What is the benefit of a model-based design of embedded software systems in the car industry?

Manfred Broy
Technical University Munich, Germany
Sascha Kirstan
Altran Technologies, Germany
Helmut Krcmar
Technical University Munich, Germany
Bernhard Schätz
fortiss, Germany
Jens Zimmermann
Altran Technologies, Germany

ABSTRACT

Model-based development becomes more and more popular in the development of embedded software systems in the car industry. On the websites of tool vendors many success stories can be found, which report of efficiency gains from up to 50% in the development, high error reductions and a more rapid increase of the maturity level of developed functions (The Mathworks, 2010) (dSPACE 2010) just because of model-based development. Reliable and broadly spread research that analyze the status quo of model-based development and its effects on the economics are still missing. This article describes the results of a global study by Altran Technologies, the chair of software and systems engineering and the chair of Information Management of the University of Technology in Munich which examines the costs and benefits of model-based development of embedded systems in the car industry.

1. INTRODUCTION

In the last 20 years the value chain in the car industry has changed drastically. All car producers and suppliers worldwide have worked on improvements in the area of mechanics, the improvement of quality requirements, and improvements in the logistic area. A lot of the potential in these areas is already exploited. A main differentiation factor turns out to be the electronics area, where a change from hardware to software development is carried out. The meaning electronics will have in the next years has been analyzed by a study of Mercer Management Consulting (Mercer, 2004). The study focuses mainly on the question how the cost factors in the development of a car will change until the year 2015 in comparison to the year 2002. In 2015 the costs for the development of electronics will have a value of 35% of the total car production costs. Whereas areas as power train and body have small increases, the costs for the development of electronic systems will be almost tripled. The predicted increases result from a variety of innovations which are being expected in this area. The majority of innovations are realized with embedded systems and especially with software. „90 percent of the future innovations in the car will be based on electronics and from that 80 percent will be realized by software" (Lederer, 2002). However, today's software development has big challenges to master like shortened development times for the cars in total versus longer development times for the software, high safety requirements and especially the growing complexity because of the rising number of functions and the increasing interaction between the functions. To master these challenges car producers and suppliers conduct a paradigm change in the software development from hand-coded to model-based development.

A model-based development process is specifically attractive in embedded domains like Automotive Software due to the fact that development in these domains is driven by two strong forces: On the one side the *evolutionary* development of automotive systems, dealing with the iterated integration of new functions into a substantial amount of existing/legacy functionality from pervious system versions; And on the other side *platform-independent* development, substantially reducing the amount of reengineering/ maintenance caused by fast changing hardware generations. As a result, a model-based approach is pursued to enable a shift of focus of the development process on the early phases, supporting a function-based rather than a code-based engineering of automotive systems. Thus, the pragmatic question arises whether *a model-based approach – focusing on model of functionality as the most stable asset – is an economic approach in a domain driven by functional evolution as well as by hardware revolutions*.

On the one hand model-based development promises considerable productivity increases, improvements in quality and cost savings. On the other hand, it brings challenges since the use of model-based design results in a major process redesign. The introduction of model-based development influences established development processes, required resources and thereby also the organizational structure. In addition, high investment costs for tools and for training of the employees are necessary.

There is a controversy in the automotive industry about the benefit of model-based software development. Some companies seem to benefit of a model-based design and some don´t. Although model-based development is used by several car producers and suppliers, no major empirical investigations of the costs and benefits of model-based development have been conducted yet. Our aim is to analyze the costs and benefits of model-based development of embedded software systems in the car industry in detail, identify criteria how to optimize the costs and benefits of a model based development and give an outlook about the potential of further model-based development in development phases like requirements engineering and architecture design. In this chapter we present some of the main results of our research work.

2. RELATED WORK

Statements about the benefits of model-based development in the car industry are quite rare. Most of the statements come from tool vendors, who report about successful projects their customers have conducted (The Mathworks, 2010) (dSPACE 2010). But a neutral investigation of the costs and benefits of model-based development in the car industry has not been conducted yet. Fieber at al. (Fieber, Regnat & Rumpe, 2009) have conducted an empirical study about the benefits of model-based development, but this study covers many different sectors, not only the automotive domain, and is only conducted within one single company. Results of the study have up to now not been published, but the authors present working hypothesis in their paper which result from six interviews, which they had already conducted. These are for example that:

- ”Models are hardly used for communication or documentation purposes but mostly for generating purposes. Most teams did not model before they introduced model driven development. The introduction of modeling activities is regarded as a necessity of the introduction of the model driven development paradigm. The terms modeling and generating are often seen as synonyms as the teams use models only for generating artifacts.
- Projects that successfully adopt the model driven development paradigm are small and agile. The paradigm is not predetermined from the organization or driven by economical considerations but is started from within the teams (“grass roots movement”). There is often one key team member pushing the adoption of the paradigm.
- The most successful adoption of modeling and the model driven development paradigm is achieved when the team members have formal qualifications (e. g. a computer science degree) and are systematically trained in (UML) modeling.
- Typical and important reasons for adopting the model driven development paradigm are raising the software quality and enforcement of consistent structures and architectures.” (Fieber, Regnat & Rumpe, 2009)

(Fey & Stuermer, 2007) examined the state of the art of quality assurance (QA) methods, the pros and cons of each method and the needed effort to use them. As a result of the analysis they report that model-based development significantly improves the quality of the automotive embedded software development process. For each investigated quality assurance method like for example model reviews or automated model checks they give statements about the automation degree, the effort and the benefit. They come to the conclusion that due to the relatively high effort required to safeguard the model-based development process, it is still desirable to reduce the effort and increase the effectiveness of the applied QA methods. (Murphy, Wakefield & Friedman, 2007) have listed best practices for verification, validation and test in model-based design. The paper concludes that model-based design improves a team's ability to deploy a high-quality embedded system on time compared to traditional methods, which rely on verification, validation and testing at the end of the process. Best practices for establishing a model-based design culture like using models to generate the production code, focus on design instead on coding or that models are the sole source of truth can be found in (Smith, Prabhu & Friedman, 2007). These best practices shall help companies in adopting model-based design and achieve gains in efficiency in the development process. Asadi and Ramsin (Asadi & Ramsin, 2008) analyzed different MDA-based methodologies and found out that the methodologies are not mature enough. The majority of the methodologies have a lack when it comes to features like round trip engineering, model verification and validation or model synchronization. Another problem is that all tool-related issues are being handled by the tool vendors. As a consequence the tools are sometimes not in coherence with the proposed MDA-methodology. Dzidek et. al (Dzidek & Arisholm & Briand, 2008) conducted an empirical evaluation to analyze the impact of model-based development in software maintenance. They conducted a controlled experiment that "investigates the costs of maintaining and the benefits of using UML documentation during the maintenance and evolution of a real nontrivial system, using professional developers as subjects, working with a state-of-the-art UML tool during an extended period of time" (Dzidek & Arisholm & Briand, 2008). They found out that the UML group had on average a statistically significant 54% increase in functional correctness of changes and an insignificant 7 percent overall improvement in design quality. A much larger improvement was observed on the first change task (56 percent), at the expense of an insignificant 14 percent increase in development time caused by the overhead of updating the UML documentation.

Another interesting work is from (Mohagheghi & Dehlen, 2008) who reviewed 25 empirical studies by evaluating reasons for and effects on applying the model driven development paradigm in industrial projects. They found out that the ultimate reason for applying the model driven development paradigm in the companies are hopes to increase the productivity and thereby shorten the development time and improving quality. When having introduced model driven development, the companies sometimes report of huge productivity losses, because of immature tools and all companies report of high costs for the process redesign from classical to model-driven development. The benefits of using models is seen in improving the understandability and communication among stakeholders. The paper ends with the result that more empirical studies have to be conducted to analyze the costs and benefits of model-based development as some companies don´t use model-based development, because of the high investment costs and the unknown benefits. Taking the results of the related work into account, the importance to analyze the costs and benefits of model-based development is confirmed by the conducted literature review.

3. PROCEDURE TO ANALYSE THE COSTS AND BENEFITS OF MODEL-BASED SOFTWARE DEVELOPMENT (MBSD)

As presented in Figure 1 our approach to analyze the costs and benefits of model-based development consists of five steps. First of all a theory, which summarizes assumed changes in costs, time and quality because of model-based development was developed. The idea behind the theory is to analyze major differences between a hand-coded and a model-based development process. Therefore a couple of hand-coded and model-based development processes, which are used by car manufacturers and suppliers, were analyzed. As a result we developed a reference process for a classical software development and a reference process for a model-based software development. These two reference processes were compared and major differences were identified. To make sure that all major differences were being identified we also used the activity index of the V-Modell XT (IABG, 2010), which summaries all essential steps in a software development. The identified differences are in our opinion responsible for the changes in costs, time and quality. After identifying the differences in the development process and analyzing their influence on cost, time and quality changes, a case study was conducted to evaluate the theory (step 2). After the case study, different cost models based on the theory were developed (step 3). These cost models have been validated with experts from industry and research. As next step a global study (step 4) was conducted to get quantitative data on cost, time and quality changes in dependency of the conducted steps (degree of MBSD) in the development process like the degree of modeling and code generation or the use of test activities on models. As last step a validation of the cost model with the data of the global study at a car producer (step 5) has been conducted to see how precise the predictions of the cost model were.

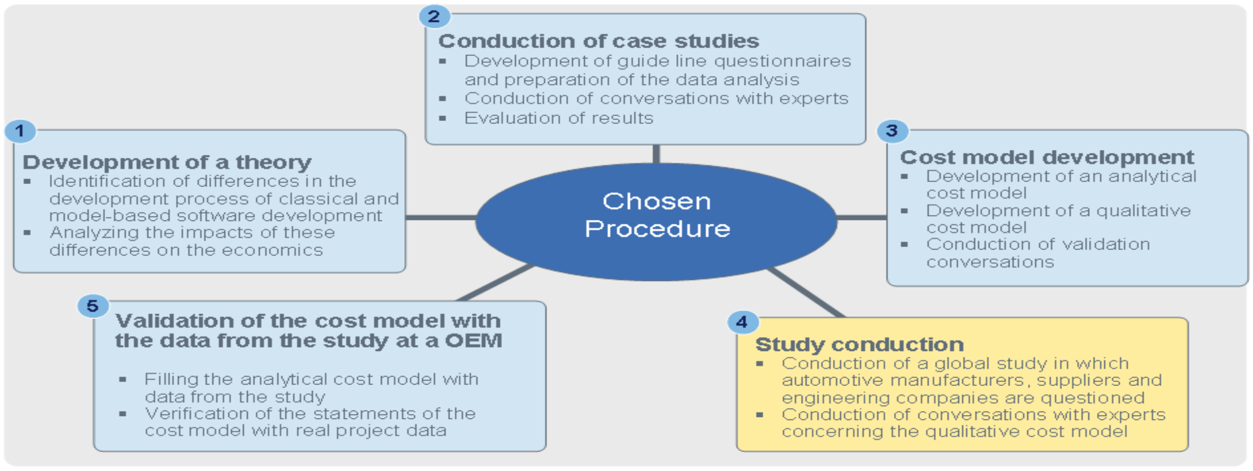

2
Conduction of case studies
Development of guide line questionnaires and preparation of the data analysis
Conduction of conversations with experts
Evaluation of results
1
Development of a theory
Identification of differences in the development process of classical and model-based software development
Analyzing the impacts of these differences on the economics
Chosen Procedure
3
Cost model development
Development of an analytical cost model
Development of a qualitative cost model
Conduction of validation conversations
5
Validation of the cost model with the data from the study at a OEM
Filling the analytical cost model with data from the study
Verification of the statements of the cost model with real project data
4
Study conduction
Conduction of a global study in which automotive manufacturers, suppliers and engineering companies are questioned
Conduction of conversations with experts concerning the qualitative cost model

Figure 1: Approach to analyze the costs and benefits of MBSD

In this paper we focus on the results of the global study (step 4), which we present in the following chapter.

