On the Model-based Documentation of Knowledge Sources in the Engineering of Embedded Systems

Marian Daun, Jennifer Brings, Bastian Tenbergen, Thorsten Weyer

paluno – The Ruhr Institute for Software Technology
University of Duisburg-Essen, Germany
{marian.daun, jennifer.brings, bastian.tenbergen, thorsten.weyer}@paluno.uni-due.de

Abstract: In the development of embedded systems the context is of vital importance, as embedded systems interact with the context through sensing and actuation. Information about the system's context is contained within different knowledge sources and must be elicited and negotiated during embedded systems development. Examples for such knowledge sources may be: laws, standards, internal process specification, systems in operation as well as stakeholders. Model-based documentation of these knowledge sources supports the analysis of the context (e.g. to aid in prioritizing of requirements, to resolve conflicts between knowledge sources, to trace the impact of changes in the context towards the system, or to gain certification of safety-critical systems). Most approaches dealing with knowledge sources are limited to elicitation and negotiation, but lack proper documentation techniques. Therefore, this paper sketches an approach that addresses the documentation of the context of knowledge, to make knowledge about the sources of contextual information comprehensibly persistent. The corresponding models of the contexts of knowledge can e.g. be used to structure the processes of requirements elicitation and context analysis.

1 Introduction

Embedded systems are highly dependent on their context. Software-intensive embedded systems (hereinafter: systems) observe their environment through sensors and manipulate it using actuators. Therefore, the context of such systems must be carefully considered during their development [MP84], [Da93]. Without proper consideration of the system's context the functional suitability, the performance, or the safety of a system may be threatened [HRH01]. Consider an automotive door control unit as a simplified example: Many modern cars lock the doors automatically at a certain velocity. For passenger safety, it is important that the doors also automatically and rapidly unlock in case of an accident. Both these functions depend on context information. From the perspective of the door control unit, the vehicle state (*driving* vs. *accident occurred*), are relevant context information which must be observed and evaluated. Specifically, it is important for the control unit to handle the information correctly, i.e. by locking the doors given a specific speed and unlock the doors immediately upon impact. There have been

approaches suggested to document existing context information, specify context assumptions, or to check context assumptions against actual context information, of either the system's context, or the system in its context (see e.g. [AH01], [Ja95], [SDP12]). However, these approaches mainly pertain to the *operational context*, i.e. the part of the context the system directly interacts with during operation. Hence, the importance of the operational context has been recognized. However, the system's context does not only pertain to its behavior during operation: context information may furthermore constrain the development itself. For example, laws in different countries may prohibit or mandate system functionality, e.g. different laws pertaining to the maximum permissible tint in car windows in the United States vs. Germany may require an automated window tinting systems to limit the tint (see the German road regulations [StVZO], and the Official Code of Georgia [Georg]). Such context information hence constrains both the system as well as its development and may give rise to requirements which the system must fulfill [PU13].

All of this context information stems from different knowledge sources. Since many knowledge sources, and particularly the context information they provide, cannot be completely known or may change during development, some of the context information must be assumed and subsequently validated for correctness. A prerequisite for this is that these knowledge sources, their relationship with one another, and their relationship to the system under development (SUD) are properly documented (cf. [ARP96]). The documentation of the context of knowledge aids in the analysis of the system's context as well. This is necessary to detect and resolve conflicts and contradictions between context information of different knowledge sources. Therefore, in this paper, we argue that explicit documentation of the context of knowledge in the engineering process of embedded systems is necessary and propose a context modeling approach that allows creating a map of the context of knowledge by documenting knowledge sources. To do so, we first elaborate on the question why documenting information concerning the context of knowledge is important for the engineering process of an embedded system in Section 2. Afterwards, Section 3 briefly reviews the related work on documenting knowledge information in software engineering development. In Section 4, we detail the fundamental ontology underlying our concept of the context of knowledge. Thereafter, Section 5 deals with the diagrammatic documentation of the context of knowledge and Section 6 shows how views onto context of knowledge models can be generated. Section 7 summarizes the paper and gives a brief outlook.

