An Exploratory Study on Improving Automotive Function Specifications

Andreas Vogelsang
Technische Universität München, Germany
vogelsan@in.tum.de

ABSTRACT

In automotive system development, function specifications describe the requirements and basic design of a function. They are central artifacts and serve as inputs for several activities. With increasing complexity of functions in a vehicle, function specifications become harder to comprehend, change and validate. For the purpose of increasing our understanding of what the reasons for these impediments are and how they are related to derive potential for improvement, we conducted a qualitative (grounded theory) study. In this study, we interviewed nine senior practitioners of an automotive company on how they work with function specifications and which problems they encounter. In this paper, we show the results of this study and report on experiences and challenges of conducting a grounded theory study in industry.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications

General Terms

Documentation

Keywords

Requirements specifications, function specifications, industry, automotive software, grounded theory

1. INTRODUCTION

Function specifications are central artifacts in the development of automotive systems. They describe the requirements and the basic design of a *function*, which summarizes a specific part of functionality a vehicle exhibits. Function specifications are used for several purposes, e.g., documenting requirements, communication between OEM and supplier, calibration of system parameters in a specific product, or testing and implementing a function.

Depending on the company, function specifications are referred to as *function definition*, *function documentation*, *function specification*, *function requirements documentation*, or *software and calibration documentation*. This artifact is a written documentation of a function's requirements and its abstract design. In many automotive companies, models such as dataflow specifications are used to augment textual descriptions (cf. [9]). These dataflow specifications may be either used to generate code for the function (e.g., by using Matlab Simulink or ASCET models) or the models just serve as an overview and the function is implemented separately (e.g., as plain C code).

The special role of function specifications in the automotive sector originates from the fact that function specifications are not solely a documentation for function developers. They are the basis for a number of activities and they are read by function developers, system integrators, functions testers and calibration engineers.

Over the last 25 years, the number of functions (and thus the number of function specifications) in a vehicle increased and the interaction and logical dependencies between functions became more relevant for their specification [10].

As a consequence, developers and testers have difficulties to comprehend the complex function specifications, to distinguish between requirements, which need to be fulfilled and design decisions, which can be altered, and to assess the impact of changes [3]. It is, however, unclear, what the reasons for these difficulties are. What are the challenges the developers and testers face? How are these challenges related? How can function specifications be improved to support development activities in a situation, where function complexity and coupling increases.

Answering questions like these calls for an in-depth study on working practices and impediments in a community of people working with function specifications. In this paper we present such a study revealing activities performed on the basis of function specifications, challenges faced when working with them and relations between these challenges.

From a series of collaborative projects with a number of automotive companies in the last ten years, we made the following observations concerning the current state of function specifications in practice:

- Function specifications are merely a documentation of the implementation and do not describe the requirements explicitly.
- Function specifications are hard to comprehend for people who are not involved in the development of these functions.

CESI '14, June 2, 2014, Hyderabad, India

- Function specifications are not complete, in the sense that some details that are necessary to understand the function are only present in the source code.

To explore and understand these observations and their implications, we conducted an exploratory study using a grounded theory approach [2] to conduct and analyze semi-structured interviews (lasting 1–2 hours) with nine practitioners from an automotive company regarding their working practices and experiences with function specifications.

In the study, we elicited eight activities performed on the basis of function specifications. The results show that there is a strong need for a high-level (more abstract) description of functions with less technical details to support the comprehension of functions. In addition, the results show that the description of abstract states of a function is becoming more important and should be integrated explicitly into function specifications.

We challenged our outcomes by a second round of interviews, in which practitioners expressed their (dis)agreement with specific outcomes of our study.

2. STUDY DESIGN

The purpose of this study is to analyze *function specifications* for the purpose of *identifying and relating challenges* with respect to *comprehensibility* from the point of view of *function developers, system integrators, function testers and calibration engineers* in the context of an *automotive software company*.

2.1 Research Questions

In the following we will describe the research questions that we were interested in and the research method that we used to answer these questions.

RQ1: Which activities are performed on the basis of function specifications?
We are interested in process steps and activities that use the function specification as input or output artifact. This research question contributes to the goal of exploring the usage and requirements for function specifications.

RQ2: What are the main challenges experienced when working with function specifications?
Where are problems and shortcomings when working with function specifications? What might be reasons for them? This research question contributes to the goal of improving function specifications for different stakeholders.