4. ESSENTIAL RESULTS OF THE STUDY

The study was conducted by Altran Technologies in cooperation with the chair of software and systems engineering and the Information Management chair of the technical university in Munich. We invited more than 850 experts personally worldwide to take part in the study. We did this by contacting costumers of Altran Technologies, companies who work together with the chair of Software and Systems Engineering of TUM and companies which have published in magazines, online or at conferences about their model-based development activities. In addition we also contacted experts in business portals like LinkedIn and Xing and made advertisements to participate in our study on the Altran Technologies website. To raise the number of participants also anonymous participation was possible. The people who answered the questionnaire anonymously only had to fill out general information about the company in our questionnaire like is the questionnaire answered from a car producer, supplier or a technology consulting company and the size of the company. About 30% of the questionnaires were answered anonymous. In total we received 67 filled out questionnaires which have been answered by almost 180 experts. The reason therefore is that the questionnaire was so comprehensive that usually 3 to 4 people of one company were needed to fill out all the questions of the questionnaire. Study participants were mainly car producers and suppliers. In addition we also involved technology consulting companies, because these companies work for the car producers and suppliers and develop a lot of software for them. Figure 2 describes the structure of the participants. It can be seen that with 58% the majority of the participants were suppliers followed by the car producers with 33%. This distribution makes sense as the number of suppliers is much higher as the number of car producers. The areas of responsibility of the study participants range from experts in research departments or the serial development of the companies till directors of electronics and CEO´s. The huge know-how of our study participants ensures the significance of the study results.

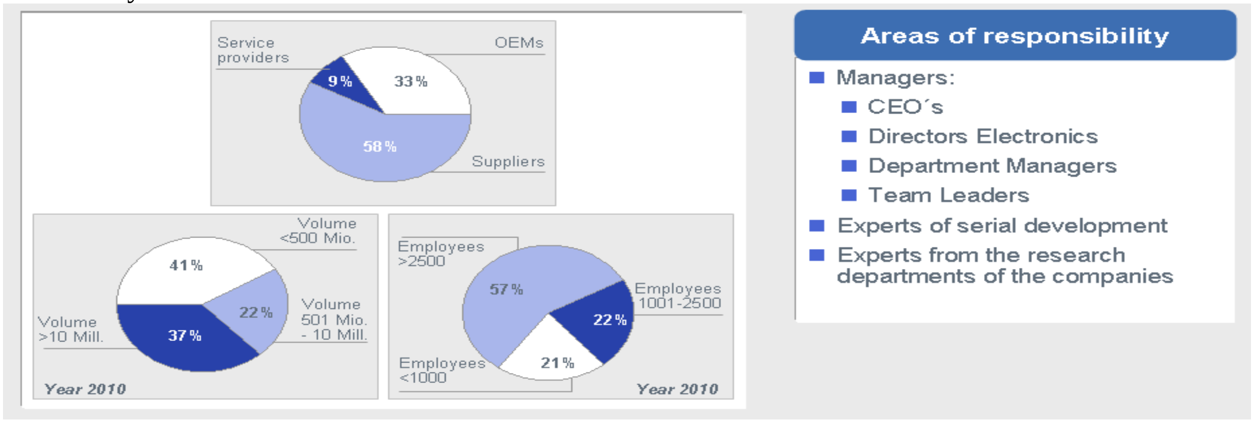


Figure 2: The structure of the study participants

The study has four main pillars. First of all we want to find out the reasons why companies use model-based development. Secondly we wanted to know the positive as well as the negative experiences the companies have made with model-based design. In the following section of the study we focus on how intensive model-based design is used in each development phase (status quo of model-based design) and what effects this has on the costs of each development phase and on the total development costs. In the following pillar of the study we concentrate on the potentials of model-based development which are mainly in the area of model based requirements engineering and model based architecture design. The study ends with recommendations for action, which were derived from the results of the study. Figure 3 summaries all the aspects which are being dealt with in the study.

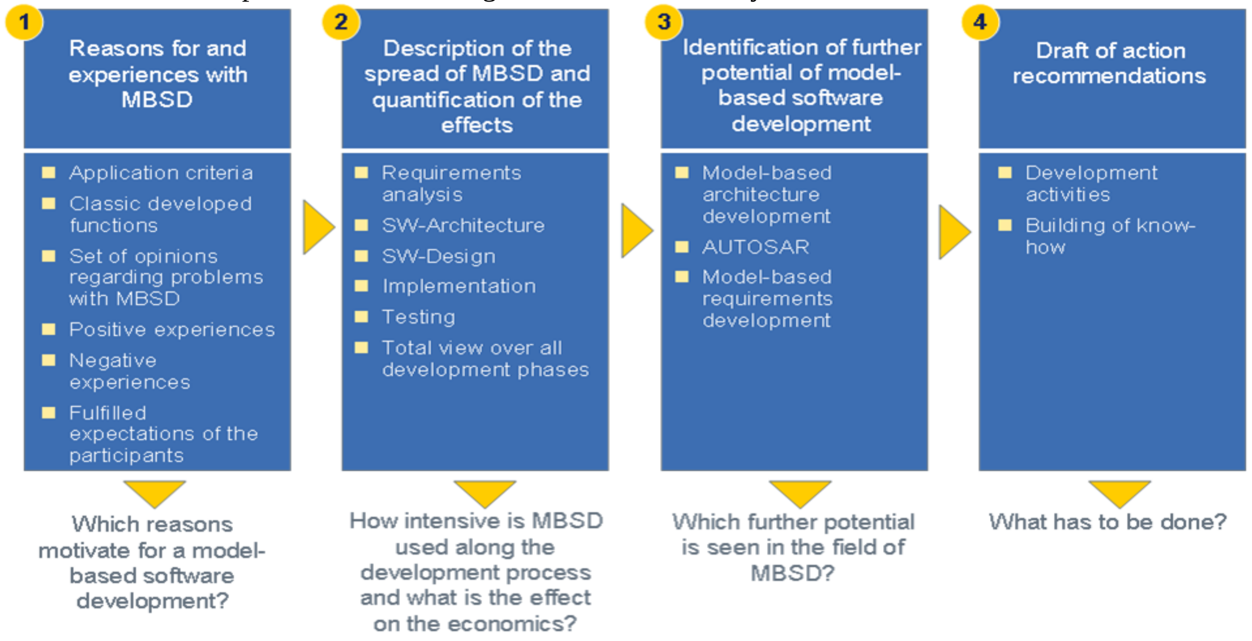

1
Reasons for and experiences with MBSD
Application criteria
Classic developed functions
Set of opinions regarding problems with MBSD
Positive experiences
Negative experiences
Fulfilled expectations of the participants
Which reasons motivate for a model-based software development?
2
Description of the spread of MBSD and quantification of the effects
Requirements analysis
SW-Architecture
SW-Design
Implementation
Testing
Total view over all development phases
How intensive is MBSD used along the development process and what is the effect on the economics?
3
Identification of further potential of model-based software development
Model-based architecture development
AUTOSAR
Model-based requirements development
Which further potential is seen in the field of MBSD?
4
Draft of action recommendations
Development activities
Building of know-how
What has to be done?

Figure 3: The four pillars of the study

4.1 Our understanding of model-based development

The term model-based development is used so widely and has so many different meanings that we like to present our understanding of model-based development before we present the results of the study. In our understanding we speak of model-based development when the following criteria are fullfield (see also (Schätz, 2004)):

- The use of a weakly coupled tool chain
- The use of models in different development phases
- Models offer different views
- Activities are conducted with tools on the model
- Generation of code and other artifacts out of the model

In the following chapter we like to present major results of the study. First of all we present the status quo of model-based development in the car industry. Afterwards we focus on the analysis of the costs and benefits of model-based development in the development phases software design and implementation, in the total view and the impact on the maintenance costs. After presenting the results of the study we focus on potentials of further model-based development.

4.2 Status Quo of model-based development in the car industry

The study shows that model-based development is being used for series development by the majority of the companies for more than five years. Especially in the development phases software design and implementation model-based design is used intensively. 96% of all participants use model-based development in both development phases. The other 4% have just begun using model-based development and use it in the SW-design. 75% of the companies use model-based design already in the requirements engineering phase by using Rapid Control Prototyping (RCP). We want to keep in mind that RCP is not model-based requirements engineering. Instead it is already designing with the modeling tool. Nevertheless the study participants report that it is an efficient method to identify missing requirements, especially while developing innovative functions. A trend can be seen that more and more companies use model-based design also in the architecture development. The aim is to increase the seamlessness in the development process and to perform early tests on the architecture model. Model-based testing (i.e. the generation of test cases out of a test model) is currently not used intensively. Only 35% of the participants use it right now, but almost 50% plan to use it in the near future.

We asked the study participants why their company uses model-based development. The top three reasons were:

- **Improvement of the product quality:** The companies hope that the frontloading of the test activities and the continuous testing during the development has a positive effect on the improvement of the product quality.

- **Development of functions with high complexity:** Functions with high complexity are difficult to design with a classical software development. The companies expect that model-based development helps them to develop high complex functions with viewer iterations and consequently less effort in the development, because of the possibility of early simulation.

- **Shorter development times:** The companies expect a more efficient development, because of the possibility to simulate early in the development process and to be able to generate artifacts like code or test cases for example. Shorter development times lead to an improvement in time to market and enable companies to present innovative functions earlier on the market than the competition.

It was interesting to see that cost reductions are not within the top three reasons. It could be seen that model-based development is also used because there is a trend in the industry to use it. 83% of the participants agreed that one reason for model-based development in their company is the trend in the industry to develop model-based. Especially suppliers mentioned that car producers want them to develop model-based. If they don´t do it, someone else, who uses model-based development gets the contract to develop the software for the car producer.
After having identified the main reasons why companies use model-based development, we asked them about their positive and also negative experiences with a model-based design.

