Third International Conference on Software Testing, Verification, and Validation Workshops

Modelling requirements to support testing of product lines

Christopher Robinson-Mallett
Berner & Mattner Systemtechnik GmbH
Berlin, Germany
robinson-mallett@berner-mattner.com

Matthias Grochtmann
Berner & Mattner Systemtechnik GmbH
Berlin, Germany
matthias.grochtmann@berner-mattner.com

Joachim Wegener
Berner & Mattner Systemtechnik GmbH
Berlin, Germany
joachim.wegener@berner-mattner.com

Jens Köhnlein
Daimler AG
Sindelfingen, Germany
jens.koehnlein@daimler.com

Steffen Kühn
Berner & Mattner Systemtechnik GmbH
Berlin, Germany
steffen.kuehn@berner-mattner.com

Abstract— **The trend towards constantly growing numbers of product variants and features in industry makes the improvement of analysis and specification techniques a key efficiency enabler. The development of a single generic functional specification applicable to a whole product family can help to save costs and time to market significantly. However, the introduction of a product-line approach into a system manufacturer's electronics development process is a challenging task, prone to human error, with the risk of spreading a single fault across a whole platform of product variants. In this contribution, a combined approach on variant-management and model-based requirements analysis and validation is presented. The approach, process and tool presented are generally applicable to functional requirements analysis and specification, since informal specifications or only an abstract idea of the required function are demanded as an input. It has been experienced in several industrial projects that the presented approach may help to reduce redundancies and inconsistencies and as a consequence it may ease and improve subsequent analysis, design and testing activities. Furthermore, the application of the presented variant-management approach may benefit from model-based specifications, due to their improved analysability and changeability. In this contribution we present our experiences and results using model-based and variant-management concepts for requirements specification to support system testing. Additionally, we present an extension to integrate testing into the variant-management concept. The presented approach and process are supported by the MERAN tool-suite, which has been developed as an add-in to IBM Rational DOORS.**

Variant management, requirements specification, model-based analysis, testing

I. INTRODUCTION

The quality of a functional specification created in the early stages of an industrial development process significantly influences the design, implementation and quality assurance steps. Formalization and modeling techniques can be key quality enablers during the specification process but are difficult to apply to informal, natural language specifications and are often impractical due to the qualifications, efforts and time required.

Furthermore, the introduction of product-line management approaches [6] into system development processes demand a new generation of testing approaches able to deal efficiently with a large variability of processes, documents, designs, code and products.

The presented approach on variant-management of specifications and model-based requirements analysis is supported by the MERAN tool-suite [1], which automates the creation and update of variant-specific requirements books from a generic base. A selection mechanism allows assigning requirements to variants. Furthermore, generic requirements can be defined using parameters replacing text fragments, numerical values or graphics that vary across a product-line. The variant-specific parameter values can be defined and managed using a separated parameter list. A generic requirement is transformed into a variant-specific requirement by replacing its parameters. Based on the variant-assignment the transformed requirements are copied into their corresponding locations.

In Figure 1 the variant management approach for specifications is presented on an example, where an automotive control function has to be developed for the coupé, limousine, convertible, and station wagon variants. The selection assigns requirements from a generic specification to some variants. As an example, the vehicle length may vary across the variants. To avoid writing a requirement regarding the vehicle length for each variant, a generic requirement is specified containing a parameter x instead of concrete values. The generic requirement is transformed into variant-specific requirements automatically using the MERAN variant manager.

