IT Security in Automotive Software Development

Sandro Schulze, Mario Pukall, Tobias Hoppe
School of Computer Science
University of Magdeburg
{sanschul, mario.pukall, tobias.hoppe}@iti.cs.uni-magdeburg.de

Abstract

In the last years, automotive systems evolved to be more and more software-intensive systems. As a result, considerable attention has been paid to establish an efficient software development process of such systems, where reliability is an important criterion. Hence, model-driven development (MDD), software engineering and requirements engineering (amongst others) found their way into the systems engineering domain. However, one important aspect regarding the reliability of such systems, has been largely neglected on a holistic level: the IT security. In this paper, we introduce a potential approach for integrating IT security in the requirements engineering process of automotive software development using function net modeling.

1. Introduction

Current automotive systems (i.e., being part of automobiles) can be considered as a network of embedded systems (e.g., Electronic Control Units *(ECUs)*) which are connected to each other via different bus systems and thus, exhibit a considerable complexity. Moreover, in the recent years automotive systems became more and more software intensive systems. This is caused by the fact that more and more functionality of an off-the-shelf car is implemented by software functions. It is expected, that in 2010 a medium-sized vehicle will contain approx. 1 GB of Software [1].

In order to develop software in an efficient way and to overcome the (still increasing) complexity anyway, techniques and concepts from the field of software engineering (SE) and requirements engineering (RE) have been proposed and applied to the development process [2], [3], [4], [5], [6]. Although the software should contribute, amongst others, to the reliability of the system, one aspect has been neglected so far in the development process: holistic concepts for *IT security*. Successful attacks on an automotive IT system can have negative implications on the safety of its human users or on the reliability of the system itself [7]. In several studies it has been shown that attacks on current automotive IT systems are possible without much effort and special expertise of vehicle components [8], [9], [10]. However, today it is common that the (mostly generated) code is retrofitted at the end of the development process in order to satisfy security concerns. This, in turn, counteracts the effort of reducing complexity by using modern concepts of SE and RE. Furthermore, this process is prone to simply ignore serious vulnerabilities and thus, allows for security leaks remaining in the system.

We argue that it is inevitable to integrate IT security considerations in the early stages of the (automotive) software development process. In the following, we sketch an approach which aims at specifying security requirements in early stages (i.e. requirements analysis) and thus, can be taken into account for design and implementation.

The remainder of this paper is structured as follows. In Section 2 we give an overview of automotive systems and the relevance of IT security within this domain. In Section 3 we point out the problem of today's automotive software development regarding the IT security. The Sections 4 and 5 encompass our approach for specifying security requirements, divided into modeling and formalization. We close our article with a conclusion and an outlook of future work.

2. Background

In this section, we give an overview of the main characteristics of automotive systems. Futhermore, we point out how IT security is interrelated with these systems and thus, why it is important to consider IT security.

2.1. An Overview of Automotive Systems

In general, it is supposed that appr. 98% of the software worldwide is implemented in embedded systems [11]. Even automotive systems, i.e., the skeleton of today's off-the-shelf cars, make no exception to this fact. Such systems are characterized by a frequent interaction of their components, in detail, dozens of ECUs, sensors and actuators which leads to a complex networked IT system. These interactions are mainly reflected by a frequent exchange of data, which leads to a data-centered character of the overall system. In Figure 1, an excerpt of an automotive system is depicted. The different components can be grouped into different subbus systems, which are in our example the *Infotainment*, *Comfort* and *Power Train* subbus system. However, for decreasing hardware costs (e.g., by saving wires) and because a notable amount of components is not limited to a certain subbus

system (regarding their functionality), such a physical representation is not appropriate anymore. Rather a logical view (considering the functionality within the overall system) should be used to reflect the scope of particular components.

