An Approach on Integrating Models and Textual Specifications

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

Abstract—**The time-to-market pressure and the introduction of new development standards into automotive industry have created interest in systematic approaches across the whole product lifecycle. Model-based design and testing methods and tools have been successfully introduced into development processes of car manufacturers and suppliers. Surprisingly, in practice the elicitation and specification of requirements remained largely unaffected by the introduction of model-based methods, while much effort has been spent on the introduction of functional specifications. As a consequence, where functional specifications exist, these are often created unsystematically and are of poor quality, leading to further problems during design, implementation and testing. In this paper we present an approach to improve specification quality through systemization of specification structures based on architectural block diagrams, behavioural statecharts and propositional logic structures. The approach has been implemented in a tool. Experiences from the application of approach and tool in industrial projects are reported.**

Requirements specification, model-based analysis

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 modelling 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, effort and time required.

A model-based requirements analysis process for product-lines was introduced in [1] and we presented details on creating test-cases from specifications integrating statecharts in [3]. The approaches presented in [1] and [3] are restricted to statecharts. In this paper an extension of these approaches is presented that allows the integration of a variety of modeling notations into textual specification using the same principles as presented in [1] and [3]. An abstraction technique is presented that maps models into labeled multi-graphs allowing the implementation of standard graph traversal algorithms to generate specification book structures from models and to check and preserve consistency of both during the following development and change process. This solution has been implemented using IBM Rational DOORS and Microsoft Visio.

Some of the following explanations may be difficult to follow without basic knowledge of the DOORS-specific nomenclature; a specification table is called a *module*; rows in a specification table are called *objects*; objects possess *attributes*, which correspond to columns in the table.

For many technical applications, e.g. driver assistance systems, block diagrams and statecharts sufficiently cover the architectural and behavioral aspects of the system under development. Furthermore, distributed and concurrent functionality can be described using a combination of both. The presented example is restricted to block diagrams and statecharts, although the approach itself is not.

The approach presented in this paper is illustrated on the example of a simplified belt-warner function given as a specification extract. The basics of the modeling notations block diagrams and statecharts used in the example are explained. Finally, experiences and results are discussed and summarized.

A. Example of a simple belt-warner

The simple belt-warner example is a simplified extract of the specification of a real driver assistance function that warns the driver at critical speeds that the safety belt is not tightened. Furthermore, it is required that the belt-warner does not bother the driver at low speeds.

B. Block Diagrams

In this paper, block diagrams are used to sketch the topology and signal routing of a functional system. Of course there are many other purposes for using block diagrams, e.g. Mathwork's Matlab Simulink makes extensive use of block diagrams to describe calculation schemes. Block diagrams are graphical structures consisting of hierarchical blocks, signal flows, signal entry points and signal exit points. A hierarchical block represents a system component, contains a functional behavior, and may contain further hierarchical blocks. In a diagram a hierarchical block is represented as rectangles. Signal flows, represented as edges in the block diagram, specify message routing between blocks. Signal entry and exit points are specialized blocks, which are used to define the system's border.

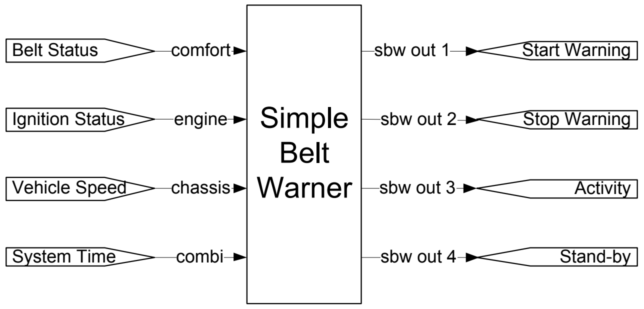

Figure 1 Architecture of Simple Belt-Warner

The example architecture of a simple belt-warner in Figure 1 contains a single block, 4 entry points and 4 exit points. The entry points define the input interface, which consists of the signals Belt Status, Ignition Status, Vehicle Speed, and System Time. The exit points define the output interface, which consists of the signals Start Warning, Stop Warning, Activity, and Stand-by.

C. *Statecharts*

Statecharts is a graphical notation for hierarchical finite state-machines that is applicable to the specification of state-based system behavior. In the context of the presented approach a sub-set of statecharts has proved to be useful and will be explained in this section. A complete and detailed description is provided in [4].

