Extent and Characteristics of Dependencies between Vehicle Functions in Automotive Software Systems

Andreas Vogelsang
Institut für Informatik
Technische Universität München
Boltzmannstr. 3, 85748 Garching
Email: vogelsan@in.tum.de

Stefan Teuchert and Jean-François Girard
MAN Truck & Bus AG
Engineering E/E System Vehicle Dynamic Functions (EEV)
Dachauer Strasse 667, 80995 München
Email: {stefan.teuchert,jean-francois.girard}@man.eu

***Abstract*—Functional dependencies and feature interactions are a major source of erroneous and unwanted behavior in software-intensive systems. To overcome these problems, many approaches exist that focus on modeling these functional dependencies in advance, i.e., in the specification or the design of a system. However, there is little empirical data on the amount of such interactions between system functions in realistic systems. In this paper, we analyze structural models of a modern realistic automotive vehicle system with the aim to assess the extent and characteristics of interactions between system functions. Our results show that at least 69% of the analyzed system functions depend on each other or influence each other. These dependencies stretch all over the system whereby single system functions have dependencies to up to 40% of all system functions. These results challenge the current development methods and processes that treat system functions more or less as independent units of functionality.**

***Keywords*-functional specifications; feature interaction; model-based development; automotive; empirical studies**

I. Introduction

Functional dependencies and feature interactions [1] are a major source of erroneous and unwanted behavior in software-intensive systems [2]. They increase the complexity of the system and are often neglected by specification and validation approaches. Furthermore, these errors and unwanted behavior are revealed very late in a classical development process as they are validated in the phase of integration or even system test.

To overcome these problems, many approaches exist that focus on modeling these functional dependencies in advance, i.e., in the specification or the design of a system (e.g., [3], [4], [5], [6], [7], [8]). We have also proposed a formalism that structures functional requirements of a system into so called system functions [9]. These system functions formalize parts of the overall behavior of a system from an actor's point of view. This corresponds roughly to what is described in a use case and captured by scenarios modeled by a set of interaction patterns that describe how to use a system for specific purposes (cf. [10], [11]). We call the actor's input actions that are relevant for a system function its *primary inputs*. However, system functions may not be influenced solely by the primary inputs of the actor but also by the state and reaction of some other system function. A typical example for this is an *Adaptive Cruise Control* function that must be deactivated in case the *Pre-Crash* function indicates an upcoming crash.

This work was partially funded by the Deutsche Forschungsgemeinschaft (German Research Foundation), grant Inserve III, BR 887/19-3

A. Problem Statement

Although existing modeling approaches for these functional dependencies show promising results in specific examples and applications, there is little empirical data on the extent and characteristics of such functional dependencies in current software-intensive embedded systems. Therefore, it is not clear if these approaches address a real problem in modern software systems. Without a comprehensive understanding of the nature of these dependencies it is also not possible to evaluate the appropriateness of these approaches.

B. Research Objective

The overall research objective of this work is to understand the role and characteristics of functional dependencies in modern software-intensive automotive systems. We aim at a clear and precise understanding of such dependencies. This is necessary to develop approaches that focus on modeling functional dependencies in order to verify and validate behavior that arises by the complex interplay of system functions.

C. Contribution

Based on the analysis of a real-life automotive vehicle system at the company MAN Truck & Bus AG we contribute

- data on the extent and distribution of functional dependencies in a modern automotive system
- a characterization of these functional dependencies

The examined vehicle system covers the software part of a compact truck with a number of vehicle functions that are distributed over several electronic control units.

D. Context

The study concentrates on the automotive domain and is performed at MAN Truck & Bus AG, which is a German-based international supplier of commercial vehicles and transport systems, mainly trucks and buses. It has over 34,000 employees world-wide of which 150 work on electronics and software development.

The organization's development process is supported by an integrated data backbone developed on the eASEE framework from Vector Consulting GmbH. On top of this backbone, a complete model-based development approach has been established using the tool chain of TargetLink and Stateflow as modeling and simulation environment as well as for C-code generation. This process provides very precise data that enables significant analyses.

We analyzed the functional architecture of a novel compact truck. The functional architecture is represented by a logical data-flow architecture that describes the realization of system functions by logical components. System functions in the context of automotive systems are also called vehicle functions.

II. Related Work

To the best of our knowledge there is no comparable work on empirical data or analyses of realistic automotive or embedded systems with the focus on dependencies between system functions. However, there is a lot of work on approaches that try to model or specify such dependencies.