Positive experiences: The experts report of the simplified communication because of the use of a function model in the SW-Design. The models provide great support in the communication with other colleagues because of the graphical design. Even colleagues from other departments or domains, who are not familiar with software development, can be involved in the software development, because of the use of models. This helps to include extra know-how in the software development. Another positive aspect is the possibility of early simulation of the function model. Model reviews, guideline checkers, Rapid Control Prototyping (RCP) and Model in the Loop – Tests (MiL) help to find errors already in the design phase. This leads to viewer iterations in the development and thereby to cost savings. The possibility to automate is seen as a further positive effect and is a key factor for a more efficient development. Besides code generation also test cases and documentations can be generated. If the generated code isn´t changed manually after the code generation the code and the function model are consistent. This is a huge benefit for the maintenance of functions.

Negative experiences: The main negative aspect, which was reported, is the extremely high process redesign costs, which have to be invested to develop model-based. The experts report of high efforts which are needed for the process redesign. They indicate that the main costs for the process redesign are not just costs for tools (although tool costs are a major cost factor), but also costs for defining a new development process, training costs for the employees and the regeneration of hand-coded projects. Another negative aspect is the high dependency from tool vendors. The reason therefore is that the tools do not offer interfaces to exchange models among different tools unless inside tools from the same tool vendor. Consequently if a company has decided for a modeling tool it is advisable to buy the rest of the tool chain from the same tool vendor, to maximize the seamlessness in the development process. Even if standardized interfaces would exist, the experts are still pessimistic about the dependency on tool vendors.

After having identified the main reasons for developing model-based and also the positive and negative experience the study participants made with model-based design we want to present the fullfield expectations of the study participants with a model-based design. This is an interesting question, because the use of model-based design is connected with certain expectations like better product quality, faster development times or cost savings for example. Therefore this question indicates how satisfied the study participants are. Figure 4 shows that the majority of participants report of fulfilled expectations. Nevertheless the majority also reports that there is still an enormous potential in model-based design which by now is not being exploited.

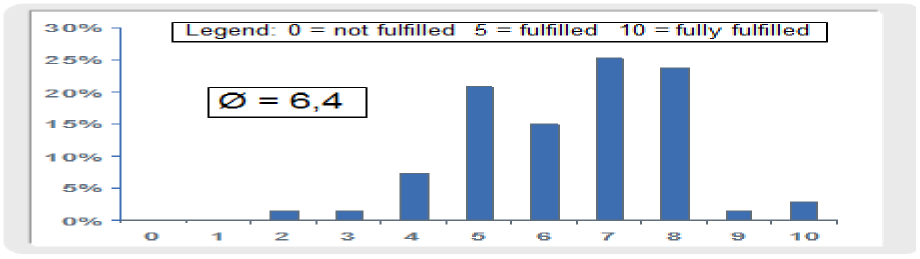


Figure 4: Fullfilled expectations

4.3 Costs- and benefits of model-based development of embedded software systems

As described in the introduction of this chapter we try to measure the influence of model-based development on costs, time and quality changes by interviewing companies how intensive they use model-based development (status quo) and how this has affected development costs, time and quality. We did this for all development phases from requirements engineering till testing. In addition we also focused on changes in the total development costs, total development time and product quality. This procedure enables us to analyze how the intensity of model-based design affects the economics. On the one hand we asked the participants about the intensity of their model-based development and on the other hand we asked about the effects on the economics. With the collected data we can conduct correlations between the conducted steps in the development process and the changes in the economics.

In this article we like to present the results of the development phase software design a development phase of major importance in a model-based development process, the development phase implementation, changes in the total view (costs, time and product quality) and concluding the influence of model-based development on the maintenance costs.

4.3.1 Software Design

The software design is the development phase where the use of model-based development leads to many changes in the development process. For example the functionality is modeled in a function model like for example in a Matlab/Simulink/Stateflow or Ascet SD model. During the development the engineer has the possibility to test this function model. The use of model-based design in the software design is the precondition to use code generation in the implementation.
We like to present how intensive the functions are being modeled or rather hand coded, how intensive the function models are being tested and how this (i.e. the modeling and simulation) effects the development costs, the development times and the product quality.

Status Quo: In average 73% of a total application functionality (i.e. only application software and no basis software) is already being modeled in a function model. There is a huge gap in the modeling degree if you take a look at the application period, in which the companies use model-based development in the software design. Companies who have just started with model-based development have a small modeling degree in the area of 5% to 20%. In comparison the companies with more than five years experience report of a modeling degree in the area around 90% to 100%.
Using model-based development in the SW-Design allows the “Frontloading” of test activities, which are conducted on code level in a classical hand coded approach already in the SW-Design. The possibility to test the function model is used widely in almost every company. Four test methods are used to test the function model: Model reviews, guideline checkers, Rapid Control Prototyping and Model- in the Loop tests. Figure 5 gives an overview how intensive these test methods are used.

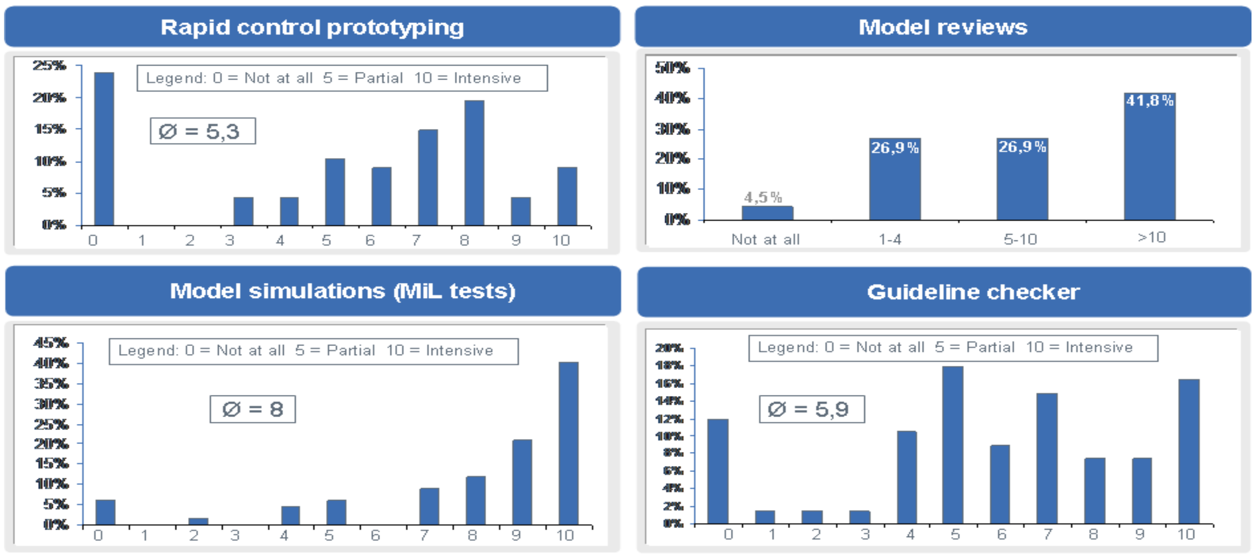


Figure 5: Overview of the intensity of tests on function model level

It can be seen that all four tests methods are being used intensively. For example 41,8% of all participants use more than ten model reviews during the whole development to check the model. Another interesting result is that the majority of the participants use MiL-Tests intensively. Later on we analyze what effect the conduction of MiL-Tests has on the number of detected errors in the module test.

<u>Changes in the economics:</u> The modeling degree has a big influence on the development costs as you can see in Figure 6. The higher the modeling degree of the function model is the higher are the costs. Study participants, who model the whole functionality in a function model report of up to 100 % cost increases in the SW-Design. Two other major aspects influence the costs in the SW-Design besides the modeling degree. The intensity the companies use RCP in the requirements analysis and the intensity of tests on function models. Companies, who use RCP in the requirements analysis, intensively report of a decline or almost no changes in the development costs, because they already frontloaded these development activities in the requirements analysis. This is a major reason for the high margin of derivation, which you can see in the figure.

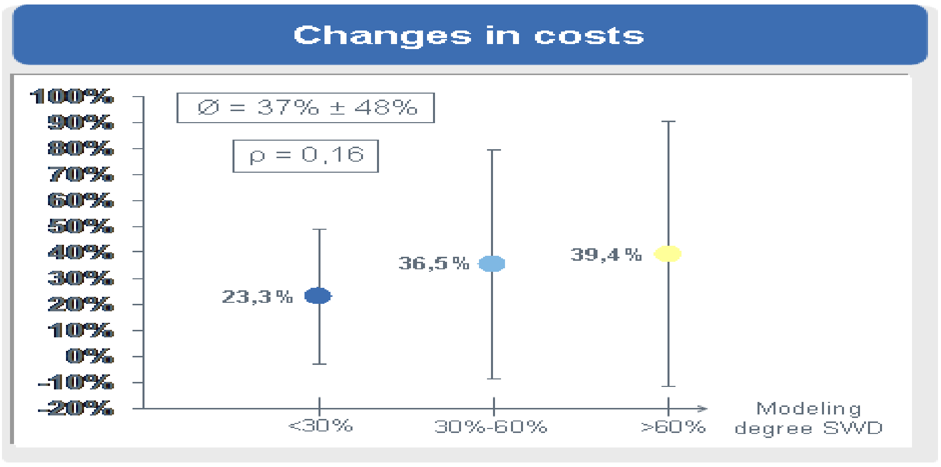


Figure 6: Correlation modeling degree in SW-Design vs. changes in costs in SW-Design

We already mentioned that the test intensity on the function model has a major influence on the cost changes in the SW-Design. Companies who use the possibility to test the function model intensively report of high cost risings in the SW-Design. These are usually the companies who also have a high modeling degree. Consequently we couldn´t analyze the single effect of the modeling degree and the test on function models on the cost changes during our data analysis. This makes sense because without a high modeling degree it wouldn´t make much sense to test the function model intensively, because it wouldn´t represent much of the total functionality.

Especially model reviews and MiL-Tests (Model in the Loop tests) are cost intensive test methods and lead to high cost increases when conducted intensively. For example if you like to conduct MiL-Tests it is advisable not just to simulate your function model with some stimuli but also to develop an environment model. Environment models have the advantage that they are more precise and complete than a selection of some stimuli. From the economic point of view you have to keep in mind that the development of an environment model is usually as expensive as the development of a function model. Consequently the development of an environment model usually only pays off, if it can be reused for different car lines. Engine models i.e. are very often used as environment models, because they are needed for the development of many functions in the car and can therefore be reused for the development of several functions from one car line and for the development of different car lines. Instead of investing the high costs to develop an environment model, lots of companies use stimuli from car tests and use them to test the function model.