RQ3: What are causal relations between the challenges?
Can the challenges be related in a causal way that they result in a comprehensive problem description? This research question contributes to the goal of understanding the reasons for problems in comprehending and specifying a function.

2.2 Research Method

To answer the research questions, we followed a *grounded theory* (GT) approach, an exploratory research method originating from the social sciences, becoming increasingly popular in software engineering research [2]. GT is an inductive approach, in which interviews are analyzed to derive a theory. It aims at discovering new perspectives and insights, rather than confirming existing ones.

As part of GT, each interview transcript was analyzed through a process of *coding*: breaking up the interviews into smaller coherent units (sentences or paragraphs), and adding

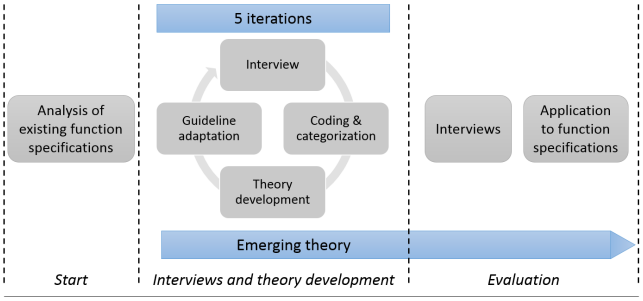

Figure 1: Phases of the study.

codes (representing key characteristics) to these units. We organized codes into *categories*. To develop codes, we applied *memoing*: the process of writing down narratives explaining the ideas of the evolving theory. When interviewees progressively provided answers similar to earlier ones, a state of saturation was reached, and we adjusted the interview guidelines to elaborate on other topics (cf. [5]).

We applied this research method in a single automotive company based in Germany with a large number of employees responsible for the research and development of software-intensive vehicle systems. The company has longtime experiences in the development of automotive systems and continuously improves its development methods and techniques.

The development processes and techniques within the company vary depending on the application domain and range from full-fledged model-based approaches with code generation to conventional code-based development. Requirements and specifications for all systems are documented in function specification documents. Systems are integrated on the basis of a signal database, which consists of all signals exchanged over a bus system or within an electronic control unit (ECU).

The company is responsible for the entire development process of single systems and functions starting from specifying requirements over system design, implementation, testing and calibration for existing vehicles.

2.3 Study Execution

We conducted the study in three phases that are illustrated in Figure 1 and described below.

2.3.1 Start

To familiarize ourselves with the development process and terminology used at the company, we analyzed two exemplary function specifications from the engine control domain.

To discuss relevant aspects in a more concrete way, we additionally handed out examples of function specifications in the conducted interviews.

2.3.2 Interviews and Theory Development

In the second phase of the study we conducted five interviews following a rough interview guideline with seven participants overall (one of the interviews had three participants). After each *interview*, the interview transcript was subject to a *coding and categorization* process. The codes were afterwards integrated and related to an emerging theory describing how developers work with function specification and which impediments they experience. If necessary, the guideline was adapted for the next interview with respect

to the evolving theory and possible new insights. In the following, the different steps of this phase are described in detail.

Data collection interview The interviews followed an interview guideline to cover all relevant aspects. The interview guideline contained open questions about activities performed with function specifications and their role in the development process, used terminology, importance of specific content for an activity, related artifacts, missing information and comprehensibility of function specifications. The interviews lasted between 1 and 2 hours and a written transcript was created for each interview.

Coding and categorization We broke up the interview transcript into smaller coherent units (sentences or paragraphs), and added *codes* (representing key characteristics) to these units (*open coding*). We afterwards grouped the codes into categories. We created the categories such that they reflect activities that are performed on the basis of a function specification.

Theory development Based on the codes and categories, we started looking for causal relations between them (*axial coding*). We annotated these relations with narrative explanations (memos), which we discussed and evaluated with the practitioners in the following interviews. The emerging theory consists of a set of codes grouped into categories representing activities, and relations between the codes representing causal relations between them.

Guideline adaptation As the theory emerged during the interviews, we adapted the guidelines to explore and validate new insights and hypotheses originating from the theory.

The process of data analysis consisted of a *constant comparison* of text units, codes, and categories, i.e., data was collected and analyzed simultaneously [2].