Functional dependencies and feature interactions have been extensively investigated in the telecommunication domain [12]. Jackson and Zave [5] introduced *Distributed Feature Composition (DFC)* as a modular, service-oriented architecture for applications in the telecommunication domain. DFC relies on the notion that a user service request can be composed of a set of smaller features, which are arranged in a *pipes-and-filters* architectural style. This architecture is especially designed to model interactions between different features.

Classical approaches like UML use cases, activities or sequences [13] specify system functions more or less in isolation. Dependencies between system functions are neglected. This makes it hard to reason about functionality that arises from the interplay of multiple system functions.

One of the most well-known specification technique for requirements is the *software cost reduction (SCR)* method [4]. In SCR, requirements are specified by a set of specification tables. The developers of SCR also noticed that understanding the relationship between different parts of a specification can be difficult, especially for large specifications [14]. Therefore, they integrated a Dependency Graph Browser that displays the dependencies among the variables in a given specification.

Further work exists that mentions functional dependencies as a major challenge for the development of future embedded software system, especially in the automotive domain (e.g., [15], [16]).

III. Study Design

In this section, we formulate the research questions, describe the study object as well as the data collection procedures. We then define the analysis procedure before we conclude with a description of how we ensure the validity.

A. Research Questions

The study aims at assessing the extent and distribution of dependencies between vehicle functions and characterizing them in order to derive important properties and principles that need to be supported by modeling approaches. In order to achieve these goals we formulate three research questions.

RQ 1: *To what extent do dependencies between vehicle functions exist?*

We focus on dependencies in the sense that the behavior of a vehicle function is not solely dependent on its primary inputs but also on the state or data of another vehicle function.

RQ 2: *How are dependencies distributed over all vehicle functions?*

We are interested in the question if dependencies are equally distributed over all vehicle functions, if there are certain vehicle functions that are central with respect to the dependencies.

RQ 3: *What categories of data characterize the dependencies?*

In order to develop and assess suitable modeling techniques and theories we have to understand the characteristics of the data that describes the dependencies. In this study we aim at extracting different semantic concepts that stand behind these dependencies. That means we want to find out what the concepts and notions are that characterize a dependency.

B. Study Object

We analyzed a vehicle system that describes the entire software architecture of a compact truck. The system comprises 55 fully specified[1] vehicle functions that are realized by an overall of 462 logical components[2]. Logical components describe the realization/implementation of a vehicle function in a purely logical fashion, i.e. without any information about the hardware the system runs on. A network of logical components describes the steps that are necessary to transform the input data into the desired output data. An example for a system that consists of

[1]The overall vehicle system comprises 142 vehicle functions. However, at the time of the analysis the system was not yet fully specified. Incompletely specified vehicle functions were excluded from the study.

[2]Many automotive companies and also MAN Truck & Bus AG call these logical components *(virtual) functions*. Since the term *function* can easily be mixed up with the term *vehicle function*, which describes a different concept, we use the term *logical component* instead (cf. [17]).

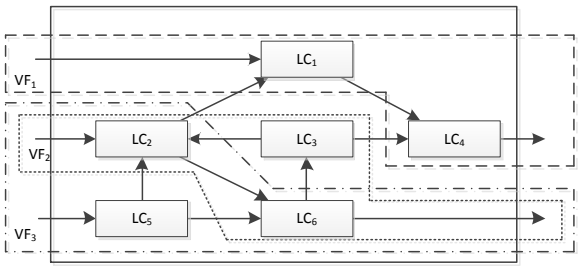

Figure 1. The logical components (rectangles) are connected by data channels (black arrows) and form a logical architecture of the system (outer rectangle). Vehicle functions crosscut this architecture by the set of logical components that contribute to their realization (dashed forms).

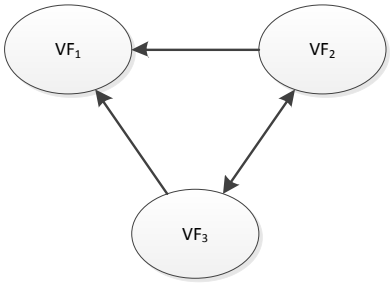

Figure 2. The vehicle function graph extracted from the logical component architecture of Figure 1.