The question arises what benefit a company can have if it uses the possibility of testing on function model level. The correlation of the number of detected errors in the module test and the intensity of MiL-Tests in Figure 7 shows the benefit of intensive simulations on function model level. Companies which don´t use any MiL-Tests find almost 30% more errors in the module test than companies who use MiL-Tests intensive. These errors have already been found and eliminated by the group which uses MiL-tests intensively on function model level (i.e. earlier error detection). The decline in the number of detected errors by the participants who don´t use MiL-tests, result from the other three possible test methods which can be used on function model level. This proves the benefit of testing on function model level. All participants use at least one of the four possible test methods on function model level and are able to reduce the number of detected errors in the module test, because they found these errors already on function model level.

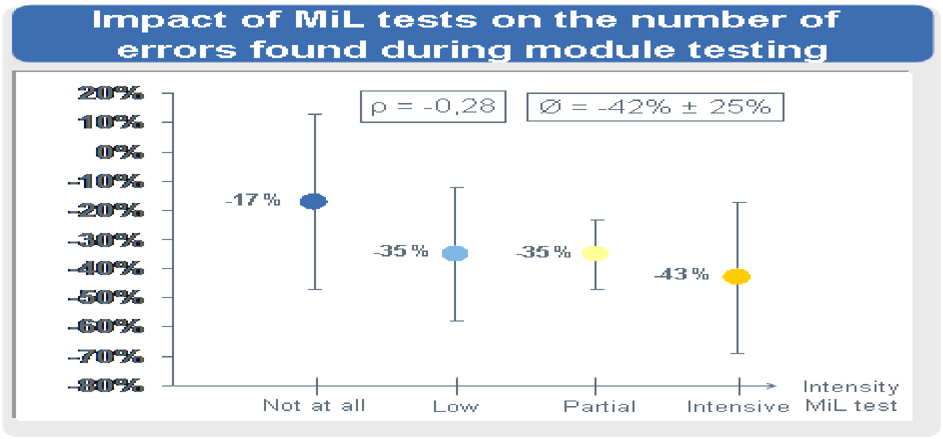


Figure 7: Impact of MiL-Tests on the number of errors found during module test

In addition we also analyzed the impact of model-based development in the software design by asking how important quality criteria in the software design changed, because of a model-based development. The positive effect can be seen in Table 1. Especially in the criteria testability, correctness and clearness very positive changes are reported.

Quality	--	-	0	+	++
Clearness	0%	1,5%	9,1%	60,6%	28,8%
Modifiability	1,5%	1,5%	30,3%	50%	16,7%
Correctness	0%	3%	12,1%	69,7%	15,2%
Detailedness	0%	4,5%	39,4%	43,9%	12,1%
Testability	0%	1,5%	13,6%	40,9%	43,9%

Table 1: Changes of the quality criteria in the software design

After describing the impact of model-based development in the SW-Design on the development costs for the SW-Design, on the impact of MiL-Tests on the detected errors in the module test and on the quality criteria in the software design, we like to present the impact of the development phase SW-Design on the total development costs. Figure 8 shows the influence. It can be seen that the more intensive model-based development is used in the software design the higher are the reported cost savings. We define intensity as the average value of the degree of modeling and the test intensity. This is a very interesting result, because the economic benefit of the frontloading activities in the SW-Design like modeling and testing can be shown. Figure 8 also shows the effect on changes in the product quality. We used the definition of product quality after DIN ISO 9126 (ISO, 1998), which defines the following criteria to be analyzed for product quality: functionality, reliability, usability, efficiency, maintainability and portability.
As in the correlation with the costs it can be seen that a small modeling and testing degree on function model level doesn´t bring any advantages compared to a classical hand coded development. With an intensity over 30% the companies begin to report of improvements. The improvements are especially high in companies with an intensity over 60% with a value of 0,88 which means strong improvement. It can be summarized that model-based development in the software design can have a very positive effect on changes in total costs and in the product quality.

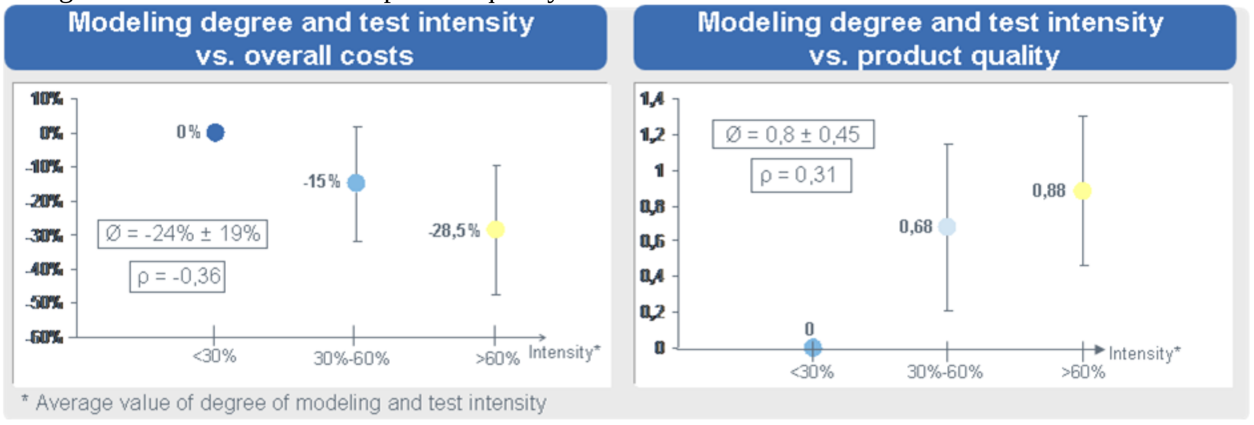


Figure 8: Correlation modeling degree and test intensity vs. overall costs and vs. product quality

In the following we present the results for the development phase implementation.

4.3.2 Implementation

Before we present the effects of code generation on the costs, we present the current status quo of model-based development in the implementation phase.

Status Quo: Almost 96% of the study participants use the automatic code generation. In average 73% of the developed application software is being developed by using a code generator. However there are major differences within the companies regarding the amount of generated code. It varies between 5%

and 100%. At least 40% of the participants report of a degree of generated code between 95% and 100%. What are the reasons for such high differences in the degree of generated code? The major influence on the degree of generated code has the modeling degree in the software design i.e. the degree of functionality, which has been modeled in a function model like a Matlab/ Simulink/ Stateflow or Ascet SD model. Besides the modeling degree the know-how of the employees in the use of a code generator and the safety relevance of the developed function play a major role. Especially at high safety relevant functions the average value of the generated code with 44% is significantly smaller than for example at uncritical functions with 77%. The reason therefore is that some companies are the opinion that the current code generators are not applicable to generate high safety relevant code. Necessary qualifications or even certifications of the code generator are aligned with very high costs, which are being avoided by the companies at the moment. In addition there are many open questions concerning the code generation like which steps in the development process are obsolete, when using a certified code generator or the question if the high costs for the certification ever amortize?
In the software design the study participants put a lot of emphasize on the test on function model level. Consequently we also like to point out the current status quo of the test activities in the implementation phase. Figure 9 shows that code reviews and SiL-Tests are used intensively by the majority of the study participants. PiL-Tests are not that intensively used. The intensive use of code reviews is on the one hand surprising, because the code is generated with a code generator but on the other hand it reflects the opinion of some study participants that they don´t trust the code generation process and therefore have to review the generated code.

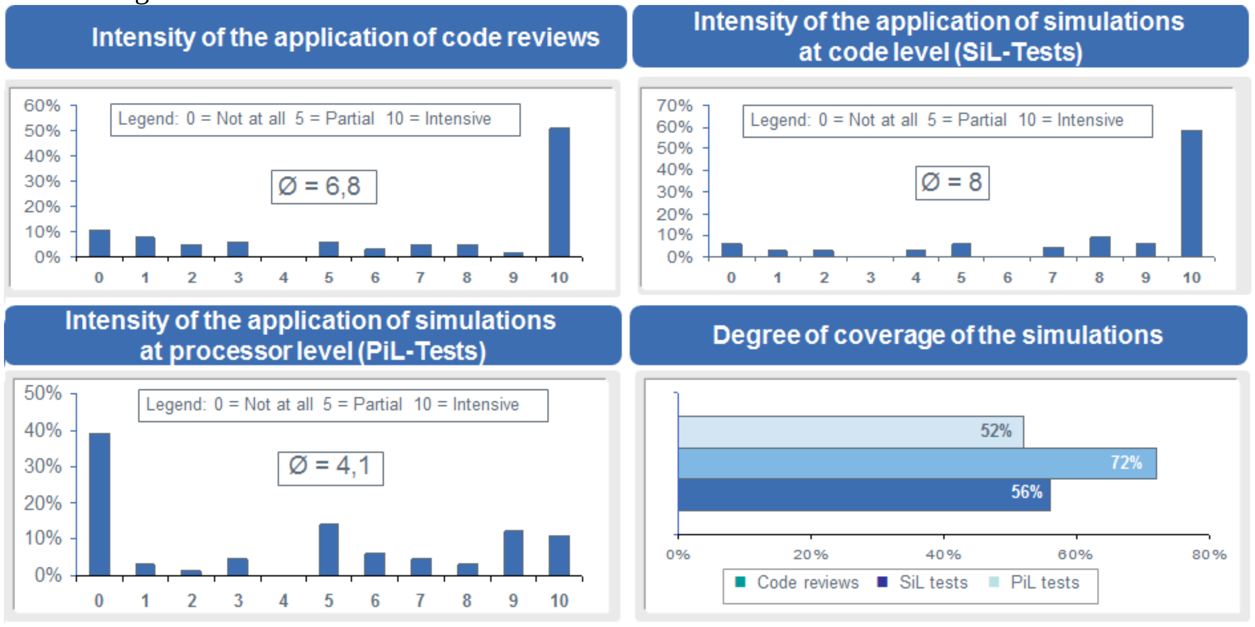


Figure 9: Status quo of the conducted test methods in the implementation phase

<u>Changes in the economics:</u> In the following we describe how the intensity of model-based development in the implementation affects the economics. Therefore we present the results of five striking correlations.

1) Correlation degree of generated code vs. cost changes in the implementation:
Aim of this correlation is to analyze the effect of the degree of generated code on the costs for the implementation. Figure 10 shows that the higher the degree of generated code is, the higher are the cost savings in the implementation. This result is not surprising, because the code generator transforms the function model, which has been developed in the software design, automatically in C-code. Hence the cost savings are effects of the frontloading of the development activities in the software design.