2.3.3 Evaluation

The theory development process of the second phase resulted in a set of hypotheses and explanations about which content of function specifications is important for which activity, where the developers encounter impediments and how these relate to each other. In the third phase of the study, we focused on the most frequent codes and categories and explored them in more detail to refine and evaluate related hypotheses and to derive implications for improvements. In the following, the different steps of this phase are described in detail.

Evaluation interviews We conducted two interviews, in which we presented the theory developed in the second study phase together with the hypotheses concerning the most prevalent codes and categories. We asked the interviewees to assess the validity of our theory and hypotheses and explain the relations between the core concepts in more detail (*selective coding*). A written transcript was created for each interview.

Application Besides the qualitative evaluation of the theory in the phase 3 interviews, we additionally developed an improved function specification structure based on the results of the developed theory. We tried to apply this new specification structure to the functions explored in the first phase of the study and assessed it with respect to feasibility and improvements. The resulting new specifications were also discussed in the interviews of the third phase.

Table 1: List of roles and corresponding participants

Role	Participants
System integrator	P1, P9
Function tester	P2, P3
Function developer	P3, P5, P7
Calibration engineer	P6, P8

Table 2: Categories of the theory ranked by the total number of appearances of codes related to the category. The middle column indicates the number of codes related to a category.

Category	Codes	Code app.
Comprehending a function	13	46
Illustrating the principle of operation	6	29
Documenting requirements	4	22
Structuring a function	5	21
Tracing requirements	3	14
Testing a function	7	13
Calibrating a function	4	13
Creating a specification	5	8

2.4 Participant Selection

For the interviews, we selected professionals with experience in working with function specifications between 5 and 10 years. We selected the participants based on their availability and experience and with the goal to cover as many roles as possible to explore the whole spectrum of activities performed with function specifications. Table 1 lists the participants and their roles.

System integrators are responsible for the integration of several functions into a system. *Function testers* check the correctness of function implementations according to their specification. *Function developers* develop a solution concept for a function based on its high-level requirements and implement this concept. *Calibration engineers* configure a function for a specific product by fixing its parameters.

3. STUDY RESULTS

In the seven interviews, we applied 53 different codes in total, which were afterwards categorized into eight categories.

3.1 Performed Activities (RQ1)

As mentioned before, we categorized the codes according to activities or processes that are performed on the basis of function specifications. Table 2 shows the eight categories and ranks them by the total number of appearances of codes related to the category.

According to this ranking, the *comprehension of a function* was the activity that is addressed most in the interviews. Especially in the interviews with function developers and

Table 3: Top 5 codes with appearances in at least half of the interviews. The second column indicates the total number of appearances over all interviews.

Code	Category	#
Highlight the basic principle of operation	Illustrating the principle of operation	15
High-level description of a function	Comprehending a function	8
Deriving requirements from change requests	Documenting requirements	8
Hierarchical structures	Structuring a function	6
Structuring modes and phases	Structuring a function	5

calibration engineers, the comprehension of a function accounted for a large part of the codes applied to the units of the interviews. P4 states: "I have to comprehend what a function does" and "The demand is to describe *what* a function does and not *how*".

The second most frequently mentioned activity is *illustrating the principle of operation* of a function. By this activity, interviewees referred to the necessity to illustrate a function at a high level of abstraction without any details of implementation. P4 says: "A physical description of a function is more useful in the beginning." and P6 states: "For an initial understanding, I refer to keywords or names of components. For technical details I might as well just read the code.".

Documenting requirements on the third place refers to the importance of the function specification artifact as a communication means for changing requirements. P5 explains: "Change requests addressing the function specification are the real requirements. If these are incorrectly documented, something wrong gets implemented.".

Structuring a function is an important activity to support the comprehension of a function. In the interviews, two main structuring principles were mentioned for functions: hierarchies and modes. P5 mentions: "Function specifications are hierarchically structured and may also be embedded in a system specification containing also other function specifications.". Modes were mentioned as suitable structuring principle for functions running through a set of phases. P4 explains: "If there are logically independent parts within a function, a mode-based structure is better than a purely dataflow-oriented model of the function.".

Tracing requirements, *Testing a function* and *Calibrating a function* are additional activities that are performed on the basis of the function specification artifact. Codes related to the *creation of a function specification* were only assigned to 6 units in the interviews.