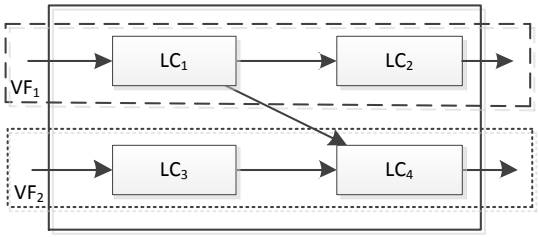

Figure 3. The vehicle function VF_2 depends on the vehicle function VF_1, since the logical component LC_1 (part of VF_1) sends values to the logical component LC_4 (part of VF_2). Thus, the behavior of VF_2 depends on data of VF_1.

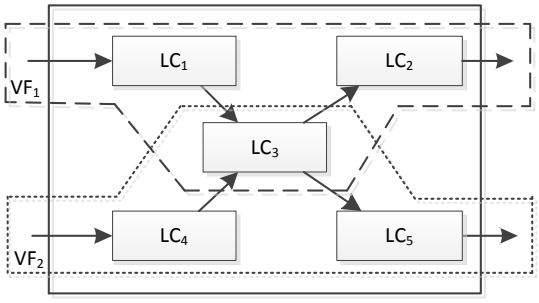

Figure 4. Vehicle functions VF_1 and VF_2 share the logical component LC_3. This might indicate a dependency between the vehicle functions. However, this cannot be verified without further knowledge about the behavior of LC_3.

3 vehicle functions that are realized by a network of 6 logical components is illustrated in Figure 1. The logical components are afterwards deployed to a set of electronic computing units that execute the behavior of the logical components.

The relation between a vehicle function and a logical component in the context of this study is the following: A vehicle function is realized by a set of logical components that are arranged in a data-flow network. A logical component can contribute to the realization of a set of vehicle functions. Thus, there is an $n : m$ relation between vehicle functions and logical components. The set of all logical components and their connections form a logical architecture of the entire system. The vehicle functions crosscut this architecture by the set of logical components that contribute to their realization.

C. Data Collection Procedures

A crucial question in this context is, what is considered to be a dependency between vehicle functions. In our initial informal definition we said that a vehicle function VF_1 depends on another vehicle function VF_2 if VF_1 is not only dependent on its primary inputs but also on the state or data of VF_2. Therefore, it is necessary to have some kind of communication between them. Communication between vehicle functions in the setting of our study object can be achieved in two ways.

1) A logical component that is part of one vehicle function has a communication channel to a logical component that is part of another vehicle function (see Figure 3).
2) A logical component is part of two vehicle functions, i.e., the vehicle functions share a logical component (see Figure 4).

A dependency between vehicle functions in the second case, however, cannot be ensured definitely without further knowledge about the behavior of the logical components. Consider the example given in Figure 4: Whether the output data transmitted over channel $LC_3 \rightarrow LC_5$ is really influenced by the input data transmitted over channel $LC_1 \rightarrow LC_3$ cannot be answered definitely without knowledge about the concrete behavior of LC_3.

As we had no information about the behavior of single logical components in our study, we only considered the

first case as a dependency between vehicle functions. This, at least gives a lower bound for the dependencies.

Our analyses eventually showed that this restriction does not adulterate the results excessively. Considering all dependencies of the second case as actually being dependencies, the number of the dependencies in the system just increased by around 10%.

Based on this definition of a dependency between vehicle functions, we can extract a vehicle function graph where each node is a vehicle function and a directed edge indicates a dependency between two vehicle functions. The resulting vehicle function graph for the example of Figure 1 is illustrated in Figure 2.

D. Analysis Procedures

In order to answer the research questions, we collected the following measures from the study object:

For RQ 1, we assessed the ratio of vehicle functions that are dependent on another vehicle function and counted the number of dependencies between vehicle functions and the number of channels that represent a dependency. This gives an impression of the extent of functional dependencies in realistic systems.

For RQ 2, we measured the dependency fan-in and fan-out as well as the PageRank for all vehicle functions on the vehicle function graph in order to see whether dependencies are distributed equally or if certain vehicle functions are more central than others.

For RQ 3, we classified the dependencies by the type of data that realize the dependencies. For this purpose, we examined the communication channels between two logical components that cause the corresponding vehicle functions to be dependent and classify the data that is transmitted over the channels. We group the data channels by their data type into three categories:

- Value: A data type that expresses a value on a continuous scale (a physical unit in most cases)
- Enum: A data type that expresses a value on a discrete finite scale with more than two values
- Bool: A data type that expresses a value on a scale with two possible values

This categorization should provide additional information about the semantic concepts that characterize the dependencies. We expect to get a better understanding of the nature of such dependencies from this categorization.