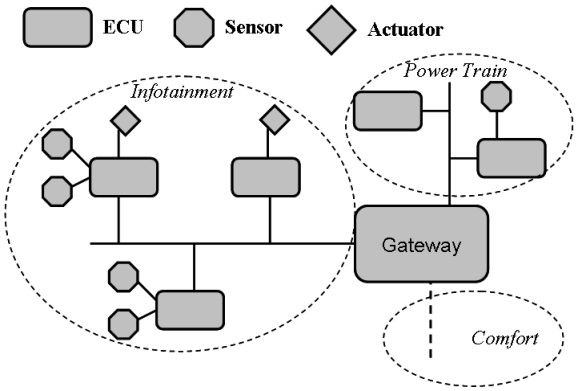

Figure 1. Exemplary Part of an Automotive System

Because each component is a self-contained embedded system, the overall system exhibits a highly heterogeneous character, the more, as different ECUs are provided by different manufacturers. Since this structure already establishes a high complexity of automotive systems, it also makes it a complex and difficult task to develop software for such systems. This is a serious problem, expecially regarding the demands for reliability and safety in such systems. Another challenging task for the software development results from the very restricted conditions and resources in automotive systems. A micro controller commonly used in such environments is characterized by memory of 50-100 KB (distributed over RAM and EEPROM) and computing power of 10-20 MHz. Furthermore, such systems exhibit real time requirements in the dimension of less than 10 ms. Thus, software development is a crucial task for automotive systems (e.g., regarding complexity or reliability) and subject to current software engineering research.

2.2. The Importance of IT Security

Along with the mentioned fundamental changes within automotive systems, the IT security becomes an important issue. Per definition, IT security means reliability in terms of preserving security aspects of information [8], namely *integrity, availability, authenticity, non-repudiability, confidentiality* and *privacy*. Because of its networked character, automotive systems exhibit vulnerabilities to malicious attacks, which, in turn, can violate one ore more aspects of IT security. The access for the execution of an attack on the system can take place in multiple forms from *outside* or *inside* the car, as shown in exemplary case studies reported in [12], [13], [14]. Regardless how access is achieved by the user, the basic attack principle is always the same. It aims at influencing a certain behaviour or state of the automotive system. Since this is done by manipulating the respective functionality (e.g., by introducing malicious code, communication or manipulating data), it directly addresses the software responsible for this functionality. Subsequently, it is reasonable to ensure the IT security of the respective software in order to increase the security of the overall system.

3. Problem Statement

As already stated, automotive systems more and more rely on software to fulfill certain functionalities. Furthermore, the complexity of such systems steadily increases, while the reliability has to be ensured. Alltogether, this is a challenging task to be managed during the software development process of such systems. Hence, different approaches of software development found their way into the systems engineering domain in order to overcome these challenges. For instance, *software product lines (SPL)* as a special concept of software engineering (SE) are used to manage commonalities and variabilities of automotive software [2], [6]. Another common practice is model-based development (MBD) of software for automotive systems, where functionality is described[1] by models [15], [16]. Afterwards, the code is generated automatically based on these models. Finally, *requirements engineering (RE)* gains more and more importance within automotive systems since a good requirements analysis is inevitable for all other stages of the development process.

However, since all mentioned techniques and concepts address the (decrease of) complexity of the system, they often omit one issue, which is important for the reliability: the *IT Security*. Although the IT security could be integrated into these concepts, it is either considered sparingly or even ignored at all. Moreover, the final code is retrofitted with regard to known vulnerabilities. This, in turn, not only endangers the reliability of the system but also counteracts the effort which is invested for decreasing the complexity.

4. Specifying Security Requirements - A Model-Based Approach

The approach we propose for specifying security requirements is twofold. The first part consists of a model which describes the underlying (automotive) system. In this regard we pay special attention to the particular functionalities, realized by the modeled system (e.g., using software) and the data utilized and exchanged for this purposes. The second part is a formalism which describes how security requirements can be propagated semi-automatically throughout the

1. This includes the specification as well as design or implementation of the functionality.

system, represented by the model. While the first part is considered in the following in more detail, the second part is subject to Section 5.

4.1. Functional Dependencies in Automotive Systems

As already mentioned, the physical dependencies between ECUs, sensors, and actuators (in the following referred to as nodes) which is embodied by subbus systems, becomes more and more blurred. Moreover, it is reasonable to consider the dependencies resulting from the functionality executed in automotive systems. The overall functionality can be divided in partial functionalities. For each partial functionality a set of nodes is responsible for its correct execution. To this end, the respective nodes exchange information, e.g., in form of data. This, in turn, evokes dependency relations between nodes of a partial functionality, since a certain node relies on the information of other nodes.

Considering these functional dependencies is advantageous for several reasons. First, it provides a *logical* view on the overall system, which abstracts from the detailed underlying architecture. Consequently, the complexity can be decreased for the dependency considerations. Second, it allows for a detailed investigation of dependencies, because we can focus on a very small set of dependencies, occurring in one partial functionality. Reversely, we can merge several partial functionalities in a bottom-up fashion to achieve a more coarse-grained overview of the logical structure and its existing dependencies. Thus, this approach is scalable which supports even the consideration of large-scaled logical structures (like automotive systems) and its dependencies in particular.