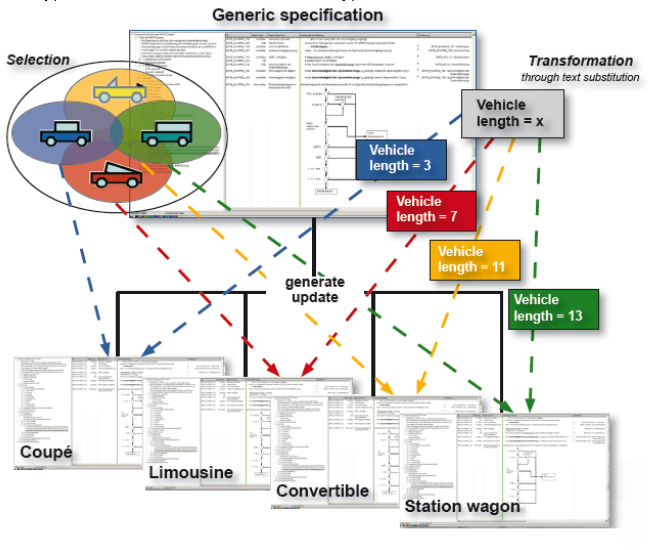


Figure 1 Selection and transformation of generic requirements

A model-based requirements analysis process for product-lines was introduced in [1] and applied successfully during several automotive system development projects. In Figure 2 the model-based MERAN requirements analysis process is presented. In workshops an abstract model of the required system is agreed, which is used to structure the initial requirements book. For each state and each transition in the model a corresponding chapter is created in the requirements book.

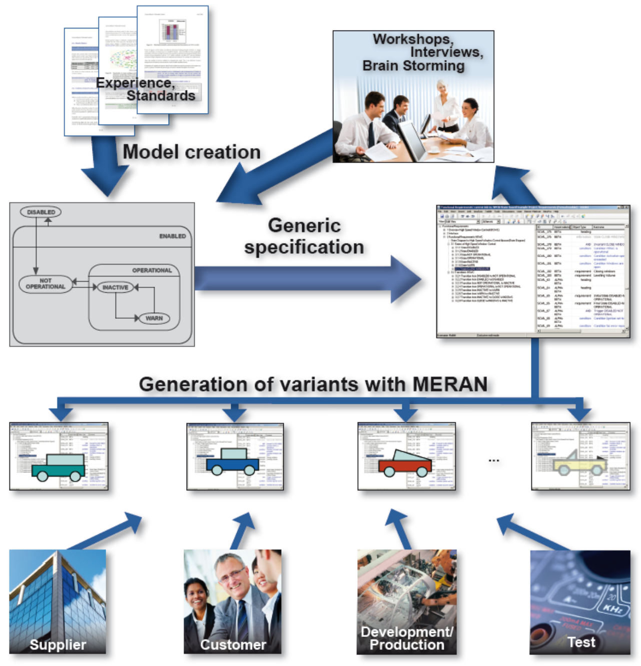


Figure 2 Model-based requirements analysis for product-lines

The use of state and transition chapter templates eases comparison, completeness and consistency checking of requirements. Iteratively, the requirements are refined and extended on the basis of the abstract model; each identified requirement is associated to a state or transition and placed into the corresponding chapter. The process is supported by the MERAN tool-suite allowing automatically creating and updating of variants and supports model construction using various tools, e.g. Microsoft Visio. Furthermore, MERAN allows to manage and link keywords, signal and test-case specifications automatically. The use of different modeling notations and semantics is supported and can be adapted to common modeling processes and tools. In this paper, we will emphasize on the use of state-based specifications on the requirements analysis process.

This paper presents an extension to the MERAN approach on model-based requirements analysis [1] that allows incorporating model-based testing into the specification process of product-lines.

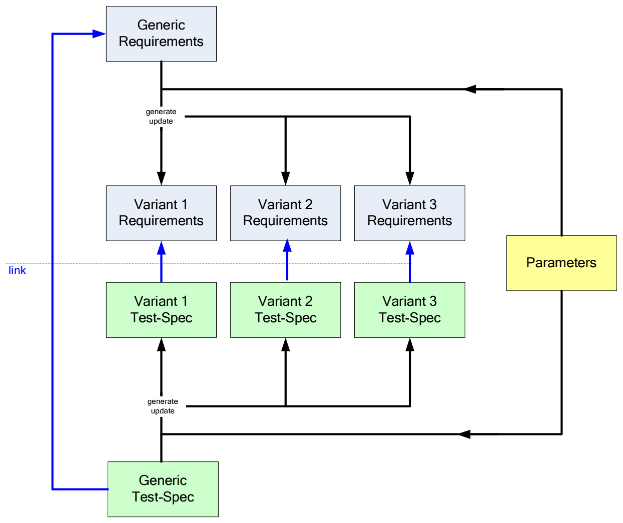


Figure 3 Requirements analyis and testing for product-lines

In Figure 3 the process of generating variant-specific requirements with corresponding variant-specific test-specifications using the MERAN approach is presented. Additionally, a generic test-specification is introduced, from which variant-specific test-specification are generated or updated. As a general process requirement, test-cases need to be traceable to the variant-specific requirements specifications. The model-based requirements analysis provides specifications that support testing through clear and countable requirements, which are structured following a state-based model. The model-based structure and the use of chapter templates enable the application of systematic and model-based testing techniques and allow partial automation of the testing process.