3.2 Main Challenges (RQ2)

To name the main challenges faced, when working with function specifications, we list the five most frequently assigned codes that appeared in at least half of the interviews. Table 3 shows the results of this list.

The most frequently assigned code is *highlight the basic principle of operation*. It describes the necessity to understand the (physical) concept that the software function needs to support (P2,P4,P6,P7,P8). P4 says: "At first, we focus on the physical aspects of the function. States and implementation details of the software come later."

A *high-level description of a function* was mentioned eight times (P2,P3,P4,P5,P6). This code was applied mainly in the context of comprehending a function, especially in the context functions someone is not familiar with. P2 claims: "It is possible to concisely describe the purpose of any function in five sentences.".

Third follows *deriving requirements from change requests* (P1,P3,P4,P5,P7). This code addresses the fact that most functionality is not developed from scratch but is driven by change requests applied to older versions of a function. P3 explains: "We come from initially assuming that there is already an existing solution related to this kind of problem. In many cases we then employ the code of the existing version as the specification for the new version.". Additionally, due to the tight collaboration between car manufacturers and suppliers in the automotive industry, the car manufacturers are very familiar with the implementations of a supplier and vice versa. A consequence of this is that change requests may be formulated on very different levels of abstraction.

Hierarchical structures were mentioned six times in the interviews as means to cope with complexity (P1,P5,P6,P7, P9). P6 explains: "Especially, if there are long and complex relations between an input and an output signal, a hierarchical structure facilitates the calibration of a function.".

Structuring modes and phases of a function was another issue that was mentioned in the interviews (P3,P4,P7,P8,P9). Describing the lifecycle of a function in terms of abstract states was considered to have a positive impact on the understandability. P3 says: "In most cases an abstract state-based view onto a function is sufficient to understand the general purpose of it." Another aspect mentioned in the interviews was to define dependencies to other functions based on modes. P7 explains: "A state-based representation of dependencies between functions is more suitable than a description solely based on data exchange.".

3.3 Relations between Challenges (RQ3)

We related the codes to each other and annotated the relations with explanations. We will present the most relevant results of this process by describing three stories, in which codes are related by causal relations. In the stories, codes are written in italics. Each story contains a hypothesis (the heading of the story) that is backed up by the relation of codes within the story that follows the hypothesis.

Comprehending a function is supported by documenting the basic principle of operation and its logical modes of operation: To comprehend a function, a *high-level description of the function* is needed. This description should *highlight the basic principle of operation. A pure documentation of the code is not sufficient* as it hides the underlying physical principles. However, the basic principle of operation describes, in many cases, only the behavior of a function, when all enabling conditions are already fulfilled. *The logic that is embedded in the numerous enabling conditions is often neglected* in the function specifications. For the comprehension of a function these logical conditions must be part of the function specification. The description of these logical conditions can be supported by *structuring the functionality according to modes of operation.*

Calibration of a function is supported by a clear description of the black-box interface of a function:

For the calibration of a function, the *function parameters have a strong influence on the behavior*. The logic and logical conditions of a function are also highly parametrized and the *logic constitutes a large part in many functions*. Logical conditions hamper the traceability of signal flow through a function design. However, *tracing signal flow is an important activity* for calibration engineers. *Understanding the black box interface* of a function is another way of tracing signals that are passed through a function. Therefore, the black box interface description of a function should contain the *most relevant signals for the physical principle of operation* and the signals influencing the logical conditions of the function.

Logical conditions should be integrated into function specifications: *The code of a function contains more details* than is documented in the function specification. In particular, logical conditions are often only contained in the code. These information about enabling conditions and logical interactions need to be integrated into the function specification because *a pure documentation of the code is not sufficient. The logic constitutes a large part in many functions* and thus it is all the more necessary to integrate this part conveniently into function specifications.

4. DISCUSSION AND IMPLICATIONS

From the results of the study, we draw two conclusions:

Conclusion 1: A high-level description of the basic principle of operation of a function is necessary to comprehend a function. This high-level description should also incorporate the logical conditions that influence the function behavior (related codes: *Highlight the basic principle of operation*, *High-level description of a function*).