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 changing into the state the initial sub-state is entered. A transition into a hierarchical state, i.e. a state containing sub-states, ends either implicitly in the initial or explicitly in one of the sub-states. An atomic state does not contain any sub-states. A state-invariant specifies valid conditions when the system resides in the corresponding state.

A state transition possesses an initial and a final state. A group transition starts from a hierarchical state S_h, i.e. it can be triggered from any sub-state of S_h. The trigger condition of a transition specifies conditions on inputs and data extension under which the transition executes. Furthermore, in a transition it is possible to specify data operations and outputs created during its execution. In this approach propositional logical is used to systematically specify inputs, outputs and operations in a single, global condition.

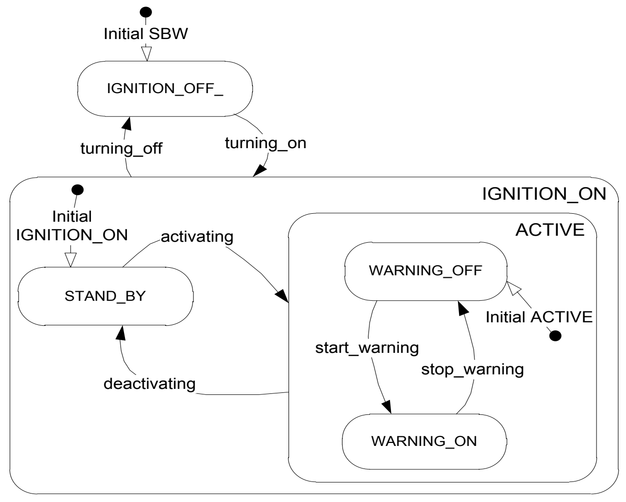

Figure 2 Statechart Simple Belt-Warner

The statechart in Figure 2 contains the states IGNITION_OFF, STAND_BY, WARNING_OFF, WARNING_ON, and the hierarchical states IGNITION_ON and ACTIVE. The control flow is specified by the transitions: Initially the function resides in state IGNITION_OFF. On turning power on a transition into state STAND_BY is triggered. Depending on the activation conditions the function transits into state WARNING_OFF. When warning conditions hold the function transits into state WARNING_ON and starts warning until warning conditions fail, the function is deactivated or the system is turned off. State STAND_BY is the initial state of hierarchical state IGNITION_ON. State WARNING_OFF is the initial state of hierarchical state ACTIVE. From every sub-state of hierarchical state IGNITION_ON the system can be turned off. From every sub-state of state ACTIVE the function can be deactivated.

II. Model-Based Requirements Specification

The integration of models and textual specifications is based on a basic rule, i.e. create a specification chapter for each element in the integrated model. Generally, model elements are related to nodes and edges of the graph structure that is underlying a model. As an example, model elements of a statechart are states, i.e. related to nodes, and transitions, i.e. related to edges. Model elements of a block diagram are blocks, i.e. related to nodes, and signal flows, i.e. related to edges. We handle both modelling notations in a common way and create a specification chapter for each node and each edge. Such a specification chapter consists of a header object containing the chapter title and a sub-object structure, i.e. the chapter skeleton, containing information and requirements specific to the associated model element. For each model element type a chapter template, including a title and skeleton pattern, can be predefined. Such a template allows creating a specification skeleton from a model, including chapter skeletons for each model element.

In many modelling notations, e.g. statecharts or activity diagrams, conditional expressions provide an important part of the modelled information. For example, in statecharts each state defines a state invariant and each edge defines a trigger condition. In the presented approach, propositional logic is used to systematically specify conditional structures.

The model abstraction technique, specification chapter templates and propositional logic structures are presented in this section.

A. *Abstracting Multi-Graphs from Models*

A model abstraction technique is proposed to generalize the handling of modeling notations. Block diagrams and statecharts are abstracted to labeled multi-graphs to allow the application of standard graph algorithms to both modeling notations.

A labeled multi-graph $G = (V,E,L_V,L_E)$ is a structure consisting of a set of nodes V and a multi-set of edges E, where an edge in E is an ordered pair of nodes, a set of node labels L_V, and a set of edge labels L_E.

Node and edge labels contain unique *identifiers*. Furthermore, labels allow the set of nodes and edges to be partitioned. Each model element is tagged with a *type* label that is an element of *T = {entry, exit, node, hierarchical node, edge}*. Future work will aim at the integration of further modeling notations, so that T may need to be extended. Additionally, labels are used to store *references* into the specification structure, to provide traceability in the textual specification.