TABLE I ABSTRACT REQUIREMENTS HSWC

High-Level requirements HSWC ALPHA	High-Level requirements HSWC BETA	General requirements
• At high speed greater than 120 km/h open windows shall be detected and an optical information message has to be displayed to the driver • HSWC is operational at speeds greater than 90 km/h. • Failures shall be processed within HSWC and stored in the diagnosis memory	• At high speed greater than 150 km/h open windows shall be detected and closed automatically. Additionally the stereo level has to be reduced to an acceptable level. • In case of a manual deactivation of the rear window actuators or in case of detected rear-seat passengers the rear-seat windows must not be closed • HSWC is operational at speed greater than 100 km/h.	• Failures may be processed by the global failure management and must lead into an isolation of HSWC from any other car-systems • HSWC needs to be activated and deactivated by coding parameters

The approach presented in this paper is being illustrated on the example of a simplified driver-assistance function given as specification an extract. The basics of the modeling notation Statecharts used in the example are explained. Furthermore, the approach on model-based requirements analysis is described. The variant-management for specifications extended by and approach to integrate test-case specification into the model-based specification of product-lines is explained and exercised on the example. Finally, experiences and results are discussed and summarized.

II. EXAMPLE OF A SIMPLE DRIVER-ASSITANCE FUNCTION

The example function High Speed Window Control (HSWC) is developed as an experimental variant ALPHA and for the serial production as a variant BETA.

An extract of the complete functional specification of HSWC is used to illustrate the approach on model-based requirements specification and testing. The complete specification is available as a project archive for IBM Telelogic DOORS at www.berner-mattner.com.

III. 3. STATECHARTS

Statecharts is a graphical notation for hierarchical finite state-machines that is applicable to the specification of state-based system behaviour. In the context of the presented approach a sub-set of Statecharts has been approved useful and will be explained in this chapter. A complete and detailed description is provided in [6].

The nodes of a statechart represent a finite set of system states, while the edges represent transitions between these states. A statechart may process a finite set of inputs and produces a finite set of outputs. A statechart may be extended with data of arbitrary number and types.

A state may contain a finite set of sub-states, of which one has to be the initial state, i.e. when transiting into the state the initial sub-state is entered initially. A transition into a hierarchical state, i.e. a state containing sub-states, ends either in the initial or in one of the sub-states. An atomic state does not contain any sub-states. A state-invariant specifies valid conditions on the values of the data-extension when the system resides in the corresponding state.

A state transition possesses an initial and a final state. A group transition may start in a hierarchical state, respectively in one of its sub-states. The trigger condition of a transition specifies valid conditions on inputs and data extension under which the transition executes. Furthermore a transition allows specifying data operations and outputs created during its execution. The introduced terminology is demonstrated on the following example:

The example of a hierarchical statechart of the function HSWC in Figure 4 contains 5 atomic and three hierarchical states. The system resides initially in state DISABLED. From the initial state the system can transit into state NOT OPERATIONAL. From each sub-state of state ENABLED HSWC may transit into states DISABLED and ERROR. From state NOT OPERATIONAL the system may transit into states INACTIVE and ERROR. Within the hierarchical state OPERATIONAL the function HSWC may transit between the sub-states INACTIVE and ACTIVE. Initial states are marked as bold nodes. The state ERROR represents the internal error processing and is reachable from any sub-state of ENABLED. The state HSWC represents the context of the whole function with all its sub-states.