E. Validity Procedures

To ensure internal validity we analyzed the system under investigation at a stage where it was already subject to an architectural review. This way design flaws and misconceptions within the analyzed model should be reduced. Additionally, we presented and discussed the results with the developers and the responsible persons at MAN, who ensured that our results are valid and reasonable.

Table I
EXTENT OF DEPENDENCIES IN THE VEHICLE FUNCTION GRAPH

	Number	**Ratio**
all VFs	55	100%
VFs with incoming dependencies	36	65.5%
VFs with outgoing dependencies	29	52.7%
VFs with incoming and outgoing dependencies	27	49.1%
VFs without dependencies	17	31.0%

Table II
DISTRIBUTION OF DEPENDENCIES IN THE VEHICLE FUNCTION GRAPH

	Dependencies (Ingoing)	**Dependencies (Outgoing)**	**PageRank**
Maximum	10	23	6.47%
Median	1	1	1.26%
Minimum	0	0	0.72%

IV. STUDY RESULTS

In this section the results of the study are presented. They are structured according to the defined research questions.

A. Extent of Dependencies (RQ 1)

Analyzing the vehicle function graph, we found 133 dependencies between the 55 vehicle functions. 17 out of the 55 vehicle functions were completely independent from any other vehicle function and did also not have any influence on other vehicle functions. 36 vehicle functions were dependent on another vehicle function and 29 vehicle functions had an influence on another vehicle function. Table I summarizes these results. There were 135 different data channels that caused the dependencies.

B. Distribution of Dependencies (RQ 2)

The extent of the dependencies shows that dependencies between vehicle functions are distributed all over the system. However, there are some vehicle functions that are more central in the sense that they have a number of dependencies to other vehicle functions. Table II shows that vehicle functions have a maximum of 23 other vehicle functions that they influence, whereas one vehicle function depends on up to 10 other vehicle functions at maximum. The computation of the PageRank [18] gives an idea about the "importance" of single vehicle functions and deviates by a factor of almost 9. Figure 5 and Figure 6 show the distribution of the number of dependencies for all vehicle functions. The figures show that there is a small number of vehicle functions with a large number of dependencies, and a large number of vehicle functions without any dependencies. However, the sum of all vehicle functions with at least one dependency is larger than the number of vehicle functions without any dependency. Figure 7 shows the vehicle functions and their dependencies as a design structure matrix (DSM) [19].

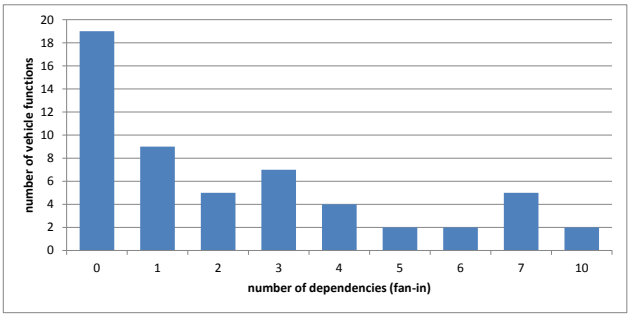

Figure 5. Distribution of the number of dependencies to one vehicle function (fan-in)

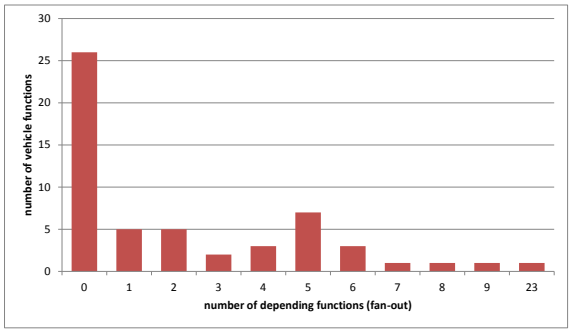

Figure 6. Distribution of dependencies to other vehicle functions (fan-out)

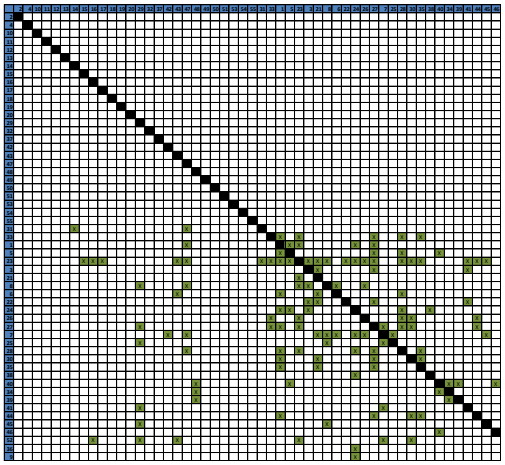

Figure 7. Vehicle Functions and their dependencies visualized as a Design Structure Matrix (DSM)