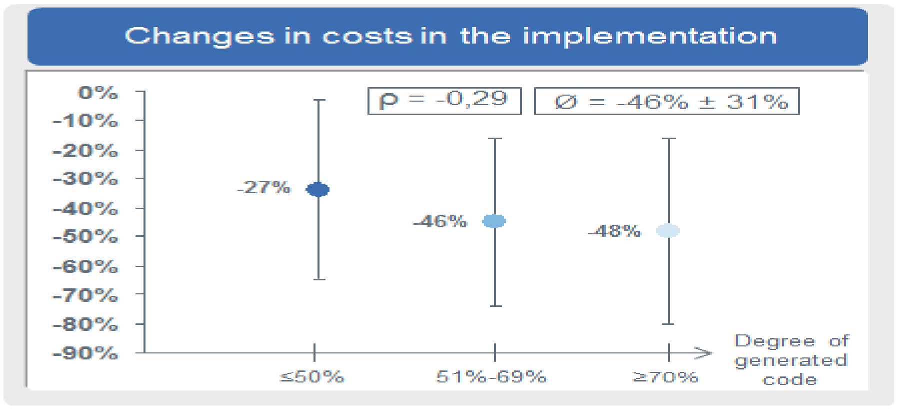


Figure 10: Correlation: Degree of generated code vs. costs in the implementation

2) Correlation degree of generated code vs. number of detected errors in the Implementation:
An interesting question is how the degree of generated code affects the number of detected errors in the implementation. Result of the correlation is that with a raising degree of generated code the number of detected errors decreases. Responsible therefore are especially the prevention of coding errors and the conducted tests in the software design (take also a look at the subsequent correlation).

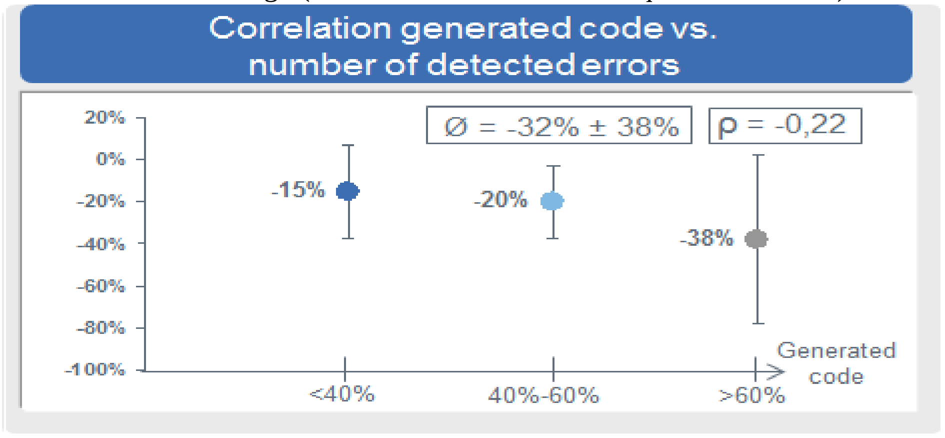


Figure 11: Correlation generated code vs. number of detected errors

3) Test in the software design vs. number of detected errors in the implementation:
This correlation provides a very interesting result and underlines the importance of tests on function model level. Participants, who test their function models intensively in the software design, report of significant less detected errors in the implementation than participants, who have tested sporadically. Especially strong is the delta between medium and intensive test activities with absolute 32% more detected errors. Consequently this correlation points out the importance of early testing (Keyword: Frontloading) in this case during software design.

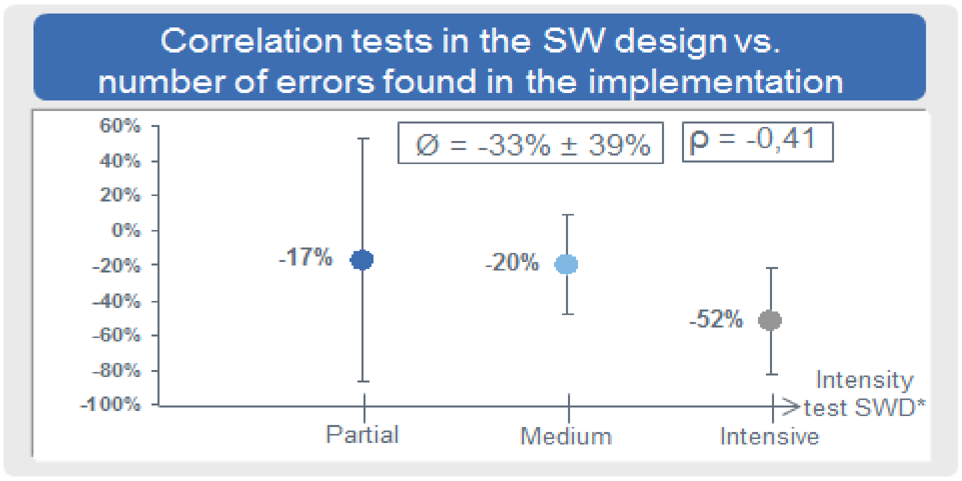


Figure 12: Correlation intensity test during software design vs. detected errors in the implementation

4) Degree of generated code and test intensity vs. changes in the total costs:
When correlating the degree of generated code and the intensity of code generation and test activities in the implementation with the total costs a trend can be seen that a raising intensity leads to a decrease in the costs.

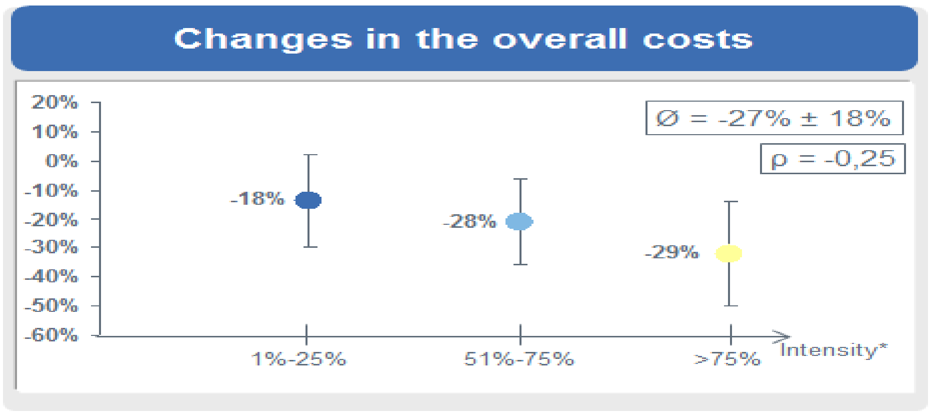


Figure 13: Correlation generated code and test intensity vs. changes in the total costs

The data analysis showed that the fundament for cost savings in the total view is laid in the software design. Participants with a high degree of generated code generally have developed a function model with a high modeling degree. For the most part these are also the participants, who have conducted intensive tests on function model level like MiL-tests and consequently also SiL-tests. The reason therefore is that conducted MiL-tests can be reused on SiL-test level, as the aim of SiL-tests is to ensure that the generated code behaves exactly the same as the function model.

5) Degree of generated code and test intensity vs. changes in the product quality criteria:
The activities in the implementation only have a small influence on the product quality. However it can be seen that even with a small intensity (<40%) strong improvements (value of 0,7) can be seen.

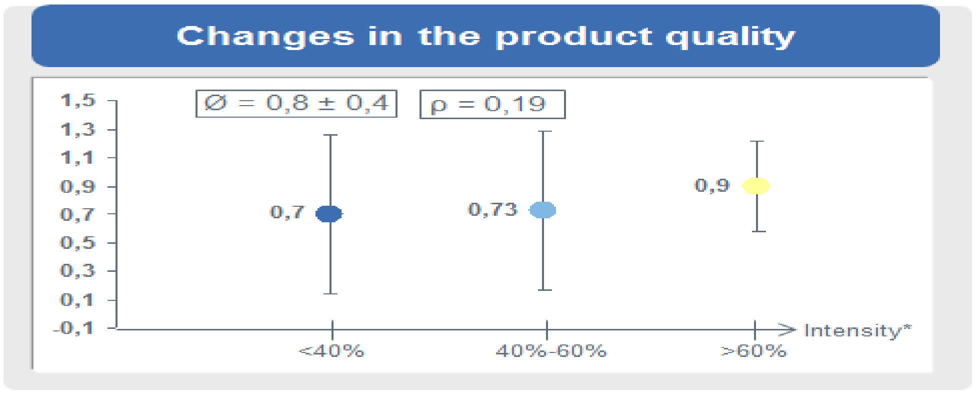
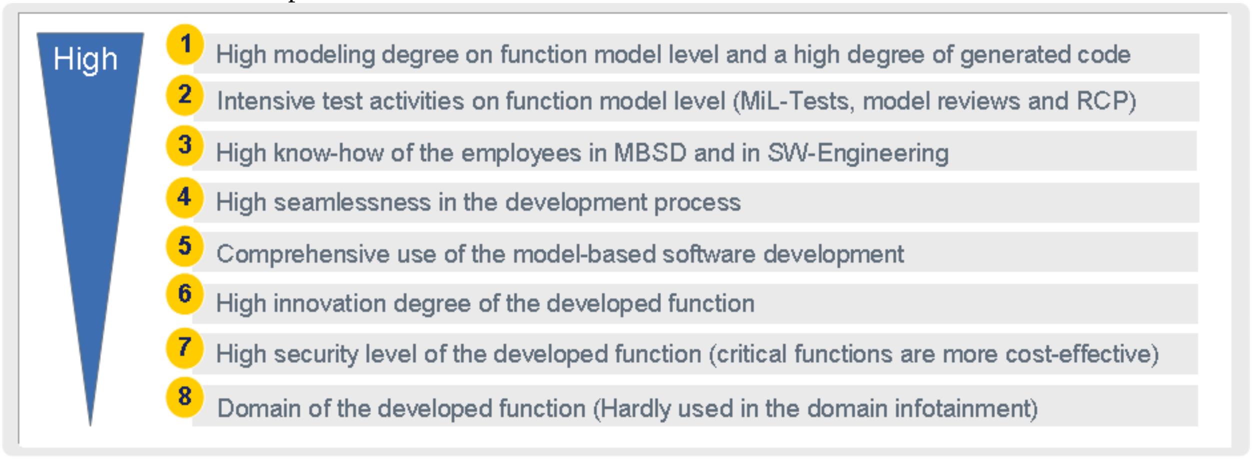


Figure 14: Correlation generated Code and test intensity vs. changes in the product quality criteria

4.3.3 Total view

In average the study participants report of cost savings around – 27% and time savings around -36%. The data analysis exposed that the statements of the study participants concerning the changes in the total cost partially deviate considerably. Some study participants reported of considerable cost savings whereas others report of considerable cost increases. The answers ranged from a minimum value of -80% (cost savings) till a maximum value of +20% (cost rising). With a detailed data analysis we found out, that study participants who use model-based development since a short time only report of cost increases. These result from not well-rehearsed development processes and too little know-how of the employees in the use of the development tools. Companies with more than one year experience already report of cost savings, in general. However, also in this group there are huge differences in the statements about savings. We analyzed the reasons for these differences and found out that the cost savings depend strongly on several factors. In the following we like to present the top influence factors on the costs, time and quality changes because of model-based development.