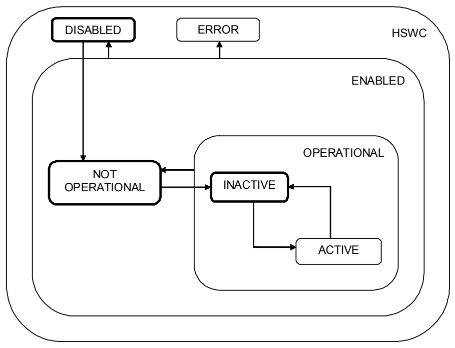
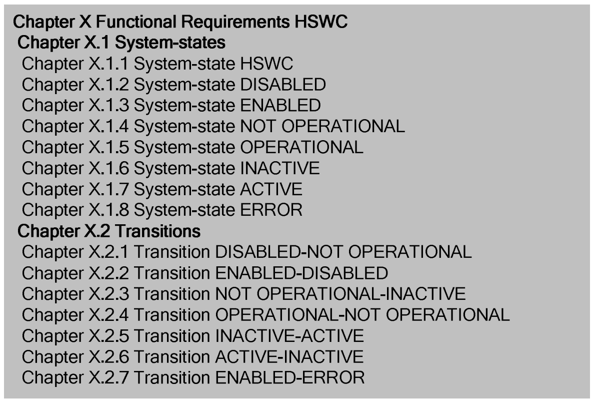


Figure 4 Bild 3: Statechart der Funktion HSWC

IV. 4. MODEL-BASED REQUIREMENTS ANALYSIS

The model-based analysis makes use of statecharts describing the functional behaviour of the required system in order to give structure to requirements during the analysis process. A statechart of the required functionality is created initially to a system development process and committed in workshops and interviews. A specification structure is created corresponding to the committed model. During iterative analysis and elicitation steps this structure is used to arrange refined and additional requirements in the context of the required function, system states or transitions.



Chapter X Functional Requirements HSWC
Chapter X.1 System-states
Chapter X.1.1 System-state HSWC
Chapter X.1.2 System-state DISABLED
Chapter X.1.3 System-state ENABLED
Chapter X.1.4 System-state NOT OPERATIONAL
Chapter X.1.5 System-state OPERATIONAL
Chapter X.1.6 System-state INACTIVE
Chapter X.1.7 System-state ACTIVE
Chapter X.1.8 System-state ERROR
Chapter X.2 Transitions
Chapter X.2.1 Transition DISABLED-NOT OPERATIONAL
Chapter X.2.2 Transition ENABLED-DISABLED
Chapter X.2.3 Transition NOT OPERATIONAL-INACTIVE
Chapter X.2.4 Transition OPERATIONAL-NOT OPERATIONAL
Chapter X.2.5 Transition INACTIVE-ACTIVE
Chapter X.2.6 Transition ACTIVE-INACTIVE
Chapter X.2.7 Transition ENABLED-ERROR

Figure 5 Structure of the HSWC specification derived from statechart

Example. In Figure 5 the structure of the HSWC specification that is derived from the statechart in Figure 3 is presented. The following process-step allocates the known requirements and such being identified during the analysis process into the structure. Furthermore, existing requirements need to be atomized and typed in a separate process step, i.e. a specification object must describe a single issue and is assigned into the types presented TABLE II.

Through atomization and typing of requirements the changeability and of the specification can be significantly improved.

Example. In TABLE III typed and atomized specification objects are presented. The information does not require any specific system properties but provides an explanation on the requirements.

Frequently, logical conditions can be found in specifications of technical control functions that are used in a similar fashion in statechart models. In order to support systematic or model-based analysis techniques we introduce condition types in TABLE IV, which allow specifying trigger conditions and state-invariants in a systematic manner.

Example. In TABLE V the example of the specification of the activation conditions of HSWC using condition types is presented.

A state is specified by optional sub-states, an optional initial state, a state invariant and optional additional requirements. In order to ease requirements changes and analyses, a state-specification template is introduced that predefines the structure of state-specification chapters.

TABLE II REQUIREMENT TYPE DECLARATIONS

Type	description
heading	a structure element
information	an explaining or commenting issue
requirement	the specification of a single required product property

TABLE III TYPED AND ATOMIZED REQUIREMENTS ON HSWC

Type	Text
information	Rear-seat passengers may be put at risk by automatically closing windows.
requirement	Manually deactivated window-opener must not be closed automatically by HSWC.
requirement	In case of rear-seat passengers detected windows must not be closed automatically by HSWC.

TABLE IV CONDITION TYPE DECLARATIONS

type	description
condition	an atomic condition
XOR	exclusive disjunctive composition of conditions
OR	disjunctive composition of conditions
AND	conjunctive composition of conditions

TABLE V ACTIVATION-CONDITIONS OF HSWC

Type	Text
AND	HSWC is activated, if every of the following conditions holds:
condition	• HSWC is operational
condition	• The speed of the system vehicle is greater than V_{aktiv}.
condition	• No rear-seat passengers are detected.
condition	• Rear-seat window actuators are operational.