C. Categories of Dependencies (RQ 3)

As mentioned before there, were 135 different data types that were involved in one or more dependencies between vehicle functions. To get a better idea of the characteristics of these data, we categorized the data channels that modeled a dependency according to their data type and grouped them into the three categories: *Value*, *Enum*, and *Bool*. The goal of this categorization is to get information about the kinds of data that is associated with the dependency in order to get a better understanding how to characterize the dependencies. This is necessary to develop suitable modeling and verification methods for such dependencies. Table III shows the results of this categorization. On the left side of the table multiple occurrences of a data type are counted separately, whereas on the right side of the table every data type that contributes to any dependency is counted exactly once, independent from how often this data type contributes to any dependency. As the results show, this distinction has no influence on the distribution to the three categories. We additionally found that the channels of type *Enum* had 16 possible values at maximum. Our results show that more than 74% of the channels had a discrete and finite data type. One possibility to interpret these numbers is that a majority of the dependencies represent a propagation of state information or control commands from one vehicle function to another.

Table III
CATEGORIZATION OF DEPENDENCY CHANNELS ACCORDING TO THEIR DATA TYPE

	Overall		Unique	
	Number	**Ratio**	**Number**	**Ratio**
Value	122	26%	35	25.9%
Enum	314	66.8%	85	63%
Bool	34	7.2%	15	11.1%
Sum	470	100%	135	100%

V. DISCUSSION

Before interpreting the results with respect to their implications and relating them to existing work and evidence, we discuss the threats to validity.

A. Threats to Validity

A threat to the internal validity is the fact that the analyzed model is already a realization/implementation of the system functions. Dependencies might thus be a consequence of a design decision made by a developer and not an integral part of the system function itself. Although the analyzed model was not entirely completed at the time of the analysis, it was already subject to a first architectural review. Another threat pertains to the definition of dependency as given in this paper. Besides the explicitly modeled dependencies that are in the focus of this paper there may also be dependencies between vehicle functions that occur when feature are implicitly connected through a feedback loop through the environment.

As our analyses were only performed on a single system of one company the external validity cannot be assured. However, the analyzed architecture of vehicle functions is similar to those of other systems within the company and also reflects our impression of architectures in other automotive companies. An additional limitation for the application of the results to automotive systems, in general, is that the software systems in modern luxury cars might be even more complex and interconnected than the examined system of a compact truck.

B. Impact/implications

The conclusions we draw point at a number of problems that occur in today's development of automotive software systems. Current development processes handle vehicle functions more or less as isolated units of functionality [15]. This has to some extent historical reasons as the car industry managed to make their different functionality as independent as possible such that cars could be developed and produced in a highly modular way. With the coming up of software-based functions in the car this independence disappeared [15].

This is especially true for testing activities, where each vehicle function is tested separately. Behavior that arises from the complex interplay of vehicle functions due to its dependencies is not in the focus of these testing activities [2].

The extent and distribution of complex dependencies between vehicle functions also challenge the process and methods for modeling the requirements to a system. Each dependency between vehicle functions corresponds to a certain behavior that, in most cases, is observable by the user of a system and thus should reflect a functional user requirement. Therefore, a method for modeling system requirements should also account for dependencies between vehicle functions.

Some vehicle functions have a large number of dependencies to other vehicle functions. They capture functionality that has an impact on many parts of the system, e.g., start-stop systems or energy management. Other vehicle functions have dependencies only to a certain group of vehicle functions. Initial clustering analyses we performed on the vehicle function graph showed that the vehicle function graph can be clustered into groups of vehicle functions with a high cohesion between the vehicle functions of a cluster. This supports approaches that structure vehicle functions in function hierarchies.

C. Relation to Existing Evidence

We found little empirical data on dependencies in (vehicle) function architectures of automotive systems. However, our results back up the challenges of [20] and [15], where the authors state that functions [of a car] do not stand alone, but exhibit a high dependency on each other, so that a car becomes a complex system where all functions act together.

The fact that most dependencies are represented by channels that have a discrete finite data type support the approach of [9], where dependencies between vehicle functions are modeled by so called modes that represent a state of the vehicle. Additionally, vehicle functions are structured into function hierarchies in this approach, which might help to handle vehicle functions with many dependencies more systematically.