As an example, the block diagram in Figure 1 can be abstracted to the following multi-graph:

Example.

G = (
{Belt Status, Ignition Status, Vehicle Speed, System Time, Start Warning, Stop Warning, Belt Warner Activity, Belt Warner Stand-By, Simple Belt Warner},
{(Belt Status,Simple Belt Warner),(Ignition Status,Simple Belt Warner),(Vehicle Speed,Simple Belt Warner),(System Time,Simple Belt Warner),(Simple Belt Warner,Start Warning),(Simple Belt Warner,Stop Warning),(Simple Belt Warner,Belt Warner Activity), (Simple Belt Warner,Belt Warner Stand-By)},
{(Belt Status,entry),(Ignition Status, entry),(Vehicle Speed, entry),(System Time,entry),(Start Warning, exit),(Stop Warning,exit)(Belt Warner Activity,exit), (Belt Warner Stand-By,exit), (Simple Belt Warner,node)})

B. Preserving Modeling Notation Semantics

A *modeling notation semantics* maps the model-elements of a specific modeling notation, e.g. block diagrams or statecharts, into multi-graphs.

An example of a modelling notation semantics for block diagrams is

S_{bd} *= ((block,node),(hierarchical block,hierarchical node),(signal flow,edge),(signal entry,entry),(signal exit,exit))*

An example of a modelling notation semantics for statecharts is

S_{sc} *= ((State,node),(Hierarchical State,hierarchical node),(Transition,edge),(-,entry),(-,exit))*

C. Generating Specification Structures

A specification structure consists of hierarchically structured header objects containing chapter titles. The title of a chapter related to an element of an integrated model can be generated from multi-graph labels and the modelling semantics template.

For example, a multi-graph node that is labelled with the identifier 'IGNITION_ON' and the type 'hierarchical node', and where the model semantics S_{bd} maps 'hierarchical node' to 'Hierarchical Block', results in a specification chapter title 'Hierarchical Block IGNITION_ON'. If we were using model semantics S_{sc} instead, the resulting specification chapter would be titled 'Hierarchical State IGNITION_ON'. Obviously, there is not much use in applying a statecharts semantics to a multi-graph that is an abstract of a block diagram, but this example demonstrates that chapter titles can be adapted very easily by changing the model semantics template only.

Chapter X Requirements Simple Belt Warner
Chapter X.1 Interface Simple Belt Warner
Chapter X.1.1 Inputs
Chapter X.1.1.1 Input Belt Status
Chapter X.1.1.2 Input Ignition Status
Chapter X.1.1.3 Input Vehicle Speed
Chapter X.1.1.4 Input System State
Chapter X.1.2 Outputs
Chapter X.1.2.1 Output Start Warning
Chapter X.1.2.2 Output Stop Warning
Chapter X.1.2.3 Output Activity
Chapter X.1.2.4 Output Stand-By
Chapter X.2 Functional Requirements Simple Belt Warner
Chapter X.2.1 System-states
Chapter X.2.1.1 Hierarchical State IGNITION_ON
Chapter X.2.1.2 Hierarchical State ACTIVE
Chapter X.2.1.3 State IGNITION_OFF
Chapter X.2.1.4 State STAND_BY
Chapter X.2.1.5 State NOT_WARNING
Chapter X.2.1.6 State WARNING
Chapter X.2.2 Transitions
Chapter X.2.2.1 Transition Initial_SBW
Chapter X.2.2.2 Transition Initial_IGNITION_ON
Chapter X.2.2.3 Transition Initial_ACTIVE
Chapter X.2.2.4 Transition turning_on
Chapter X.2.2.5 Transition turning_off
Chapter X.2.2.6 Transition activating
Chapter X.2.2.7 Transition deactivating
Chapter X.2.2.8 Transition start_warning
Chapter X.2.2.9 Transition stop_warning

Figure 3 Structure of the Simple Belt Warner specification derived from block diagram and statechart

Example. In Figure 3 the structure of the Simple Belt Warner specification that is generated from the block diagram in Figure 1 and the statechart in Figure 2 is presented.

D. Generating Specification Chapter Skeletons

A chapter related to a model element contains information and requirements objects specific to the model element. For example a state chapter contains an invariant object while a transition chapter contains a trigger object.