If you like to sum it up as an elevator pitch you can say that a high modeling degree of the function model in the software design and a high degree of generated code, intensive test activities on function model level and the know-how of the employees are the three main influencing factors on the economics of model-based software development. The study participants, who have high values for these three criteria, reported of the highest total cost and time savings and the best improvements in product quality. It was interesting to see that the domain of the function doesn´t have a big influence on the costs. Except in the domain infotainment, where model-based development is currently hardly used, the statements about costs were likewise independent of the domain.



1 High modeling degree on function model level and a high degree of generated code
2 Intensive test activities on function model level (MiL-Tests, model reviews and RCP)
3 High know-how of the employees in MBSD and in SW-Engineering
4 High seamlessness in the development process
5 Comprehensive use of the model-based software development
6 High innovation degree of the developed function
7 High security level of the developed function (critical functions are more cost-effective)
8 Domain of the developed function (Hardly used in the domain infotainment)

Figure 15: The top 8 identified influence factors on costs, time and quality because of model-based development

In the introduction we pointed out that our motivation is to analyze how the economics change because of model-based design and also to take the process redesign costs into account. We asked the companies if the costs for the process redesign have already amortized and if yes, how long it took? 65% of the companies report of amortized process redesign costs. In average it took them 3 ½ years. This is of course dependent on the number of conducted model-based projects. The more projects are conducted the faster the costs can be amortized.

We also asked the study participants if it is easier to introduce model-based development in a small or a large company. The majority believes that the introduction of a model-based design can be difficult independent of the size of the company. The size only plays a role because the companies face other problems. Small companies often don´t have the money to afford the redesign costs, especially the high tool costs. Large companies have the money for the process redesign, but face the problem that there is often a barrier in the company to change their established development processes. Consequently the introduction of model-based development is a challenge independent of the size of the company.

4.3.4 Influence of MBSD on maintenance costs

For us it was interesting not only to analyze the effect of model-based development on the development costs but also on the maintenance costs. We asked the study participants about the relevance model-based development has on the maintenance costs. 24% think that it has a very high and 67% that it has relevance on the maintenance costs. That the effect is positive can be seen when taking a look at the reported changes in the maintenance costs. In average the overall maintenance costs reduced about 15% compared to a classical development. Savings around 20% are reported when it comes to enhance the functionality of the developed function for example for a face lift of a car. One main reason therefore is that the function model can easily be added with new functionality.

When correlating the modeling degree and the degree of generated code with the changes in the maintenance costs we found out that the higher the modeling degree and the degree of generated code is the higher are the cost savings in the maintenance. This is a further indication that model-based development has a positive effect on maintenance costs.

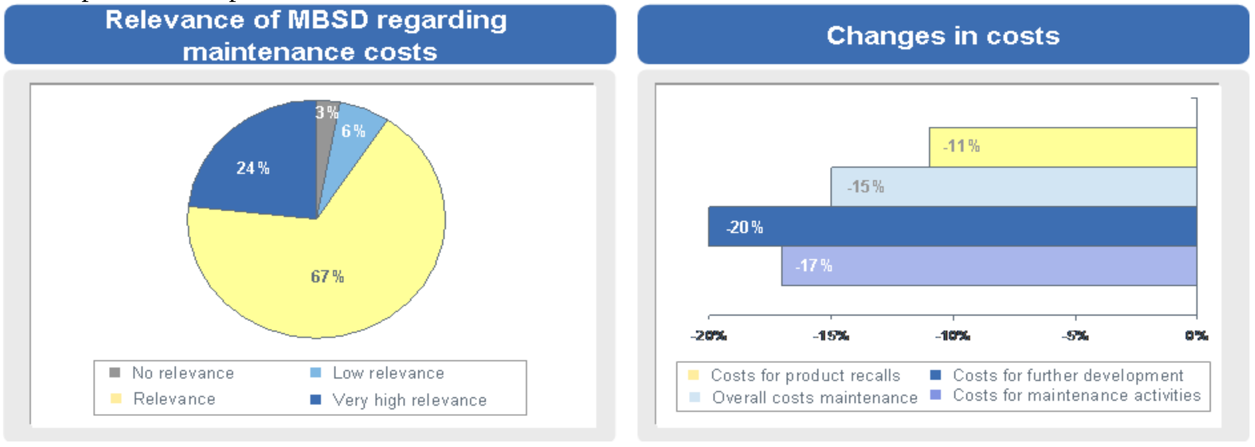


Figure 16: Influence of model-based development on the maintenance costs

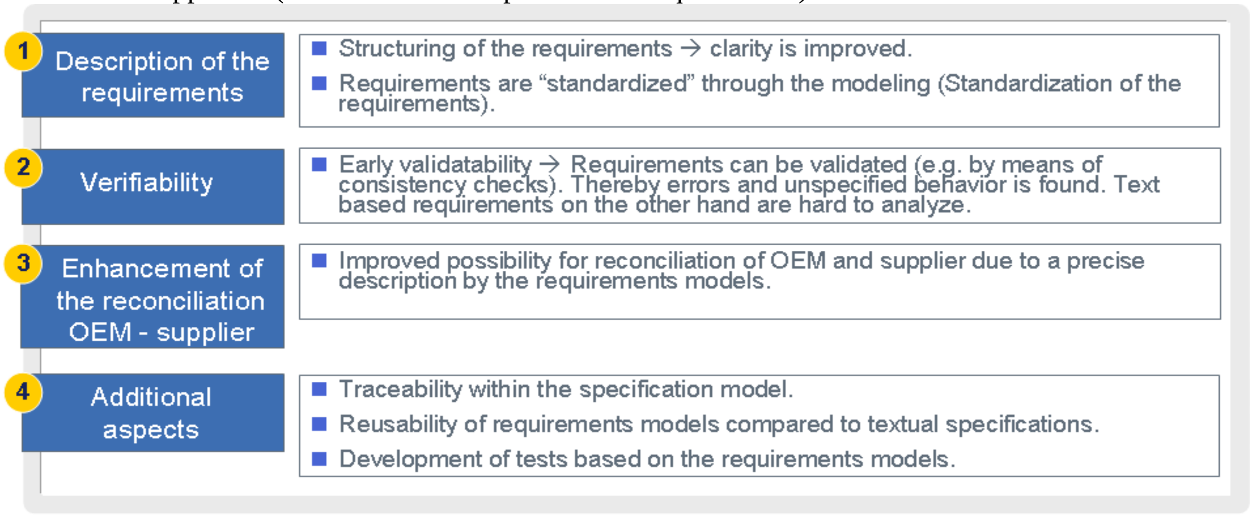
4.4 Potential of model-based requirements and architecture modeling

The results of the study show, that model-based development is currently used widely in the development phases SW-Design and implementation. This was also a result of the case studies (see point 2 in figure 1), which we conducted prior to the global study. More information about the results of the case study can be found in (Kirstan & Zimmermann, 2010). That's why we also included questions in the questionnaire for the global study about current challenges in requirements analysis and architecture design and the valuation of the possible benefit to also use a model-based approach in these development phases. These challenges are afterwards being mapped with the expectations our study participants have of modeling in these development phases to see if a model-based approach can help to master the current challenges. The results are presented in the following.

4.4.1 Requirements engineering

To capture the requirements right (i.e. complete and consistent) is the basis for the development of software. The study participants report that the description and the verifiability of the requirements and the reconciliation OEM – supplier are currently the biggest challenges they face in the requirements analysis. The challenges in describing the requirements are firstly to find an adequate degree of abstraction and level of detail. Secondly to control the system complexity is seen as a further challenge. Study participants report that up to 15000 requirements per function has become a normal value. Besides the question how to describe the requirements, it is also a challenge how to verify them. Text based specifications are error prone. Therefore there is a huge demand in the industry to validate them for example by tools. This can only be done if the requirements are described in a formal description language. The reconciliation between OEM and supplier is another challenge, especially in the documentation of the requirements. Not all of the requirements are documented by either the supplier or the OEM, which leads to an additional workload.

Model-based requirements engineering can be a solution to these problems. We asked the study participants which additional value they see in model-based requirements engineering. Surprisingly the answers of the participants could be grouped into the three groups, which have been reported as challenges: Description of requirements, verifiability and enhancement of the reconciliation OEM – supplier. In addition they reported some additional aspects where they also see a potential of using a model-based approach (i.e. a formal description of the requirements).



1 Description of the requirements
- Structuring of the requirements → clarity is improved.
- Requirements are "standardized" through the modeling (Standardization of the requirements).

2 Verifiability
- Early validatability → Requirements can be validated (e.g. by means of consistency checks). Thereby errors and unspecified behavior is found. Text based requirements on the other hand are hard to analyze.

3 Enhancement of the reconciliation OEM - supplier
- Improved possibility for reconciliation of OEM and supplier due to a precise description by the requirements models.

4 Additional aspects
- Traceability within the specification model.
- Reusability of requirements models compared to textual specifications.
- Development of tests based on the requirements models.

Figure 17: Expected potential of model-based requirements engineering

It can be seen that especially the point description of the requirements and the verifiability are strongly connected with the use of formal requirement models. Without them the verifiability of the requirements for example is difficult or even not possible.

4.4.2 Architecture design

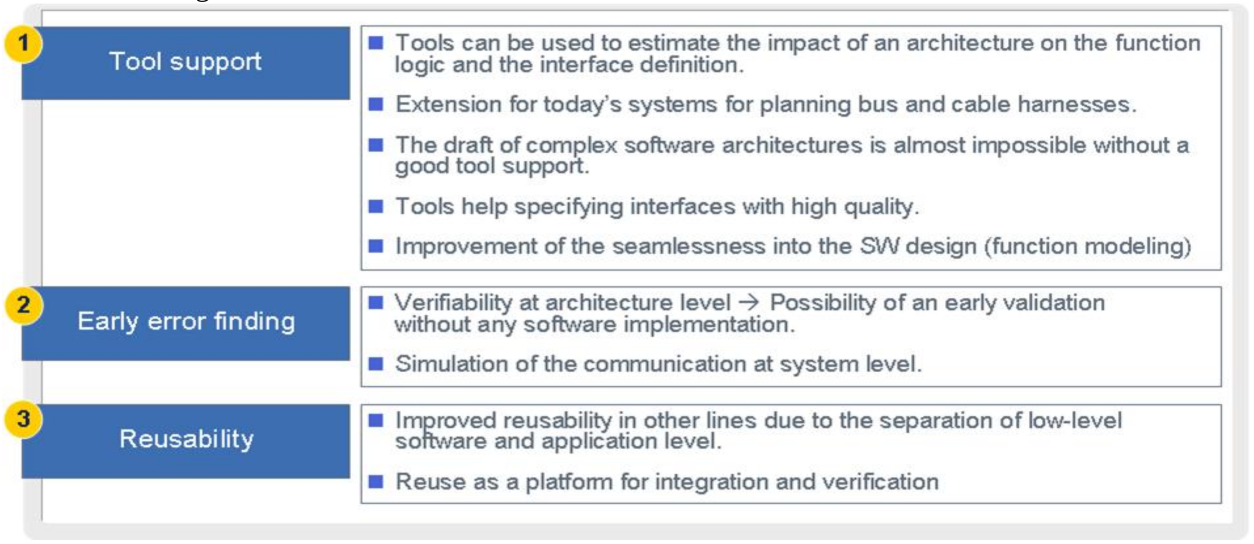
The architecture design is a very important development phase, because it has an impact on the integration, maintainability and the allocation of the work load. The challenges the companies face in architecture development can be broken down into three points. These challenges focus also on the current modeling tools, because some study participants use them already for the series development:

- **Tool support:** There are two challenges concerning the tools. First of all the tools do not offer the necessary functionality. This is especially the case when using informal tools like PowerPoint, Excel, Visio etc.. Secondly the seamlessness to the modeling tools is missing. Consequently the results of the architecture design have to be manually transformed in a function model and that can lead to mistakes not just because there can be errors in drafting the architecture in the function model, but also that the software designer changes the architecture without notice.