TABLE VI SPECIFICATION OF STATE OPERATIONAL

Type	ALPHA	BETA	Name	Text
heading	X	X		X.1.5 State OPERATIONAL
information	X	X	State OPERATIONAL	The state OPERATIONAL represents the operational function HSWC.
requirement	X	X	Sub-states OPERATIONAL	State OPERATIONAL includes State INACTIVE and State ACTIVE.
requirement	X	X	Initial State OPERATIONAL	The initial sub-state in State OPERATIONAL is State INACTIVE.
AND	X	X	Condition Operation HSWC	The function HSWC is operational, if every of the following conditions hold:
condition	X	X	Condition HSWC is enabled	• The function HSWC is enabled; in State ENABLED.
condition	X	X	Condition Operation Speed exceeded	• The vehicle speed is greater or equals $V_{Operation}$.
condition		X	Condition No rear-seat passengers	• On the rear-seats no passengers are detected.

TABLE VII GENERIC REQUIREMENTS SPECIFICATION OF TRANSITION INACTIVE-ACTIVE

Type	ALPHA	BETA	Name	Text
heading	X	X		X.1.4 Transition from INACTIVE to ACTIVE
information	X	X	Transition INACTIVE-ACTIVE	The transition from state INACTIVE to state ACTIVE represents the activation of HSWC.
requirement	X	X	Initial State	The initial state is State INACTIVE.
requirement	X	X	Final State	The final state is State ACTIVE.
AND	X	X	Condition Activation HSWC	The function HSWC is activated if every of the following conditions hold:
condition	X	X	Condition HSWC is operational	• The function HSWC is operation; the function is in State OPERATIONAL.
condition	X	X	Condition Windows are open	• At least one window is open.
condition	X	X	Condition Vehicle exceeding v_{act}	• The vehicle speed is greater or equals $v_{Activation}$.
requirement		X	Command closing windows.	On activation of HSWC windows must be commanded to close.
requirement	X		Activation Message	On activation of HSWC a message must be displayed to the driver.
requirement	X	X	Activity Report	On activation of HSWC the activity bit has to be set.

Example. In TABLE VI the specification of state OPERATIONAL based on the state-specification template is presented. The state-invariant of state OPERATIONAL is specified using the proposed condition types and extends the state-invariant of state ENABLED with conditions on activation speed, window actuator status and rear-passenger detection.

A transition is specified by initial state, final state, trigger condition, and optional requirements. In TABLE VII the specification of the transition from state INACTIVE to state ACTIVE presented. In order to ease requirements changes and analyses, a transition-specification template is introduced that predefines the structure of transition-specification chapters.

V. VARIANT-MANAGEMENT FOR SPECIFICATIONS

The variant-management approach for specifications includes selection mechanisms to assign specification objects to variants and a parameter replacement mechanism to transform generic into variant-specific specification objects. A variant, e.g. Limousine or Convertible, is an instance of a variant class, e.g. vehicle body design. For many applications of the presented approach a single class of variants should be sufficient. However, there may be cases where the use of multiple variant classes is beneficial, e.g. a stereo system is available as standard and high-end model and may be ordered in a Limousine and Coupé vehicle body design. Consequently, the variant management approach allows combining the two variant classes Body Design and Stereo Type. In the presented example of a car system a single variant class is applied to the requirements specification. The testing approach for the same example function makes use of two variant classes.

In the example in TABLE VI and TABLE VII binary attributes are used to assign specification objects to variants. In TABLE VI the condition 'No rear passengers' is assigned to variant BETA only. In TABLE VII the requirement 'Activation Message' is assigned to variant ALPHA only.

The use of parameters to create generic specifications allows using similar specification texts over many variants and to define and manage variant-specific contents in a separated list. An automatic mechanism allows generating and updating variant-specific texts and specification collections when required. The use of generic specifications reduces the effort of creating and changing variants due to a minimization of redundancy. Furthermore, the risk of specification faults and inconsistencies is reduced as a consequence of avoiding repetitive tasks.

