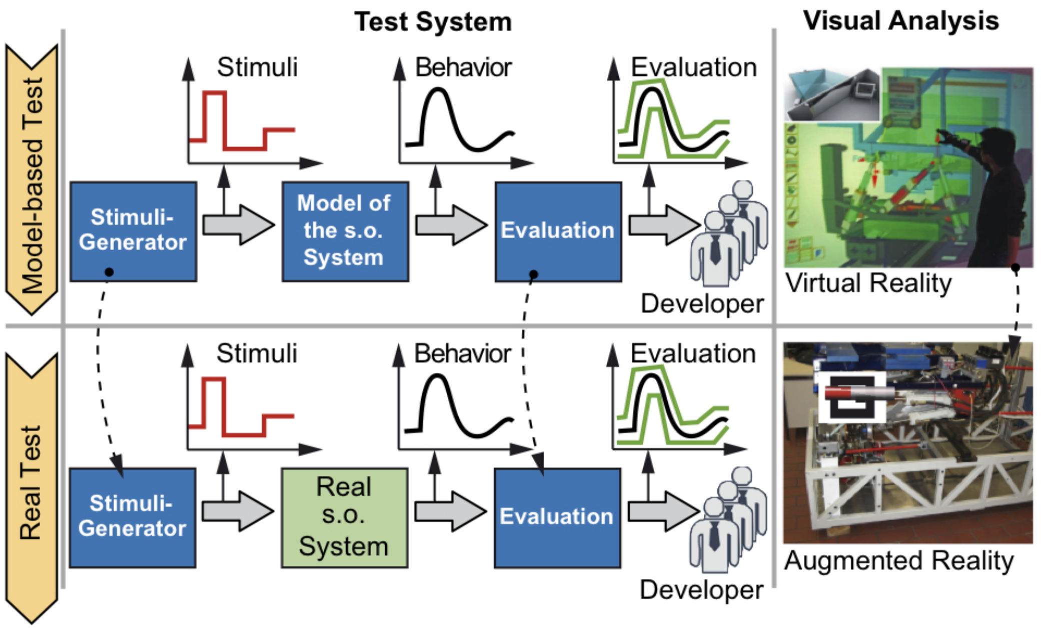

Figure 6. Overview about the AR/VR-based systematic test of a self-optimizing system

3.1 VR for the analysis of model-based tests

We use VR for the analysis of model-based tests as well as to plan further real tests at a test bench. Figure 7 shows the developers' view onto the model under test. As example, we use a module of a RailCab. In this case it is a so-called tracking chassis with chamber adjustment. It is an active chassis, it pitches its wheels autonomously, according to the characteristics of the track.

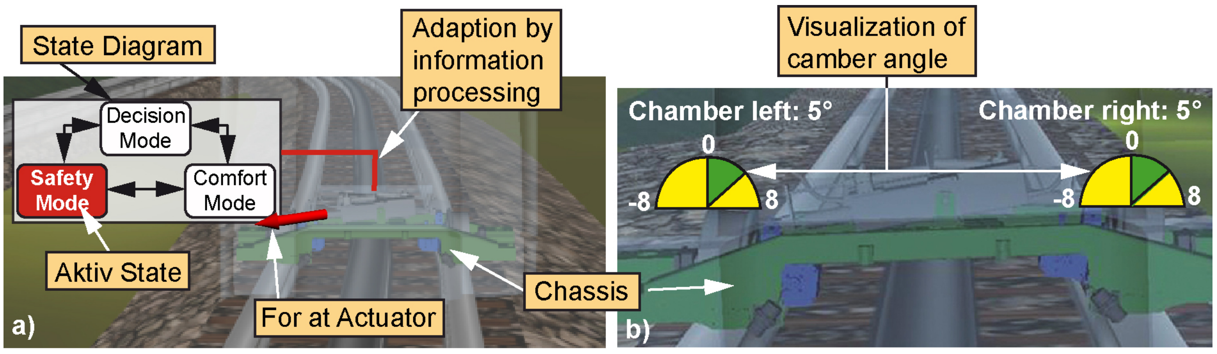

Figure 7. a) Visualization of the shape, the information processing, and b) the behavior of a active chassis with adjustable chamber angle

The VR application shows all 3D models of the chassis and all actuators and sensors, which are necessary to operate the chassis functions. The 3D models are linked to a multi-body simulation. When the stimuli generator stimulates the system, the engineer can observe the movements of the chassis directly. In addition, the information processing is presented by plausible visualizations. In the example, a state diagram visualizes the current state of the chassis' controller. Ongoing optimization processes are shown by a color code as well as the adaption of system parameters. A red frame around a state indicates an active state. A yellow frame indicates an ongoing optimization process. Lines refer e.g. to actuators or sensors, which parameters are currently optimized by the information processing. Additionally, different system parameters and their trespassing of limits are visualized. Figure 7a shows the visualization of current working forces. For this purpose, an arrow is used. This way, single movements can be compared and their limits can be indentified.