A chapter template C_{bd} defines the chapter skeletons of block diagrams.

C_{bd} *= (*
(signal entry,(introduction,signal type)),
(signal exit,(introduction,signal type)),
(hierarchical block, {introduction,sub-blocks,statechart)),
(block,(introduction, statechart)),
(signal flow,(introduction,initial block, final block,signal type))

A chapter template C_{sc} defines the chapter skeletons for statecharts:

C_{sc} *= (*
(hierarchical state,(introduction,sub-states, initial,invariant)),
(state,(introduction,invariant)),
(transition, (introduction,intial, final, trigger))

TABLE I SPECIFICATION OBJECT TYPES

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

E. Conditional Specification Structures

In TABLE I specification object types are presented [3], which allow the separation of the primary requirements from the secondary information and structure objects.

TABLE II 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

As an extension to this concept TABLE II presents conditional specification object types [3], which allow requirements objects to be divided into atomic conditions and logical operators. In combination with tree object structures, i.e. operators may contain conditions and operators, propositional logic expressions can be specified.

TABLE III TYPED AND ATOMIZED REQUIREMENTS OF ACTIVATION HYSTERESIS SIMPLE BELT WARNER

Type	Text
information	The driver must not be bothered or distracted from the traffic by unnecessary warnings of the belt-warner at low speeds.
requirement	The belt-warner shall be activated if vehicle speed exceeds 15 mph.
requirement	The belt-warner shall be deactivated if vehicle speed drops below 12 mph.
requirement	The belt-warner shall be deactivated if a relevant malfunction occurs.

Example. In TABLE III typed and atomized specification objects of the activation hysteresis of the simple belt-warner are presented.

TABLE IV WARNING DEACTIVATION CONDITION OF SIMPLE BELT-WARNER

Type	Text
AND	**The warning sound of simple belt warner is stopped immediately if every of the following conditions hold:**
condition	• An acoustic warning is emitted by the belt-warner.
XOR	Based on the vehicle speed the warning is escalated, if one of the following conditions holds:
condition	• At speeds ≤ 25 mph the warning sound 1 is emitted.
condition	• At speeds > 25 mph the warning sound 2 is emitted.
OR	**A safety belts condition is critical, if at least one of the following conditions holds:**
condition	• The driver seat safety-belt is not fastened.
AND	**The passenger seat-belt is in critical condition, if every of the following conditions hold:**
condition	• The passenger seat is occupied.
condition	• The passenger seat-belt is not fastened.

Example. In TABLE IV the example of the specification of the deactivation condition of the warning sound is presented.

Figure 4 illustrates the resulting integrated specification of the example of the simple belt-warner. The textual specification complements the models with details. Specifically, the statecharts are complemented with textual specifications of the state invariants and transition triggers.