- **Earlier error finding:** The participants would like to test their architecture design more intensively for example with simulations to find architecture errors earlier in the development process. This is a very important point, because usually errors in the architecture are detected in the integration phase (i.e. very late in the development phase). To correct these errors is very cost intensive.

- **Reusability:** Strategies for the modularization of the architecture are often missing. This hampers reuse of architectures. The reuse is a main aim of the companies for example via product lines or in the field of variant management.

The question arises which improvement potential the experts see in architecture modeling? Can it handle all the challenges or at least help to solve some of them? Figure 18 summarizes the answers of the study participants. The answers could also be grouped into the same three aspects like the challenges in the architecture design.



1 Tool support
- Tools can be used to estimate the impact of an architecture on the function logic and the interface definition.
- Extension for today's systems for planning bus and cable harnesses.
- The draft of complex software architectures is almost impossible without a good tool support.
- Tools help specifying interfaces with high quality.
- Improvement of the seamlessness into the SW design (function modeling)

2 Early error finding
- Verifiability at architecture level → Possibility of an early validation without any software implementation.
- Simulation of the communication at system level.

3 Reusability
- Improved reusability in other lines due to the separation of low-level software and application level.
- Reuse as a platform for integration and verification

Figure 18: Expected potential of model-based architecture design

In requirements engineering as well as in the architecture design it could be shown that the challenges and the expectations match. That means that the majority of the study participants are convinced that model-based development in these development phases can solve or at least help to master their current challenges in requirements engineering and in the software architecture design.

4.5 Threads to validity

The study was conducted with highest carefulness. Nevertheless threads to validity have to be mentioned which could have influenced the results of the study.

1. The number of filled out questionnaires: We received 67 answered questionnaires. Therefore almost 180 people were involved in answering the questionnaire. In total we invited more than 850 people to take part in the study. Reasons like confidentiality and the effort to answer the questionnaire (2 hours were needed in average to answer it completely) were the main reasons

why the number of filled out questionnaires were not higher. Nevertheless we were able to win several large car producers, suppliers and technology consulting companies worldwide as participants, which have a huge experience in model-based development. Their expertise ensures the validity of the study.

2. The given data by the study participants are approximations: Because a function for series development is not developed hand-coded as well as model-based the participants could only give approximations how the costs, time and quality change because of model-based development. Anyway the approximations are a good indicator, because all the companies reported about functions which they have developed hand-coded for a previous car line.

3. The design of the questionnaire: The design of the questionnaire may have influenced the answers of the study participants. We tried to minimize this thread by using best practices in the design of the questionnaire and validating the questionnaire by various experts before the questionnaire was sent to the participants.

4. The selection of the participants: The majority of the participants were personally invited to take part in the study. There is a thread that only companies took part which have a higher degree of model-based development than the industry average. By analyzing the data of the study participants we found out that the participants range from companies which have just changed their development process till companies which are using model-based development for over 10 years.

5. POTENTIAL OF MODEL-BASED DEVELOPMENT IN THE DEVELOPMENT PHASES REQUIREMENTS ENGINEERING AND SOFTWARE ARCHITECTURE

As we have seen in the previous sections, model-based development has proven its value for the SW-Design and Implementation phase. As a result, in domains like automotive or aeronautic systems, its application has become state of the art. This is reflected in the definition or adaptation of recent development standards like the ISO 26262 (ISO, 2010) or the up-coming DO-178C. Unsurprisingly, these standards specifically address model-based development for software unit design and implementation, adapting the required quality assurance techniques. For example, “Model Inspection” and “Model Walk-Through” are highly recommended practices in these phases.

Nevertheless, about 80% of all fatal errors and more than 50% of all heavy errors are made in the requirements specification or architecture definition phase (Jones, 1991). Consequently, these standards mentioned above already address the importance of models in the requirement specification or architecture phase. For example, (ISO, 2010) highly recommends the “use of unambiguous graphical representations” in general. Furthermore, (ISO, 2010) also highly recommends the use of “semi-formal notations” for requirements specification as well as for the software architectural design for systems with higher criticality. It specifically mentions the use of “executable models” for the verification of requirements and architecture.

Due to their general nature and their intention to define the currently achievable state-of-the-art, the recommendations provided by standards like (ISO, 2010) are not very explicit about the potential and benefits of model-based development outside SW unit design and implementation. However, current research and development has proven the following model-based techniques to be both doable and beneficial:

1. Specification of functional requirements: The current use of models in the requirements phase is often limited to structural aspects of the system under development (e.g., specification of the bus interface). The behavioural aspects are generally only described informally, using (structured) natural language,

complicating their validation and verification. Here, precise requirements – in textual form of structured tables as well as graphical form like sequence diagrams – can help to establish a high quality of the requirements constructively as well as analytically (Schätz, 2009). Furthermore, such specifications also provide a basis for quality assurance in later steps, e.g., in form of test cases, which can be executed in text execution frameworks.

2. Component integration: In the current model-based development process, models are generally used to design and implement individual software components including their interfaces. However, without the use of an overall architecture, integrating all these components, issues like inconsistencies between these components and their interfaces may arise. Prominent examples like the loss of "Mars Climate Orbiter" space probe due to a conflict of metric- and imperial-based interfaces show the consequences of those inconsistencies. Here, an integrated architectural model can avoid these sources of inconsistencies (Schätz, 2009).
3. Unit integration: Component integration ensures consistency on the level of the logical architecture. However, gaps in current tool chains often leave room for inconsistencies when mapping the logical architecture to the SW level, for example when interpreting a bit-signal differently when communicated between the sender and receiver components. By using a model-based approach combining models of the logical and technical architecture as well as the deployment mapping from the former to the later, these inconsistencies can be avoided.
4. Test case generation: The use of models for the specification of system- or component-level requirements allows to immediately reuse these specifications as test cases. However, these models can even be exploited further: Using corresponding synthesis techniques, additional test cases can be generated automatically to provide a better coverage of the specified requirements or to obtain additional robustness tests.

Besides these applications, models can furthermore provide support for several additional aspects of the development of embedded systems, especially in the automotive domain. Examples for these aspects include: the *automation of the design-space exploration,* by synthesizing development artefacts like safe task and communication schedules; the *modeling of variability aspects* in function, architecture, and software as well as hardware to support the definition of product lines, and their use to generate specific product configurations from them; the provision of *model-based diagnosis*, both on-board and off-board, by deducing possible fault locations based on models of the intended behaviour; the extension of the architectural and behavioural models to include the explicit *modeling of failure assumptions*, supporting the construction of model-based assurance cases.
Finally, by adding explicit models of the environment to the development process, even more advanced techniques can be added, supporting, e.g., an early validation of requirements as well as a virtual commissioning of the system under development by use of simulated environment.

6. CONCLUSION

Main results of the study are that model-based development can bring significant cost savings, but only with a "well-chosen" model-based development and an established development process with defined interfaces and role allocations. Otherwise model-based development can be much more expensive than a hand-coded manual software development. Different factors have been analyzed which have a big impact on the changes of the development costs because of model-based development. During the development phase software design for example the intensity of testing on function model level is a major key success factor besides a high modeling degree during model-based development. The earlier the models are being tested, the earlier errors are being found. Besides the time when testing is started, the intensity of testing is very important. We found out that the more intensive function models were tested, the more savings

could be achieved in the total development costs. One reason is that the study participants that tested their function model intensively report a major decline in the number of detected errors in the module and SW-SW-integration test, whereas the study participants, who only test sporadically, have just a slight decline in the number of detected errors in the module and SW-SW-integration test.
In addition we also analyzed further potentials of model-based development. Therefore we took a look at the current challenges on the development phases requirements engineering and software architecture and asked the study participants what potential they see if a model-based approach is also used in these development phases. As a result we could see that the majority of the participants see a huge benefit in using a model-based approach in both development phases.
Furthermore we presented the potential of model-based development in the development phases requirements engineering and architecture design from a theoretical point of view. Thereby we presented current research issues in this field, which can be a further potential to improve a model-based design.

It was interesting to see that the results of our study match with results of the studies presented in the related work chapter. Beginning with the motivation companies have in using model-based development till statements to productivity changes and quality improvements.

To sum it all up the study shows that model-based design has a huge **benefit** *in a domain driven by functional evolution as well as by hardware revolutions*, as it can bring cost and time savings in the development and in the maintenance, the quality of single development artifacts and the product quality in total improve, and in addition the developed function models can be easily reused in different car lines. With the detailed data analysis we found out that there is a huge potential within the companies to optimizes the cost and benefit ratio of their model-based design. In many companies the potential of model-based design is not fully used because they develop just punctual model-based. As a consequence they hardly see any benefit of a model-based software design. Primary the consequent use of model-based development pays off. With the extension of the model-based development also on other development phases like architecture design, requirements engineering and model-based testing there can be an additional potential to raise the economics of model-based development even more.

REFERENCES

Asadi, M. & Ramsin, R. (2008). MDA-Based Methodologies: An Analytical Survey. In: I. Schieferdecker and A. Hartman (Eds.): ECMDA-FA 2008, LNCS 5095, pp. 419-431

dSPACE. (2010). From dSPACE: Overview TargetLink: http://www.dspace.de/de/gmb/home/products/sw/pcgs/targetli.cfm

Dzidek, W., & Arisholm, E., & Briand, L. (2008). A Realistic Empirical Evaluation of the Costs and Benefits of UML in Software Maintenance. IEEE Trans. Software Eng. 34(3): 407-432

Fey, I., & Stürmer, I. (2008). Quality Assurance Methods for Model-Based Development: A survey and assessment. *SAE 2007 Transactions Journal of Passenger Cars: Mechanical Systems.* Detroit.