4.2. Function Net Modeling

With the logical architecture in mind, a model is needed reflecting this view, its dependencies and potential vulnerabilities of certain nodes. Since we consider partial functionalities and the corresponding interactions of the responsible nodes, it is reasonable to use *function nets* as the modeling approach. In the recent past, this has already been done in order to achieve an overview (of functions) on a more abstract level or to reduce modeling complexity [17], [18]. For modeling such function nets, the *SysML* [19] can be used, which comes along with two main advantages. First, it has been developed and standardized explicitly for the usage in system engineering, which is the context of our target domain as well. Second, SysML supports modeling functional dependencies by special components, e.g., *block definition diagrams (bdd)* or *internal block diagrams (ibd)*. With the help of these components, it is possible to model the logical architecture with different degrees of granularity.

A function net based on SysMl could be structured as follows. The used components, called *blocks*, encompass corresponding devices in a black box manner. For instance, the *Input Control System* block encompasses devices (e.g., ECUs) used for the input of data to the navigation system. Different blocks can be connected by *ports*, which indicates, that they exchange data (and the direction of the data flow). Furthermore, a port specification can be used to indicate which data is exchanged. Despite this enclosed block notion, functional dependencies may exist beyond the boundaries of a certain block. However, since this model is useful for a first, logical overview of the considered system, it is not helpful to investigate functional dependencies in detail. Therefore, we use the already mentioned *internal block diagrams*. With the help of such a diagram, we can achieve an insight view of one or more blocks. As a result, we can achieve a complete description of the logical architecture of the system.

However, since we are only interested in the data flow (which establishes the dependencies by our means), we even abstract from the introduced SysML model for the rest of this paper. Therefore, we introduce our own model representation, which is exemplary depicted in Figure 2 for the navigation system. Note, that this abstract representation is at least equivalent to the SysML model from a logical point of view. The model consists of several nodes representing existing automotive hardware, namely ECUs, sensors and actuators (which would be visible in the *ibd* of the SysML model). Furthermore, in our example a special node exists, representing the user which is subject to the authentification. It is worth to note, that all nodes could have an unique representation as well. However, we differentiate between these nodes to provide additional semantic information to the respective stakeholder.

Additionally, directed and named edges are used to connect certain nodes (which is done by *ports* in the SysML model). These edges describe the information flow (or data flow respectively) between nodes connected by them, where the names of the edges represent the exchanged data. By the usage of nodes and edges we can even divide a partial functionality in several graphs. In the context of the model, every graph represents a certain feature within the function net. For instance, the *navigation input* feature is composed of the nodes *operating unit*, *input control*, *navigation unit* and the respective edges connecting them.

This graph-based view decreases the complexity of the model further, since it supports the user in extracting nodes, exhibiting dependencies, from the function net. Another character of this view is that it allows for observing the inheritance of data by illustrating the information flow and thus, the inheritance of potential vulnerabilities can be tracked as well. Based on this view, we can also define two general types of node: a *Provider* and a *Consumer* node. The former provides information (in form of data) to one or more

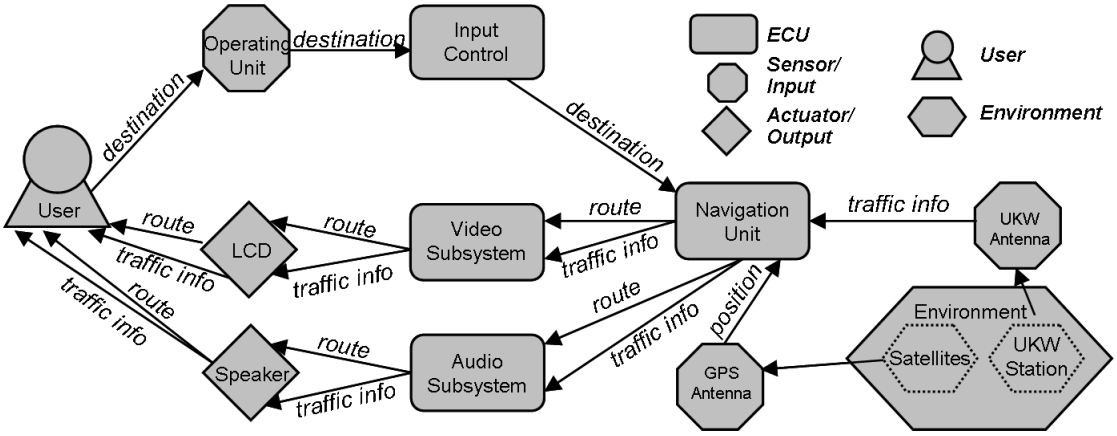

Figure 2. Abstract Model Representation of the exemplary Funcion Net for Navigation System

nodes while the latter consumes information. In particular cases, a node can be both, a provider *and* a consumer, e.g., the *navigation unit* node in Figure 2. These node types are helpful for our requirements analysis approach, because it allows for preliminary predictions if certain security aspects are required. An overview of the relation between node type and typical security aspects is given in Figure 3.