Moreover, the chamber angle of each individual wheel is visualized (Figure 7b). The angle is too small in order to be recognized when observing the 3D model; a wheel tilts max. about ±8°. Therefore, the chamber angle of each wheel is illustrated by a diagram. Visualized is the area of a semicircle, which

represents a data range from -8° to 8°. The semi-circle is placed over the wheels of the chassis, in parallel to the image plane. The current value of the chamber angle is marked, as the appropriate proportion of the circular arc is colored green. In the figure, both wheels fall 5° to the right. In summary, VR facilitates the analysis of evaluation of systematic test and the comparison of test results.
Furthermore, VR is used for the planning tests at a real system. Real tests are expensive and time-consuming and must be well-considered. Based on the analysis in VR, engineers can first localize the critical system states or the related stimuli. For instance, these are stimuli to which the system reacted unexpectedly or even a permitted range of tolerance has been violated.

3.2 AR for the Analysis of Real Tests

In order to support the engineer during the test at a test bench, we developed an AR application, which explains the behavior of the test bench using different visualizations. In the following, we explain our approach with several examples, which we have tested at different modules of the RailCab system. Figure 8 shows two views of an engineer through a head mounted display, a special viewing device, which is necessary for AR. In Figure 8a) a visualization is presented, which we use to show the overstepping of limits. An arrow highlights the part, which violates the limit. Figure 8b) shows the visualization of stimuli that are generated by the stimuli-generator. The image shows two test signals (sinus waves), the stimuli are shown as 2D head up displays.

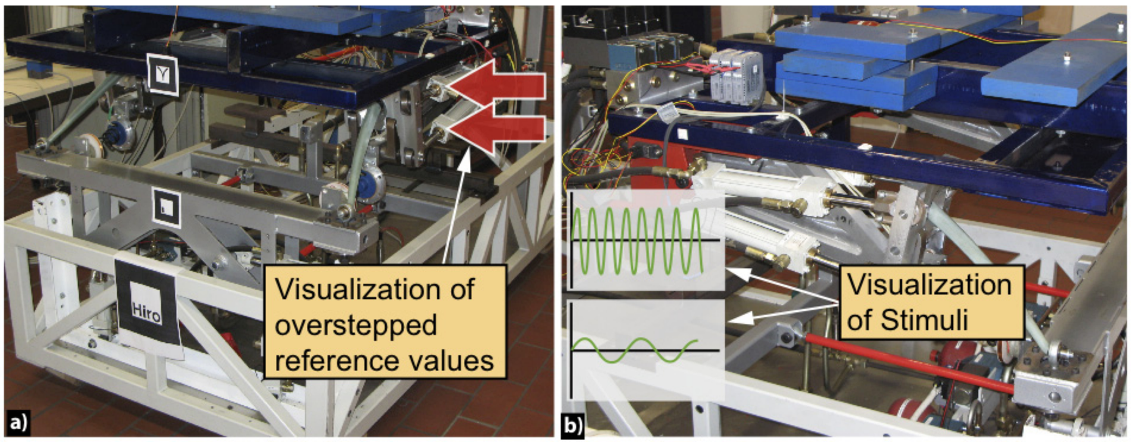

Figure 8. a) Actuators, violating a limit, are highlighted by arrows. b) Stimuli of the stimuli-generator are presented as a 2D head-up display

Both visualizations direct the attention of the engineer to a critical situation. If everything works well, there is no need for the engineer to observe the data. The tests are running automatically. But if anything happens, the engineers are able to identify the situation. This simplifies the analysis of systematic tests in a post-process because AR shows relevant states and limits the search space.
Figure 9 shows a second visualization; it shows the progress of a systematic test. The systematic test should test relevant system states by different stimuli onyl. During the test, it is evaluated, whether the behavior operates within the specified limits. Because there are many possible stimuli, these are produced by a stimuli generator. If the stimuli are secondary only, e.g. if they result from the movement of another mechanical component of the test bench, these movement must be measured. In order to check, whether the essential stimuli were tested, test sequences are defined. These test sequences describe the range of values, in which stimuli have to run.
The test sequence describes the range of a single input value of a component or also combined for several components. While the test is running, it is checked whether a test sequence was already tested. If this is the case, the results are stored in a so-called coverage metric. AR represents this metric, respectively the number of input sequences, which are already checked on the related component. Therefore, a bar chart is visualized with spatial relation to the related component. With increasing number of tested sequences the fill level of the bar chart increases. This allows the engineers to see during the test, how successfully the system is tested. If the level of the bar chart

stagnates or remains at the same level, no relevant new results are generated. Then the experiment should be aborted or changed.