Fieber, F., Regnat, N., & Rumpe, B. (2009). Assessing usability of model based development in industrial projects. In T. Bailey, R. Vogel, & J. Mansell (Hrsg.), *CTIT Workshop Proceedings Series WP09-07*, (S. 1-9). Enschede

IABG. (2009). From IABG: Das V-Modell XT: http://v-modell.iabg.de

ISO 9126 (1998). "IEEE Standard for Software Test Documentation," New York

ISO 26262 (2010). „Road vehicles – Functional safety"

Jones, C., (1991). Applied Software Measurement. McGraw

Kirstan, S., & Zimmermann, J. (2010). Evaluating costs and benefits of model-based development of embedded software systems in the car industry – Results of a qualitative Case Study; In Proceedings Workshop C2M:EEMDD „From code centric to model centric: Evaluating the effectiveness of MDD" ECMFA 2010 Paris

Lederer, D. (2002). The key to success: Seamless Systems Engineering Process- an urgent task. *Elektronik Automotive*. http://www.vectorconsulting.de/portal/medien/cmc/press

Mercer Management Consulting und Frauenhofer Gesellschaft. (2004). *Future Automotive Industry Structure 2015*

Mohagheghi, P., & Dehlen, V. (2008). Where is the Proof? - A Review of Experiences from Applying MDE in Industry. In I. S. Hartman (Hrsg.), *ECMDA-FA 2008* (S. 432-443). Berlin: Springer.

Murphy, B., Wakefield, A., & Friedman, J. (2008). "Best Practices for Verification, Validation and Test in Model-based Design", www.mathworks.com/mason/tag/proxy.html?dataid=11031

Smith, P., Friedmann, S., & Prabhu, S. M. (April 2007). Best Practices for Establishing a Model-Based Design culture. (S. International, Hrsg.) *Sytems Engineering 2007*.

Schätz, B. et al. (2004). Model-based Software and Systems Development – A White Paper. http://www4.in.tum.de/~schaetz/papers/ModelBased.pdf

Schätz, B. (2009). *Model-Based Development of Software Systems: From Models to Tools.* Habilitationsschrift, Technische Universität München, München.

The Mathworks. (2010). User Stories from costumers: http://www.mathworks.com/control-systems/userstories.html

ADDITIONAL READING SECTION

Broy, M. (1997). Requirements Engineering for Embedded Systems. *Proceedings FemSys'97*. München.

Broy , M., Feilkas, M., Grünbauer, J., Gruler, A., Harhurin, A.& Hartmann, J. (2008). *Umfassendes Architekturmodell für das Engineering eingebetteter Software-intensiver Systeme*. Technical Report, Technische Universität München, München.

Broy, M., Huber, F., Schätz, B., Philips, J., Prenniger, W.& Pretschner, A. (2004). *Model-based Software and Systems Development - A White Paper*. From http://www4.in.tum.de/~schaetz/papers/ModelBased.pdf

Broy, M., & Rumpe, B. (2007). Modulare hierarchische Modellierung als Grundlage der Software- und Systementwicklung. *Informatik Spektrum Volume 30, Number 1*, (S. 3-18). München.

Broy, M., Feilkas, M., Herrmannsdoerfer, M., Merenda, S., & Ratiu, D. (2010). Seamless Model-Based Development: From Isolated Tools to Integrated Model Engineering Environment. *Proceedings of the IEEE, Volume 98 Issue 4*, (S. 526-545).

Bowen, J. P. (2005). Ten Commandments Revisited: A Ten-Year Perspective on the Industrial Application of Formal Methods. *Proceedings of the 10 th Workshop on Formal Methods for Industrial Critical Systems (FMICS 2005)* (S. 8-16). Málaga: ACM Press.

Erl, H.-P., & Kirstan, S. (2007). *Kosten- und Nutzen modellbasierter Softwareentwicklung im Automobil.* Studie, Arthur D. Little, München.

Juergens, E., Deissenböck, F., Hummel, B., & Wagner, S. (2009). Do Code Clones Matter? *ICSE '09: Proceedings of the 31st International Conference on Software Engineering*, (S. 485-495). Vancouver.

Juergens, E., Deissenboeck, F., Domann, C., Feilkas, M., Hummel, B., & Schaetz, B. (2010). Can Clone Detection Support Quality Assessments of Requirements Specifications? *ICSE '10: Proceedings of the 32nd International Conference on Software Engineering*, (S. 79-88). Kapstadt.

Kalix, E., & Bunzel, S. (2005). Integration modellbasierter Entwurfsverfahren in Softwareverifikation und –entwicklungsprozesse. *Tagungsband ASIM/GI-Fachgruppe 4.5.5 „Simulation technischer Systeme"*, (S. 111-124). Berlin.

Kalix, E., & Schütte, O. (2007). Obstacles to the Adoption of Model-based Design within the Automotive Supply Industry. *Tagungsband des Dagstuhl-Workshops 2007*, (S. 29-34). Braunschweig.

Kofler, T., Herrmannsdörfer, M., Merenda, S., Ratiu, D., & Thyssen, J. (2010). Model-based Development Tools for Embedded Systems in the Industry - Results from an Empirical Investigation. *ENVISION 2020 '10: Erster Workshop zur Zukunft der Entwicklung softwareintensiver, eingebetteter Systeme.* Paderborn.

Kramer, A. (2006). Modellbasiertes Testdesign - Ein Erfahrungsbericht. In R. B. Heinrich C. Mayr (Hrsg.), *Proceedings Modellierung 2006 LNI 82 GI 2006*, (S. 265-268). Innsbruck.

Kühl, M., & Reichmann, C. (Oktober 2007). Modellbasierte Architekturentwicklung von E/E-Systemen. *Automobil Elektronik*, S. 22-24.

Lamberg, K., & Beine, M. (2005). Test Methoden und Tools in der modellbasierten Funktionsentwicklung. *Proceedings der Jahrestagung der ASIM/GI-Fachgruppe 4.5.5 `Simulation technischer Systeme'*, (S. 30-39). Berlin.

Leupers, R. (2000). Umfrage über Methoden und Techniken einer effizienten Codegenerierung zur Verbesserung der Produktivität in der Codegenerierung. *Proceedings of the 13th International Symposium on System Synthesis (ISSS'00)*, (S. 173-178). Madrid.

Michailidis, A., Spieth, U., Ringler, T., Hedenetz, B., & Kowalewski, S. (2010). Test front loading in early stages of automotive software development based on AUTOSAR. *DATE '10 Proceedings of the Conference on Design, Automation and Test in Europe*, (S. 435-440). Leuven.

Mohagheghi, P., & Dehlen, V. (2007). An Overview of Quality Frameworks in Model - Driven Engineering and Observation on Transformation Quality. *Proceedings of the 2nd Workshop on Quality in Modeling Co-located with MoDELS 2007*, (S. 3-17). Nashville.

Moretti, G. (2007). *Best Practices for Adopting Model-Based Design in Electronic System Development.* From www.mathworks.com/mason/tag/proxy.html

Pretschner, A. (2005). One Evaluation of Model-Based Testing and its Automation. *Proceedings 27th International Conference on Software Engineering (ICSE'05)*, (S. 392-401). St. Louis.

Pretschner, A., & Leucker, M. (2005). Model-Based Testing – A Glossary. In M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, Pretschner, & Alexander, *Model-Based Testing of Reactive Systems.* Springer Verlag.

Schätz, B. (2009). *Model-Based Development of Software Systems: From Models to Tools.* Habilitationsschrift, Technische Universität München, München.

Schätz, B., Fleischmann, A., Geisberger, E., & Pister, M. (2005). Modellbasierte Anforderungsmodellierung in AutoRAID. *Proceedings of Informatik 2005 Workshop Modellbasierte Qualitätssicherung.* Bonn: Springer.

Schätz, B., Pister, M., & Wisspeinter, A. (2003). Von Anforderungsanalyse in der modellbasierten Entwicklung am Beispiel von AutoFocus; Vortrag bei dem GI-Treffen Fachgruppe RE, 27/12/03 - 28/12/03: http://www4.in.tum.de/~schaetz/slides/Schaetz-GI-RE-271203.pdf abgerufen

Schätz, B., Pister, M., & Wisspeintner, A. (2004). Anforderungsanalyse in der modellbasierten Entwicklung am Beispiel von AutoFocus. *Softwaretechnik-Trends 24(1).*

Schätz, B., Pretschner, A., Huber, F., & Philipps, J. (2002). Model-Based Development of Embedded Systems. In J.-M. In: Bruel, & Z. (. Bellahsene (Hrsg.), *Advances in Object-Oriented Information Systems OOIS 2002 Workshops.* Montpellier.

Schlosser, J. (2005). Architektursimulation von verteilten Steuergeräten, Technische Universität München, München, Doktorarbeit.

van Lamsweerde, A. (2000). Formal Specification: a Roadmap. *Proceedings of the Conference on The Future of Software Engineering*, (S. 147 - 159). Limerick.

Vitkin, L., Dong, S., Searcy, R., & BC, M. (2006). *Effort Estimation in Model-Based Software Development.* Technical Paper, SAE Technical Paper Series, In-Vehicle Software & Hardware Systems (SP-2028), Detroit.

von der Beeck, M., Braun, P., Rappl, M., & Schroder, C. (2002). Model based requirements engineering for embedded software. *Proceedings of the 10th Anniversary Joint IEEE International Requirements Engineering Conference (RE'02)*, (S. 92). Essen.

Weber, M., & Weisbrod, J. (2003). Requirements Engineering in Automotive Development – Experiences and Challenges . *IEEE Software, 20(1), 2003*, (S. 16-24).

Wohlgemuth, F., Dziobeck, C., & Ringler, T. (2008). Von Autosar im Entwicklungsprozess: Vorgehen bei der Serieneinführung der modellbasierten AUTOSAR-Funktionsentwicklung: http://www.dspace.de

Ziegenbein, D., Freund, U., Braun, P., Sandner, R., Bauer, A. & Romberg, J. (2005). AutoMoDe - model-Based Development of Automotive Software. *Proceedings of the 2005 Conference on Design, Automation and Test in Europe (DATE)*, (S. 171-176). München

Key Words: Model-based development, cost and benefit analysis, function model, modeling degree, frontloading, simulation, economics, empirical study, return on investment

Model based development: See definition in the main chapter.
Cost and benefit analysis: The aim of a cost and benefit analysis is to analyze what an investment in a new product or technology would cost and which benefit i.e. cost savings it can bring.
Function model: The function model is the model which is being used during the SW-Design. In the function model the functional logic is being modeled. The function model will be used in the implementation phase for the code generation.
Modeling degree: Degree of functionality which has been modeled in a function model like a Matlab/Simulink/Stateflow or ASCET-SD model.
Frontloading: In our case frontloading means the possibility to shift the testing activities in earlier development phases compared to a classical hand-coded design (i.e. from module test into the SW-Design).
Simulation: Simulation is a technique to test i.e. the function model if it is modeled correctly.
Economics: We define economics as the factors of the magical triangle of the project management: Cost, time and quality.
Empirical study: An empirical study is a survey, where companies are asked which quantitative changes they have i.e. in development costs. The data, which has been collected via the survey will be analyzed via statistical methods.
Return on investment: The aim of a return on investment analyses is to find out how long it takes until the investments have been amortized.