Security Aspect	Required by	Measure by
Confidentiality (C)	Provider	Both
Integrity (I)	Consumer	Both
Availability (A)	Consumer	Provider
Authenticity (U)	Consumer	Both
Non-Repudiability (N)	Both	Both
Privacy (P)	Provider	Provider

Figure 3. Relevance of Security Aspects regarding the Node Type

In the following section we point out how this modeling approach can be used for security requirements specification on a holistic level, i.e., for the whole automotive system.

5. Formalization of Security Requirements

The modeling approach, introduced in the previous section, is useful for identifying security requirements in a fine-granular manner. Nevertheless, it is still insufficient since the model has to be investigated manually for identifying these requirements with regard to the dependencies of the underlying system. This is a cumbersome and tedious work which furthermore can lead to incomplete requirements. To overcome this shortcoming, we propose a formalization which aims at a semi-automatic propagation of security requirements within a function net model, based on certain information like required security aspects or data flow. The results of this formalization are propositions, whether a certain security aspect has to be fulfilled by a certain node of the system (with respect to a certain data item).

Firstly, we make some definitions, needed for the formalization. Amongst others, these definitions represent different components of the abstract model representation (cf. Figure 2). The definitions are as follows:

- $V = \{v_1, \ldots, v_n v\}$ is the set of all nodes within the automotive system (AS), e.g., ECU.
- $D = \{d_1, \ldots, d_n d\}$ is the set of data, contained and exchanged within the AS.
- All edges are defined as $E = \{e_1, \ldots, e_n e\}$. A particular edge is a triple $e_i = \{v_j, v_k, d_i\}$, where $(v_j, v_k) \in E$ are the nodes connected by this edge and d_i is the data exchanged between these nodes (= edge name).
- All graphs are defined as set $G = \{G_1, \ldots, G_n g\}$ with $G_m = \{V_m \subset V, E_m \subset E, s_m, t_m, d_m \in D\}$ as a particular graph. The functions $s_m, t_m : E \rightarrow V$ assign a source node (*source*) and a target node (*target*) respectively to a particular edge [20].
- All function nets are encompassed as $FN = \{f_1, \ldots, f_n f\}$ with a particular function net $f_m i = \{G_i \subset G\}$.
- Finally, the security aspects are represented by the set $S = \{I, U, A, C, N, P\}$ (cf. Figure 3). Additionally, we define two subsets, representing the security aspects for the two node types *Provider* and *Consumer* (cf. Figure 3). To this end, $CR \subset S = S/\{C, P\}$ is the set of typical security aspects possibly required by the node type *Consumer* and $PR \subset S = S/\{I, A, U\}$ the respective set for the node type *Provider*.

With the help of these definitions, we can now formalize how security requirements are propagated within and across function nets. In order to do this, we choose a bottom-up approach, i.e., we initially start our formalization with single graphs (which correspond to a feature in a partial functionality) and apply multilevel composition until we achieve a formalization for the whole system.

Security Requirements for a Single Graph

As a precondition for our formalization, we assume, that all function nets for an automotive system exist (in the best case, there is one huge function net). Subsequently, these function nets are divided into graphs, where a single graph represents a feature of a partial functionality (e.g., considering the *navigation input* feature of our *navigation system* function net). Such a single graph is identified by the data item which is exchanged between the nodes of this graph. As a result, a single graph can be defined as follows: $G_m = (V_m \subset V, E_m \subset E, s_m, t_m, d_m \in D)$.