Conclusion 2: In the past, the complexity of a function was mainly driven by its control engineering part. Nowadays, the logical part of a function, i.e., its enabling conditions, its modes of operation, etc., accounts for an increasing part of the complexity. There are even functions that are trivial in the controlling of some values but highly complex in the situations, in which this controlling should be applied. This logical complexity should be considered in the function specifications (related code: *Structuring modes and phases*).

Especially the latter conclusion is well illustrated by an anecdotal example: In automobiles, there is a number of software functions, which, over time, learn an adaptation factor and apply it to the measured value of a specific sensor to compensate for incorrect measurements for example due to dirt accumulation. Specifying this learning and application process is relatively easy in terms of control engineering theory (e.g., by giving a differential equation). However, for this function additional complexity is added, when it comes to specifying the conditions that need to be fulfilled before the learning process can be started. Furthermore, these conditions may also require interactions with other functions (e.g., requesting the permission of a scheduler). Before applying the actual control cycle, the function needs to pass several phases and conditions that account for a large part of the complexity. We speak of the logic of a function.

In many function specifications this logical part is just somehow added to the initial control engineering model, which does not really cater for such kinds of specifications. The phases and states of the function are no longer comprehensible in the resulting function specification.

5. CREDIBILITY AND LIMITATIONS

A potential threat to the credibility of our findings is that we misinterpreted the data we collected and thus derived a theory that does not fit the data. To mitigate this threat, we performed *data triangulation* by reflecting our findings with a number of existing function specifications and *member checking* by presenting and discussing our findings and interpretations with the interview participants. We conducted the interview study over a period of eight months, in which findings and interpretations were constantly discussed and applied to existing data.

Another threat is a possible bias of the researcher. We tried to mitigate this threat by *making the researchers' intentions clear* and *co-opting* two additional researchers from the automotive company who participated in the interviews and the discussions of findings [8].

We used the frequency of code occurrences as an indicator for importance. This might be wrong since "*challenge x* is not important" might contribute to the importance of a code related to *challenge x*. We tried to mitigate this threat by separating negative quotations related to a code into another code (e.g., *code documentation is not sufficient*).

The interviews and the coding and categorization process were conducted in German and translated only for the presentation in this paper. To mitigate the threat of incorrect/improper translations by the authors, the translations were checked by experts of the domain.

We conducted the interviews only in one company, which poses a threat to the generalizability of our findings. However, in our experience, gained in a number of projects with other automotive companies, the situation faced in the observed company is similar in other automotive companies.

6. STUDY RETROSPECTIVE

During the study, we encountered challenges concerning the research method, its execution and the results. Here, we describe these challenges and ways to address them.

Research Method: Grounded theory (GT) as a research method facilitated the systematic collection and analysis of qualitative data for exploratory purposes. The conclusions drawn can be traced back to the level of quotations supporting these conclusions. However, we encountered some practical issues, for which the GT literature does not give any concrete hints, especially with regard to the analysis of the data. For the organization of codes into categories, for example, the GT literature just says that the categories "emerge" from the codes [2]. To operationalize this vague description, we decided to extract categories that represent activities or processes. This turned out to be a good choice not only since we were specifically interested in these activities (RQ1) but it also alleviated the relation of codes with each other. For this purpose, we built a large mindmap containing the categories, the related codes and references to all quotations associated with the codes. On this basis, we compared the codes pairwise, starting with the most frequent codes, and tried to find causal relations between them. When we found a hypothetical causal relation between two codes, we added an edge to the mindmap and annotated the edge with a memo containing a causal explanation. These hypothetical relations were assessed in the next iteration of interviews by means of the memos. Persistent relations and their describing memos formed the basis for the results of

RQ3. The resulting stories were particularly helpful for the dissemination of the study results.

Study Execution: GT itself does not prescribe a specific method for conducting interviews. In our study, we conducted the interviews based on a guideline, sketching out a rough catalogue of topics and questions. Additionally, we provided existing function specification documents as basis for discussion, which helped to clarify statements of the participants by pinpointing examples in the documents. We experienced that strictly following the interview guidelines were, in most cases, not possible, which was not a big issue, since interesting results often arose from discussions initiated by a question and then continued with a deeper explanation of personal experiences of the participants. Nevertheless, the guidelines helped us to stay focused and, in the later phases, assess the hypotheses of the emerging theory.