VI. Conclusions and Future Work

In this section, we summaries the conclusions from our analysis and describe directions for future research.

A. Summary of Conclusions

From the results we draw three basic conclusions:

1) Dependencies between vehicle functions are numerous and pervade the whole system. Our analysis shows that at least 69% of the analyzed vehicle functions depend on other vehicle functions or influence other vehicle functions.
2) Dependencies between vehicle functions are distributed over the whole system. However, some vehicle functions are more central than others. In our case, single vehicle functions have dependencies up to 23 out of 55 vehicle functions.
3) The information that represents a dependency is in most cases a value from a discrete finite set of possible fixed values. In our analysis around 74% of the channels that represented a dependency had an enumeration or boolean data type.

B. Future Work

As next steps we want to extend the study to other companies and vehicle systems. Based on the results of our work we will examine the possibility of introducing a mode model (see also [9]) within MAN to systematically capture the dependencies between vehicle functions and integrate them into a comprehensive specification method for vehicle functions.

References

[1] P. Zave, "Requirements for Evolving Systems: A Telecommunications Perspective," in *5th IEEE International Symposium on Requirements Engineering*. IEEE Computer Society, 2001.

[2] S. Benz, "Generating Tests for Feature Interaction," Ph.D. dissertation, Technische Universität München, 2010.

[3] S. Apel, C. Lengauer, B. Möller, and C. Kästner, "An algebraic foundation for automatic feature-based program synthesis," *Science of Computer Programming*, vol. 75, no. 11, 2010.

[4] C. Heitmeyer, "Using the SCR* Toolset to Specify Software Requirements," *Industrial-Strength Formal Specification Techniques, Workshop on*, p. 12, 1998.

[5] M. Jackson and P. Zave, "Distributed Feature Composition: A Virtual Architecture for Telecommunications Services," *IEEE Trans. Software Eng*, vol. 24, no. 10, 1998.

[6] K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study," Software Engineering Institute, Carnegie Mellon University, Tech. Rep. CMU/SEI-90-TR-21, 1990.

[7] D. L. Parnas, J. Madey, and M. Iglewski, "Precise Documentation of Well-Structured Programs," *IEEE Transactions on Software Engineering*, vol. 20, 1994.

[8] P. Zave, H. H. Goguen, and T. M. Smith, "Component coordination: a telecommunication case study," *Computer Networks*, vol. 45, no. 5, 2004.

[9] M. Broy, "Multifunctional software systems: Structured modeling and specification of functional requirements," *Science of Computer Programming*, vol. 75, no. 12, 2010.

[10] A. Cockburn, *Writing Effective Use Cases*, 1st ed. Addison-Wesley Professional, 2000.

[11] I. Jacobson, "Use cases and aspects-working seamlessly together," *Journal of Object Technology*, vol. 2, no. 4, 2003.

[12] M. Calder and E. H. Magill, Eds., *Feature Interactions in Telecommunications and Software Systems VI*. IOS Press, 2000.

[13] M. Fowler and K. Scott, *UML distilled - a brief guide to the Standard Object Modeling Language (2. ed.)*, ser. notThenot Addison-Wesley object technology series. Addison-Wesley-Longman, 2000.

[14] C. L. Heitmeyer, J. Kirby, and B. G. Labaw, "The scr method for formally specifying, verifying, and validating requirements: Tool support," in *ICSE*, 1997, pp. 610–611.

[15] M. Broy, "Challenges in automotive software engineering," in *Proceedings of the 28th international conference on Software engineering*. New York, NY, USA: ACM, 2006.

[16] M. Broy, I. Krüger, A. Pretschner, and C. Salzmann, "Engineering Automotive Software," *Proceedings of the IEEE*, vol. 95, no. 2, 2007.

[17] M. Broy, "Model-driven architecture-centric engineering of (embedded) software intensive systems: modeling theories and architectural milestones," *ISSE*, vol. 3, no. 1, 2007.

[18] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 17, 1998, proceedings of the Seventh International World Wide Web Conference.

[19] T. Browning, "Applying the design structure matrix to system decomposition and integration problems: a review and new directions," *Engineering Management, IEEE Transactions on*, vol. 48, no. 3, pp. 292 –306, Aug. 2001.

[20] A. Pretschner, M. Broy, I. H. Krüger, and T. Stauner, "Software Engineering for Automotive Systems: A Roadmap," in *2007 Future of Software Engineering*. IEEE Computer Society, 2007.