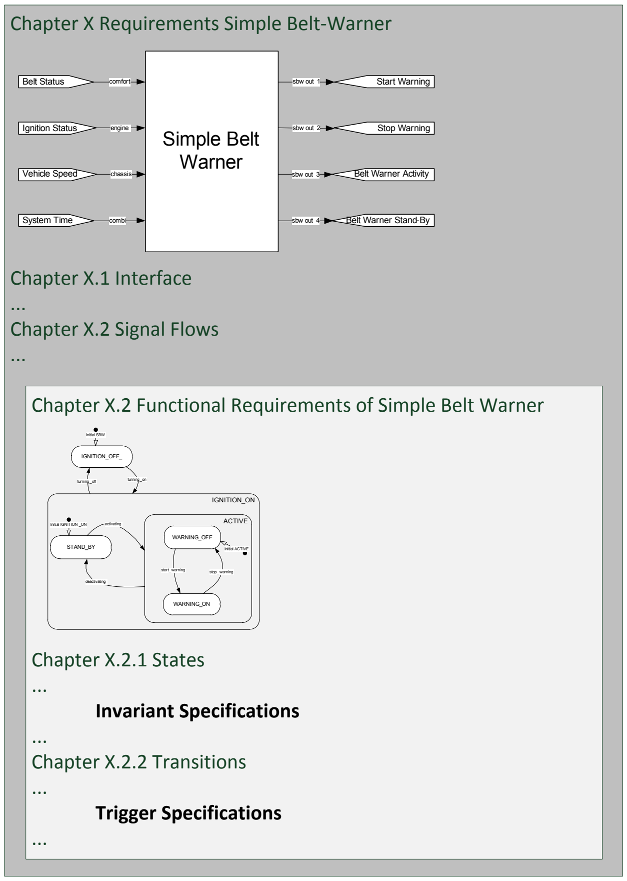

Figure 4 Integrated Model-Based Specification of Simple Belt-Warner

III. RELATED WORK AND EXPERIENCES

The presented approach has been developed and applied for the requirements elicitation and specification of driver assistance systems at a German car-manufacturer. Our experiences are restricted to (a) the viewpoints [5] of functionality and architecture exercisable and observable by the driver, (b) the description of architecture using block diagrams, (c) the description of functionality using state-based models, and (d) 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 and improved analysis and elicitation techniques are a topic of ongoing research. The presented approach allows creating models for each of the five basic modelling views presented in [6]. Although in this paper only solutions to structural and behavioural views have been presented, the adaptation of the approach to scenario, interaction and data views is part of ongoing research. In contrast to [6] this approach does not support a common model for all views, but it is designed to be highly adaptable to various modelling notations applied in practice. The presented approach has been applied to computational independent models (CIM) and Platform Independent Models (PIM), e.g. the OpenUP/MDRE process [7]. In contrast to [7], the presented approach aims at the use of various modelling notations and views, although it may contribute to implementing the OpenUP/MDRE process.

Among the specified driver assistance systems are 4 parking systems, 2 break-assistance systems, and a camera-based traffic information system.

TABLE V STATISTICS OF PROJECTS APPLYING MODEL-BASED SPECIFICATION

System	Comp.	Models	States	Trans.	Effort
Parking_1	4	8	40	69	500
Parking_2	4	5	19	38	650
Brake_1	1	2	6	8	700
Brake_2	1	2	15	18	800

TABLE V presents statistics on four system specifications that were created using the presented approach. The rows refer to the specified systems. The columns contain counts of components, models, states and transitions included in these specifications. The parking systems were specified using the presented tool. Since the system complexity and efforts spent on the specifications were at least equivalent while the parking systems sizes were significantly increasing, we assume that this is the positive effect of the higher degree of automation achievable with the tool applied.

The resulting specifications provide connection points to sub-sequent model-based analysis, design and testing activities, such as sequence enumeration [5], model-based testing [9] or model checking [10].

The presented approach focuses on formalizing the structure of a specification through synchronization with formal models. The formalization of the atomic conditional expressions would enable further application of automation and formal proof techniques. Nevertheless, formal notations only find little acceptance in automotive practice and specifically during requirements analysis prose is needed to allow many people to participate in the elicitation process.

IV. SUMMARY

In this paper, we presented an approach on integrating models and textual specifications. It is implemented as an add-in for IBM/Rational DOORS and Microsoft Visio as part of the MERAN tool-suite available at www.berner-mattner.com.

We have demonstrated usability of the approach on the example of a simplified driver assistance function. The approach and tool have been exercised in more than ten specification projects at German car manufacturers during the past 3 years. We have demonstrated that automation is a key efficiency enabler for this approach.

The presented approach emphasises the use of natural language for the specification of atomic requirements, but provides a formal structure incorporating these atoms to form an arguably complete and consistent whole. Furthermore, the presented approach is applicable to various modelling notations.

Furthermore, the presented approach contributes to requirements tracing problems and provides a connection point for model-based development, analysis and testing activities.

V. REFERENCES

[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] C. Robinson-Mallett, J. Köhnlein, M. Grochtmann, J. Wegener, S. Kühn, Modeling requirements to support testing of product-lines, A-MOST/ICST 2010, Paris, 2010

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

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

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

[7] G. Loniewski, A. Armesto, E. Insfran: An Architecture-Oriented Model-Driven Requirements Engineering Approach, Workshop on Model Driven Requirements Engineering, Trento, 2011

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

[9] 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

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

Acknowledgements

The late Prof. Dr. Jesse Poore had significantly inspired this work and always was an invaluable discussion partner. Also, many thanks to Prof. Dr. Robert M. Hierons for reviewing this paper, to the team of the Berner & Mattner Competence Center for Requirements and Validation, and to our customers for the many suggestions, time and efforts to develop this approach and applying it in their projects. This work has been supported by the German Federal Ministry of Education and Research under grant number SPES XT: 01IS2005Q.