Study Results: Although a major goal of a research method is to produce *reliable* results, i.e., repeating the study leads to equal or at least similar results, the results gained from a GT approach may be biased for several reasons, e.g., the chosen codes and their organization in categories. However, GT, at least, makes the process of deriving the results more transparent. Thus, other scientists may investigate the categories, codes and the underlying original data to support or oppose the results.

7. RELATED WORK

We have found only few related work that considers empirical data and evidences for the use of function specifications in industry. Adam et al. [1] report on lessons learned from several RE process improvement case studies with small and medium sized enterprises applying the Fraunhofer IESE ReqMan approach. In their paper, they reported that 80% of their case study companies are challenged with writing requirements for the developers in an appropriate manner. They additionally define four *hot topics*, where the companies plan to make improvement, which are "elicit functional requirements", "elicit non-functional requirements", "review requirements", and "document developer requirements". The results presented in our paper can contribute to these topics by providing causes and explanations for the stated problems.

Gross and Doerr [6] describe the vision and advantages of view-based requirements specifications. They argue that in order to create high-quality requirements specifications that fit the specific demands of document stakeholders, the research community needs to better understand the particular information needs of downstream development roles. Our results provide answers to these information needs. They also show that, depending on the task to fulfill, some information within the specification document is more relevant than others. This supports the idea of view-based specifications.

Ferrari et al. [4] state that the quality of a natural language requirements document strongly depends on its structure. A proper document structuring enables a better understanding of the requirements, and eases the modifiability of the overall requirements specification. In their work, they provide an automatic approach to evaluate the *relatedness* of requirements in a document by means of clustering algorithms. With our approach, we follow the same direction.

Besides the academic approaches there exists a number of standards and templates for the specification of functions. In IEEE standard 29148 [7] for software requirements specifications (SRS) and system requirements specifications (SyRS). The standard recommends specifying system modes or states. "Some systems behave quite differently depending on the mode of operation. For example, a control system may have different sets of functions depending on its mode: training, normal, or emergency." [7]. The SRS proposes an alternative specification structure for systems with modes. However, the standard does not consider different levels of abstraction.

8. SUMMARY AND FUTURE WORK

In this article, we reported on an interview-based study we conducted to explore and expose challenges and shortcomings in the current specification of automotive functions. In a retrospective, we discussed challenges and success factors concerning the used research method and its execution. As a result of this study, we identified two major problems in function specifications, namely the separation of the basic principle of operation from the technical implementation and the inadequate specification of logical conditions, states and phases of a function. As follow-up research for this study, we plan to propose a new structure for function specifications and evaluate the proposed specification structure with respect to the findings of this study.

9. ACKNOWLEDGEMENTS

The author would like to thank the colleagues from the company, who conducted the study with the author. We thank all interviewees for their commitment. The work was partly funded by the German Federal Ministry of Education and Research (BMBF), grant "FoMoStA, 01IS12028".

10. REFERENCES

[1] S. Adam, J. Doerr, and M. Eisenbarth. Lessons learned from best practice-oriented process improvement in requirements engineering: A glance into current industrial re application. In *REET*, 2009.

[2] S. Adolph, W. Hall, and P. Kruchten. Using grounded theory to study the experience of software development. *Empirical Software Engineering*, 2011.

[3] M. Broy. Challenges in automotive software engineering. In *ICSE*, 2006.

[4] A. Ferrari, S. Gnesi, and G. Tolomei. Using clustering to improve the structure of natural language requirements documents. In *Requirements Engineering: Foundation for Software Quality*. Springer Berlin Heidelberg, 2013.

[5] M. Greiler, A. van Deursen, and M.-A. Storey. Test confessions: A study of testing practices for plug-in systems. In *ICSE*, 2012.

[6] A. Gross and J. Doerr. What you need is what you get!: The vision of view-based requirements specifications. In *RE*, 2012.

[7] IEEE. Systems and software engineering – life cycle processes – requirements engineering. *ISO/IEC/IEEE 29148:2011(E)*, 2011.

[8] A. Onwuegbuzie and N. Leech. Validity and qualitative research: An oxymoron? *Quality & Quantity*, 2007.

[9] C. Robinson-Mallett. An approach on integrating models and textual specifications. In *MoDRE*, 2012.

[10] A. Vogelsang and S. Fuhrmann. Why feature dependencies challenge the requirements engineering of automotive systems: An empirical study. In *RE*, 2013.