As a first step, we now have to determine the initial values for an arbitrary node v_i in this graph (with respect to data item d). As a constraint, this node has to communicate the data item to at least one other node v_x in the graph, i.e., $\{(v_i, v_x) \mapsto v_x\} \in t_m$. The initialization is described by Equation (1). It determines, if a security aspect s_i has to be considered for a data item d_j at a certain node.

$$nec_{v_i,d_j}(s_i) = \begin{cases} 1 & v_i, d_j \overset{requires}{\rightarrow} s_i \in A \\ 0 & else \end{cases} \quad (1)$$

Based on this initial node, the security requirements of all other nodes of the respective graph should be derived automatically as far as possible. As an intermediate step, we determine in Equation (2) if a particular security aspect is relevant for a certain node in the graph. Therefore, we take the node types *Consumer (C)* and *Provider (P)* into account.

$$relNT(v_x, s_i) = \begin{cases} 1 & (Type(v_x) =' C' \wedge s_i \in CR) \vee \\ & (Type(v_x) =' P' \wedge s_i \in PR) \\ 0 & sonst \end{cases} \quad (2)$$

Considering our formalization up to now, we can assign initial values to an arbitrary node and we can check if a certain security aspect is relevant for a certain node. We now need two further conditions to be fulfilled for a propagation of security requirements. First, the source and the target node (v_n and v_m) have to be adjacent, which is expressed by Equation (3). The second condition is, that a directed edge between the source node and target node exists, which is expressed by Equation (4)

$$adjacent(v_n, v_m) \Leftrightarrow (v_n, v_m) \in E_m \quad (3)$$

$$inherit(v_n, v_m) \Leftrightarrow adjacent(v_n, v_m) \wedge \{(v_n, v_m) \mapsto v_m\} \in s_m \quad (4)$$

With the previously defined equations we are now able to specify the propagation of a security requirement $s_i \in S$ from one node to another one. Keep in mind, that the equations are only valid, if the security aspect is required for the considered data item. The complete equation for describing this propagation is given in equation (5).

$$delegate(v_n, v_m, s_i) \Leftrightarrow inherit(v_n, v_m) \wedge relNT(v_m, s_i) \wedge vecn_{v_n,d_m}[s_i] = 1 \quad (5)$$

Composing Graphs to Function Nets

After specifying the security requirements for all graphs of a function net in the described manner, these graphs have to be composed to the original function net (with the obtained information about security). Along with this process it has to be considered that nodes which occur in multiple graphs may only occur uniquely in the final function net. Hence, we use the *union* operation to describe this composition, which is expressed in Equation (6).

$$f_m = \bigcup_{k=1}^{j} G_j \text{with } k, j \in \mathbb{N},\ j \geq 1 \text{ as \#Graphs} \quad (6)$$

Furthermore we have to take into account, that at each node the information about security requirements is available for *every* data item which is processed at this node. For instance, the node *navigation unit* in Figure 2 manages several data items, e.g., *destination* or *traffic info*. In the composed function net the security requirements for each data item have to be stored separately. However, this is more a design and implementation problem than a conceptual one. One possibility is to store information about security aspects in vector-like structures for each data item.

6. Conclusion and Future Work

In this paper, we pointed out the importance of IT security in automotive systems and how does it relate to the software development process of such systems. As a result, we proposed to integrate IT security considerations from the very early beginning of this process, e.g., the requirement analysis. Therefore, we proposed an model-based approach, which aims at specifying security requirements within function nets, where a function net provides a view on the logical architecture of the system. In order to automate this specification process as far as possible, we furthermore introduced a formalization which describes, how security requirements can be propagated across a function net. Since this formalization is just a conceptual idea up to now, it could be used for implementing the security specification (e.g., by extending existing tools). However, the presented approach is just the result of a first, abstract

idea of increasing security awareness in automotive systems. It can be improved from several points of views. For instance, we intend to evaluate how this approach can be applied to Simulink models, which are widely-used in the automotive domain and where we meet rather signals than data items. Furthermore, we want to extend our own model representation, e.g., by using attributed or types graphs. This could be helpful by considering larger graphs with more than one data item. Moreover, it is reasonable to think about how to deal with composed or derived data. However, the probably biggest challenge is to find ways how to evaluate security specifications with respect to suitable countermeasures. In detail, we want do discover rules, which more or less automatically state, whether a countermeasure is needed and, if so, what is the most suitable one.

7. Acknowledgements

The work described in this paper has been supported in part by the European Commission through the EFRE Programme ”COMO” under Contract No. C(2007)5254. The information in this document is provided as is, and no guarantee or warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability. Furthermore, the authors thank Prof. Jana Dittmann and Prof. Gunter Saake for their advices and fruitful discussions durign creation of this paper.

References

[1] A. Pretschner, M. Broy, I. Krüger, and T. Stauner, “Software Engineering for Automotive Systems: A Roadmap,” in *Proceedings of Future of Software Engineering (FOSE) at the ICSE*, 2007, pp. 55–71.