Example. In TABLE VII and TABLE VI the conditions on exceeding operational resp. activation speeds are specified by using the parameter $v_{activation}$ and $v_{operational}$.

TABLE VIII PARAMETER LIST

Parameter Name	Parameter Value			Type
	default	ALPHA	BETA	
$v_{Activation}$	150	120		km/h
$V_{Operation}$	100	90		km/h

TABLE VIII presents the parameter list for the examples in TABLE VI and TABLE VII. The definition of parameters is allowed for pictures, OLE-objects, text fragments and numerical data.

VI. TEST-CASE SPECIFICATION FOR PRODUCT-LINES

The model-based specification supports testing through its clear, lean structure and countable requirements. Since the structure is based on a state-based model, coverage criteria can be applied to states, transitions, and paths. Furthermore, specification of conditions is systematic and allows the application of condition coverage criteria.

TABLE IX TESTING TYPE DECLARATIONS

type	description
test-case	containing pre-/post-conditions, and test-steps
pre-condition	specifies initial system state
test-step	specifies separated input and response pairs
post-condition	specifies final system state

The variant-management approach for specifications has been presented to be making use of a single class of variants, the product type. Furthermore, the variant-management can be extended to multiple variant classes, e.g. purpose classes, function classes, or process steps applicable. As an additional variant class for testing, the extension by a purpose has been seen beneficial. Typical testing purposes may be positive and negative testing, where positive test-cases are expected to stimulate the expected response, while in response to negative test-cases the system must not produce the specified output. Using this additional variant class allows us to specify compact test-cases following a systematic approach, e.g. using equivalence class and boundary partitioning methods [3]. In order to support systematic model-based testing further specification object types are introduced in TABLE IX that allow the structured specification of test-cases, pre-conditions, test-steps, and post-conditions.

TABLE X TEST-CASE SPECIFICATION FOR THE ACTIVATION BOUNDARY OF HSWC

Type	Mark		Purpose		Input	Expected Response
	ALPHA	BETA	positive	negative		
test-case	**X**	**X**	**X**	**X**		
pre-condition	X	X	X	X	HSWC is in state INACTIVE	
pre-condition	X	X	X	X	HSWC is operational	
test-step	X	X	X		Open rear-windows.	Windows are opening.
test-step	X	X		X	Close any windows.	Windows are closed.
test-step	X	X	X		Accelerate car to a speed of $v_{activation}$	Car speed is equal or greater $v_{activation}$.
test-step	X	X		X	Accelerate car to a speed of ($v_{activation}$-1)	Car speed is less than $v_{activation}$.
post-condition		X	X	X		Signal 'Close Windows' emitted.
post-condition	X		X	X		Signal 'Close Windows' not emitted.
post-condition	X		X	X		Warning message is displayed.

TABLE XI POSITIVE TEST-CASE SPECIFICATION OF HSWC ACTIVATION

Type	Mark		Input	Expected Response
	ALPHA	BETA		
test-case	**X**	**X**		
pre-condition	X	X	HSWC is in state INACTIVE	
pre-condition	X	X	HSWC is operational	
test-step	X	X	Open rear-windows.	Windows are opening.
test-step	X	X	Accelerate car to a speed of $v_{activation}$	Car speed is equal or greater $v_{activation}$.
post-condition		X		Signal 'Close Windows' emitted.
post-condition	X			Signal 'Close Windows' not emitted.
post-condition	X			Warning message is displayed.

TABLE XII NEGATIVE TEST-CASE SPECIFICATION 1 OF HSWC ACTIVATION

Type	Mark		Input	Expected Response
	ALPHA	BETA		
test-case	**X**	**X**		
pre-condition	X	X	HSWC is in state INACTIVE	
pre-condition	X	X	HSWC is operational	
test-step	X	X	Close any windows.	Windows are closed.
test-step	X	X	Accelerate car to a speed of $v_{activation}$	Car speed is equal or greater $v_{activation}$.
post-condition		X		Signal 'Close Windows' emitted.
post-condition	X			Signal 'Close Windows' not emitted.
post-condition	X			Warning message is displayed.