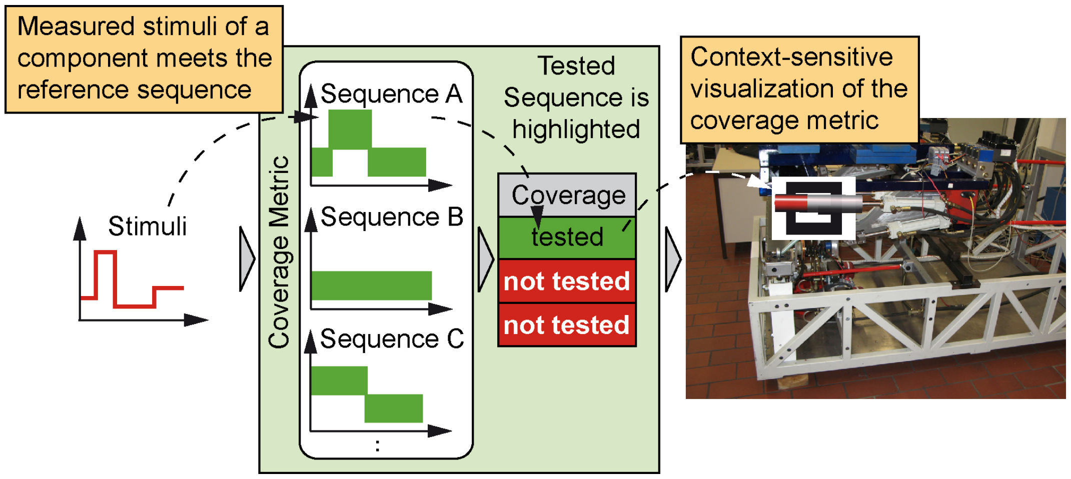
Measured stimuli of a component meets the reference sequence
Stimuli
Coverage Metric
Sequence A
Sequence B
Sequence C
:
Tested Sequence is highlighted
Coverage
tested
not tested
not tested
Context-sensitive visualization of the coverage metric

Figure 9. Visualization of a coverage metric with spatial relation to a real component

Besides this, AR is used for the visual analysis. One precondition for validating functions is comparable tests and results at the real and simulated system. Because the tests and results are never identical, the engineer must evaluate its comparability subjectively. AR enables a visual comparison by a graphical overlay of the test results of the model-based test with spatial relation to the real test bench. Thus, AR becomes an essential medium for a successful validation of the functions of the self-optimizing system.

4. Résumé and Outlook

Self-optimizing systems demand formal methods for the development and test of the controlling and self-optimizing information processing. A common method is model-based testing. For the analysis and evaluation of self-optimizing systems, an extension of the common method is necessary. For that purpose we have chosen the classification tree method and enhanced it. Using our extensions, test quality criteria and acceptance criteria can be generated directly from the principle solution. This facilitates a systematic testing of a self-optimizing system. Moreover, we can guarantee that all relevant system states are tested. A methodical gap between the principle solution and the verification plan is closed by this work. However, visualization techniques are necessary for planning, analysis and evaluation of systematic tests, which illustrate to the developer, how the system operates or in which situations errors occur. Special attention has spent on visualizations, which explain the controlling and self-optimizing information processing. For that purpose, we use the technology VR and AR. In contrast to usually used 2D diagrams, texts and other digital displays, VR and AR simplify information recognition, which guides the developer during the test. It facilitates to get a holistic overview about the entire situation. Altogether, we showed that the combination of model-based test and VR/AR based analysis simplifies the validation of functions substantially. Thus, the AR/VR based systematic testing becomes an indispensable tool for the development of self-optimizing systems.
In further work, we will develop a self-optimizing experimentation environment as well as autonomous visualization agents. By a self-optimizing experimentation environment we refer to a tool for the controlled execution of systematic tests, whereby decisions about experiments, which have to be executed, are shifted into the execution phase of the experiment. Autonomous visualization agents are needed, in order to explain the behavior of optimizing systems during learning and optimization processes to the developers. By making use of a visualization agent we will understand an autonomous software component, which acts as an “expert” for a certain visualization. The component contains knowledge about its visualization, their aimed application and the functions, which an engineer can solve by visualization. They act autonomously and indicate independently information about the system to the developer.