[2] S. Thiel and A. Hein, “Modeling and Using Product Line Variability in Automotive Systems,” *IEEE Software*, vol. 19, pp. 66–72, 2002.

[3] P. Braun, M. von der Beeck, M. Rappl, and C. Schröder, “Model-Based Requirements Engineering for Embedded Systems,” in *Proceedings of the International Conference on Requirements Engineering (RE)*, 2002, pp. 92–94.

[4] E. Geisberger, J. Grunbauer, and B. Schätz, “Interdisciplinary Requirements Analysis Using the Model-based RM Tool AUTORAID,” in *Proceedings of the Workshop on Automotive Requirements Engineering (AURE) at Int’l Conf. on Requirements Engineering (RE)*, 2006.

[5] M. Broy, “Challenges in Automotive Software Engineering,” in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2006, pp. 33–42.

[6] C. Tischer, A. Muller, M. Ketterer, and L. Geyer, “Why does it take that long? Establishing Product Lines in the Automotive Domain,” in *Proceedings of the International Software Product Line Conference (SPLC)*, 2007, pp. 269–274.

[7] A. Lang, J. Dittmann, S. Kiltz, and T. Hoppe, “Future Perspectives: The Car and its IP-Address - A Potential Safety and Security Risk Assessment,” in *Proceedings of the International Conference on Computer Safety, Reliability and Security (SAFECOMP)*, 2007, pp. 40–53.

[8] T. Hoppe, S. Kiltz, A. Lang, and J. Dittmann, “Exemplary Automotive Attack Scenarios: Trojan horses for Electronic Throttle Control System (ETC) and replay attacks on the power window system,” in *Proceedings of the 23. VDI/VW Gemeinschaftstagung Automotive Security*, 2007, pp. 165–183.

[9] T. Hoppe and J. Dittmann, “Sniffing/Replay Attacks on CAN Buses: A Simulated Attack on the Electric Window Lift Classified using an adapted CERT Taxonomy,” in *Proceedings of the Workshop on Embedded Systems Security (WESS) at EMSOFT 2007*, 2007.

[10] E. Kaspersky, “Viruses coming aboard?, viruslist.com weblog-eintrag vom 24.1.2005,” http://www.viruslist.com/en/weblog?discuss=158190454\&return=1.

[11] M. Broy and W. Pree, “Ein Wegweiser für Forschung und Lehre im Software Engineering eingebetteter Systeme,” in *Informatik Spektrum*, 2003, pp. 3–7.

[12] A. Barisani and B. Daniele, “Unusual Car Navigation Tricks: Injecting RDS-TMC Traffic Information Signals,” in *Proceedings of the CanSecWest Conference*, 2007.

[13] C. Paar, “Remote keyless entry system for cars and buildings is hacked,” 2008, http://www.crypto.rub.de/imperia/md/content/projects/keeloq/keeloq_en.pdf.

[14] T. Hoppe, S. Kiltz, and J. Dittmann, “Security threats to automotive CAN networks practical examples and selected short-term countermeasures,” in *Proceedings of the International Conference on Computer Safety, Reliability and Security (SAFECOMP)*, 2008, pp. 235–248.

[15] I. Stürmer, M. Conrad, I. Fey, and H. Dörr, “Experiences with Model and Autocode Reviews in Model-Based Software Development,” in *SEAS ’06: Proceedings of the 2006 International Workshop on Software Engineering for Automotive Systems*, 2006, pp. 45–52.

[16] I. Stürmer, I. Kreuz, W. Schäfer, and A. Schürr, “Enhanced Simulink/Stateflow Model Transformation: The MATE Approach,” Proc. of MathWorks Automotive Conference (MAC 2007), Dearborn (MI), USA, Juni 2007.

[17] M. von der Beeck, “Function Net Modeling with UML-RT: Experiences from an Automotive Project at BMW Group ,” in *UML Modeling Languages and Applications*, 2005, pp. 94–104.

[18] H. Gröninger, J. Hartmann, H. Krahn, S. Kriebel, and B. Rumpe, “View-Based Modeling of Function Nets,” in *Proceedings of the Object-Oriented Modelling of Embedded Real-Time Systems (OMER4) Workshop*, 2007, pp. 40–45.

[19] Object Management Group, “SysML Specification Version 1.1,” Nov. 2008.

[20] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer, *Fundamentals of Algebraic Graph Transformation*. Springer-Verlag New York, 2006.