TABLE XIII NEGATIVE TEST-CASE SPECIFICATION 2 OF HSWC ACTIVATION

Type	Mark		Input	Expected Response
	ALPHA	BETA		
test-case	**X**	**X**		
pre-condition	X	X	HSWC is in state INACTIVE	
pre-condition	X	X	HSWC is operational	
test-step	X	X	Open rear-windows.	Windows are opening.
test-step	X	X	Accelerate car to a speed of ($v_{activation}$-1)	Car speed is less than $v_{activation}$.
post-condition		X		Signal 'Close Windows' emitted.
post-condition	X			Signal 'Close Windows' not emitted.
post-condition	X			Warning message is displayed.

TABLE XIV POSITIVE TEST-CASE HSWC ALPHA ACTIVATION

Type	Input	Expected Response
pre-condition	HSWC is in state INACTIVE	
pre-condition	HSWC is operational	
test-step	Open rear-windows.	Windows are opening.
test-step	Accelerate car to a speed of 120 km/h	Car speed is equal or greater 120 km/h.
post-condition		Signal 'Close Windows' not emitted.
post-condition		Warning message is displayed.

TABLE XV NEGATIVE TEST-CASE 1 HSWC ALPHA ACTIVATION

Type	Input	Expected Response
pre-condition	HSWC is in state INACTIVE	
pre-condition	HSWC is operational	
test-step	Close any windows.	Windows are closed.
test-step	Accelerate car to a speed of 120 km/h	Car speed is equal or greater 120 km/h.
post-condition		Signal 'Close Windows' not emitted.
post-condition		Warning message is displayed.

TABLE XVI NEGATIVE TEST-CASE 2 HSCW ALPHA ACTIVATION

Type	Input	Expected Response
pre-condition	HSWC is in state INACTIVE	
pre-condition	HSWC is operational	
test-step	Open rear-windows.	Windows are opening.
test-step	Accelerate car to a speed of (120 km/h-1)	Car speed is less than 120 km/h.
post-condition		Signal 'Close Windows' not emitted.
post-condition		Warning message is displayed.

In TABLE X the test-case specification corresponding to the requirements specification of the activation of HSWC in TABLE VII is presented. The generic test-case specification makes use of the two variant classes Mark and Purpose. The example implements test-cases corresponding to boundary analysis and equivalence partitioning. Therefore, for each condition a minimal number of positive test-cases covering every input class that enables the expected behaviour is created and for each input class disabling the expected behaviour a negative test-case is created that isolates the disabling input. Values from an input class are preferably chosen on the class boundaries.

The MERAN variant-management tool uses the generic test-case specification in TABLE X as an input and generates in two steps the variant-specific test-case specifications. In the first generation step intermediate specifications are generated from variant class Purpose. The resulting test-case specifications are presented in TABLE XI, TABLE XII, and TABLE XIII. In the second generation step the final variant-specific test-case specifications are generated based on variant class Mark. The resulting test-case specifications are presented in the tables below:

TABLE XVII POSITIVE TEST-CASE HSWC BETA ACTIVATION

Type	Input	Expected Response
pre-condition	HSWC is in state INACTIVE	
pre-condition	HSWC is operational	
test-step	Open rear-windows.	Windows are opening.
test-step	Accelerate car to a speed of 150 km/h.	Car speed is equal or greater 150 km/h.
post-condition		Signal 'Close Windows' emitted.

TABLE XVIII NEGATIVE TEST-CASE 1 HSCW BETA ACTIVATION

Type	Input	Expected Response
pre-condition	HSWC is in state INACTIVE	
pre-condition	HSWC is operational	
test-step	Close any windows.	Windows are closed.
test-step	Accelerate car to a speed of $v_{activation}$	Car speed is equal or greater $v_{activation}$.
post-condition		Signal 'Close Windows' emitted.

TABLE XIX NEGATIVE TEST-CASE 2 HSCW BETA ACTIVATION

Type	Input	Expected Response
pre-condition	HSWC is in state INACTIVE	
pre-condition	HSWC is operational	
test-step	Open rear-windows.	Windows are opening.
test-step	Accelerate car to a speed of (150 km/h-1)	Car speed is less than (150 km/h).
post-condition		Signal 'Close Windows' emitted.

VII. RELATED WORK AND EXPERIENCES

The presented approach on variant-management for specifications has been applied to several industrial projects at German car-manufacturers. As a consequence, the effort of creating, updating and managing functional specifications of product-lines [6] could be reduced significantly.