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.

References

Adelt, P.; Donoth, J.; Gausemeier, J.; Geisler, J.; Henkler, S.; Kahl, S.; Klöpper, B.; Krupp, A.; Münch, E.; Oberthür, S.; Paiz, C.; Porrmann, M.; Radkowski, R.; Romaus, C.; Schmidt, A.; Schulz, B.; Vöcking, H.; Witkowski, U.; Witting, K.; Znamenshchykow, O.: “Selbstoptimierende Systeme des Maschinenbaus – Definitionen, Anwendungen, Konzepte”. HNI-Verlagsschriftenreihe, Band 234, Paderborn, 2008

Conrad, M.; Dörr, H.; Fey, I.; Yap, A.: “Model-based Generation and Structured Representation of Test Scenarios.” In: Workshop on Software-Embedded Systems Testing (WSEST), Gaithersburg, USA, 1999

Conrad, Mirko: “A Systematic Approach to Testing Automotive Control Software.” In: Proc. of 30. Int. Congress on Transportation Electronics (Convergence ’04), pages 297–308, Detroit, MI, USA, October, 2004

Conrad, Mirko; Krupp, Alexander: “An Extension of the Classification-Tree Method for Embedded Systems for the Description of Events.” In: Second Workshop on Model Based Testing, MBT 2006, Vienna, Austria, 2006

Grimm, K.: “Systematisches Testen von Software - Eine neue Methode und eine effektive Teststrategie.” Number 251 in GMD-Report. GMD, Oldenbourg, 1995.

Grochtmann, Matthias; Grimm, Klaus: “Classification Trees for Partition Testing.” In: Software Testing, volume 3(2) – Verification and Reliability, pages 63–82, 1993

Katzenbach, A.; Haasis, S.: “Virtual and Mixed Reality in a SOA Based Engineering Environment.” In: Design Synthesis, CIRP Design Conference, April, 7-9, 2008, Enschede (NL), 2008

Klinker, G.; Dutoit, A.; Bauer, M.; Bayer, J.; Novak, V.: “Fata Morgana - A Presentation System for Product Design.” In: Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR 2002), 2002

Krupp, A.; Müller, W.: “Classification Trees for Random Test and Functional Coverage.” In: Design, Automation and Test in Europe (DATE 2006), Munich, Germany, 2006

Lamberg, K.; Beine, M.; Eschmann, M.; Otterbach, R.; Conrad, M.; Fey, I.: “Model-based Testing of Embedded Automotive Software using MTest.” In: SAE 2004 Transactions, Journal of Passenger Cars - Electronic and Electrical Systems, 7:132–140, 2005

Metaio: “The LEGO Group to boost retail with metaio”, Pressemitteilung Dezember 2008, http://www.metaio.com

Rau, A.: “Model-Based Development of Embedded Automotive Control Systems.” PhD thesis, Dept. of Computer Science, University of Tübingen, Germany, 2002

Spors, K.; Martin, A.; Leetz, A.: “Möglichkeiten fotorealistischer Visualisierungen im Produktionsprozess eines Automobils.” In: ATZ - Automobiltechnische Zeitschrift (03/2009), pp. 1-8, Vieweg+Teubner Verlag, 2009

Wittke, M.: “AR in der PKW-Entwicklung bei Volkswagen.” In: Schenk, M. (Hrsg.): IFF-Wissenschaftstage - Virtual Reality und Augmented Reality zum Planen, Testen und Betreiben technischer Systeme, 4. Fachtagung zu Virtual Reality, 27. - 28. Juni 2007, Fraunhofer IFF, Magdeburg, 2007

Ye, J.; Badiyani, S.; Raja, V.; Schlegel, T.: “Applications of Virtual Reality in Product Design Evaluation.” In: J. Jacko (Ed.): Human-Computer Interaction, Part IV, HCII 2007, LNCS 4553, pp. 1190–1199, 2007 Springer-Verlag Berlin Heidelberg 2007

Zimmer, D.; Böcker, J.; Schmidt, A.; Schulz, B.: “Elektromagnetische Direktantriebe im Vergleich.” In: Antriebstechnik, Ausgabe 2/2005, Vereinigte Fachverlage GmbH, Mainz, 2005

Dr.-Ing. Rafael Radkowski
Heinz Nixdorf Institute, Product Engineering
Fürstenallee 11, 33102 Paderborn, Germany
Telephone: +49 5251 / 606228
Telefax: +49 5251 / 606268
Email: rafael.radkowski@hni.uni-paderborn.de
URL: http://www.hni.uni-paderborn.de/en/pe