The combination of model-based analysis and variant-management potentially allows further efficiency improvements; specifically the improved document structures lead to a significant reduction of effort and time of changes in the generic specification, which can be automatically transferred to variant-specifications. The theoretical cost saving of (1-1/n)•100 % for a specification involving n variants could never be achieved. Practically, specifications contain objects which are assigned to a single or few variants only, thus variant-management is of restricted use. However, we have seen that effort and time creating and updating functional specifications were reduced by more than 50 % compared to specifications of similar functionality not using this approach, though generating reliable empirical data is a topic of ongoing research.

The presented approach has been applied to embedded system development projects in automotive industry and therefore our experiences are restricted to (a) the viewpoint [9] of functionality that is exercisable by the driver, (b) the description of functionality using state-based models, and (c) using reviews and inspection as the main requirements analysis and elicitation techniques. The application of the presented approach allowing other viewpoints, modelling notations and semantics, see [8] as an example, and improved analysis and elicitation techniques are a topic of ongoing research. Furthermore, the integration of non-functional and hardware requirements seems to be promising but needs further research and validation.

The resulting specifications provide connection points to sub-sequent analysis, design and testing activities. The formalization of requirements, such as resulting from the presented approach, is topic of ongoing research. An extension to sequence enumeration [4] of improved efficiency is topic of ongoing research and contributes to the problem of formalizing requirements by applying a formal structure to requirements sets. Such formalization techniques take natural language specifications as an input and generate formal specifications that are traceable to the input. Furthermore, the resulting specifications can be used as an input to further formal analysis, testing and design activities, e.g. model checking [10], model-generation [2], or model-based testing [5].

VIII. SUMMARY

In this paper, we presented a variant-management approach for specifications that supports model-based requirements analysis and systematic testing. We have demonstrated the practicability of this approach on the example of a simple driver assistance function and shown that it is possible to reduce effort on requirements writing significantly, when product variants need to be considered.

The presented approach emphasises on the use of natural language specifications. Furthermore, the presented approach is applicable to functional specifications of arbitrary format and quality. It is implemented as an add-in for IBM/Rational DOORS as part of the MERAN tool-suite available at www.berner-mattner.com.

The purpose of the presented approach is to improve efficiency during the requirements analysis and validation in the early and final stages of a system development process. Furthermore, the presented approach contributes to requirements tracing problems and provides a connection point for model-based development, analysis and testing activities.

The tool and the method are currently being applied in several projects on automotive system development and testing at German car manufacturers. Further experiences and empirical results will be reported in subsequent publications.

[1] C. Robinson-Mallett, J. Köhnlein, M. Grochtmann, J. Wegener, Model-based specification of product lines, Elektronik im Kfz, Baden-Baden, VDI, October 2009, English abstract, further information on www.berner-mattner.com/en/berner-mattner-home/products/meran

[2] C. Robinson-Mallett, R: M. Hierons, Jesse H. Poore, Peter Liggesmeyer, Using communication coverage criteria and partial model generation to assist software integration testing. Software Quality Journal 16(2): 185-211 (2008)

[3] B. Beizer, Black-box testing: techniques for functional testing of software and systems, John Wiley & Sons (1995)

[4] S. Prowell, J. H. Poore, Foundations of Sequence-Based Software Specification, 2003 IEEE Transaction of Software Engineering, Vol. 29, No. 5

[5] G. Walton, J. Poore, C. Trammell, "Statistical Testing of Software Based on a Usage Model", Software: Practice and Experience, Vol. 25, No. 1, January 1995, 97--108

[6] P. Clements, L. Northrop, Software Product Lines: Practices and Patterns. Addison-Wesley, 2001

[7] Object Management Group (OMG): Unified Modeling Language (UML), Version 2.2, 2009, verfügbar auf www.omg.org

[8] E. Geisberger, J. Grünbauer, B. Schätz: A Model-Based Approach To Requirements Analysis, 2007, Dagstuhl Seminar Methods for Modelling Software Systems

[9] L. Leite and P. Freeman. Requirements Validation Through Viewpoint Resolution,. IEEE Transaction on Software Engineering, 1991

[10] E. M. Clarke, O. Grumberg, D. A. Peled, Model Checking. MIT Press. Boston. 2000