2 Needs for Explicit Documentation of the Context of Knowledge

Explicit documentation of the context of knowledge provides several benefits. On the one hand, information from the context of knowledge can aid in guiding the developers in eliciting necessary and relevant information for system development. By systematically analyzing different categories of knowledge sources, context information can be documented depending on their respective source. This also aids in monitoring the knowledge sources for changes that affect the system and thereby lead to changes in the systems requirements, functional design, logical, or technical architecture. On the other hand, the context of knowledge can be used to identify conflicts between knowledge

sources and to detect missing knowledge sources. Specifically, the benefits of explicit documentation of information from the context of knowledge include:

- **Separation between Context and System.** The context of knowledge allows developers to define the system boundary [Po10], i.e. the scope of development. This allows for ascertaining what aspects of the system are subject to engineering activities and what aspects of the development project are context information, which constrains the system and/or its engineering process.
- **Correct Definition and Interpretation of Requirements.** Requirements must be defined in a concrete context of use [PU13]. Therefore, it is necessary to know where the requirements stem from, which is relevant for, e.g., requirements prioritization [LKK04]. For example, the information that an automotive window tinting system shall be sold for cars in the United States as well as in Germany make it necessary to know the respective laws pertaining to the maximum tint level. Only when the respective knowledge sources (i.e. US vs. German road traffic regulation) are identified and are documented, will the developers be able to validate the correctness of the requirements for the United States version and the German version of the window tinting system.
- **Persistence and Availability of Knowledge.** It is of significant importance for any company to keep its expertise in house and available. By contracting out, many original equipment manufacturers risk expertise emigration, i.e. the slow and stepwise drain of knowledge pertaining to development projects to suppliers, as such expertise is typically and exclusively bound to single persons. The result may be loss of technological leadership, market share, and may lead to insolvency [KW93].
- **Support for the Decision Making Process.** Knowledge sources may be conflicting with one another. For example, an organizational guideline may prescribe testing to be conducted in a certain way using certain tools and a certain level of coverage. However, when stakeholders such as project managers set too ambitious deadlines and strict budget restrictions, meeting the requirements prescribed by the organizational rules may be hard, even impossible [St12]. In this case, constraints from either knowledge source (organizational guideline vs. stakeholder) must be negotiated and possibly prioritized. A prerequisite for this, however, is to know that this conflict exists, which can only be done when the relevant knowledge sources have been identified and context information has been documented.
- **Identifying and Documenting all Relevant Knowledge Sources.** Identifying all relevant knowledge sources such as stakeholders is a prerequisite for valid requirements [Po10]. If important knowledge sources are omitted, there is a risk that the requirements specification will not be complete or otherwise defective, potentially leading to increased development time and cost [Bo81].
- **Support for Change Management.** Changes in the context of a system may defect its suitability, functionality or safety [ISO10]. Changing the interface of a composite system results in changes in the system itself. Not only must changes in the operational context be detected, even changes in the knowledge sources belonging to another system may result in defects. For example, in case a law is changed, it must be investigated whether or not the system must be adapted.

3 Related Work

The importance of considering the operational context during development of embedded systems has frequently been argued (see e.g. [BB12]). Model-based documentation of information of a system's operational context has been argued to foster specific challenges inherent to the development of embedded systems, such as increasing complexity, interoperability, and context sensitivity (e.g. [Ja95], [DTW12]). Therefore, most commonly UML-based profiles and extensions are suggested to properly document context aspects (e.g. [BC06], [Dh09]). Of course, beside these extensions addressing the explicit documentation, context properties are often embedded within diagrams specifying the system. This is, for example, the case in sequence diagrams. Furthermore, it has been suggested to define the operational context in order to ease model checking (e.g. [Dh09]) and validation (e.g. [We10]) of development artifacts. Other modeling theories suggest the explicit specification of the context and the system in parallel [AH01]. Along with other benefits, doing so allows the explicit verification of the system against specific changes within the context.

Particularly in the requirements engineering literature, explicit consideration of context aspects, also of aspects that are not operational in nature, is considered a basic building block for good requirements [Po10]. While some approaches to structure context information have been proposed (e.g. [PU13], [RR13]) these approaches are typically meant to guide the identification, the elicitation, and the prioritization of knowledge sources. However, only few approaches dealing with the explicit model-based documentation and structured analysis of these sources with regard to their dependencies and their relationships to the system under development have been suggested. Mostly, these approaches deal with specific aspects of context information. For example, [RR13] suggest the use of stakeholder maps to visualize and document the relations between stakeholders and the system under development. The approach suggests categories of stakeholders, e.g. users, business stakeholders, and impacting/interested stakeholders. Similarly, prior work has focused on the documentation of the relationship between the system under development and the interacting entities of its operational context, but has not focused on the nature of the interaction or on sources of context information (e.g. [DTW12]).

Aside from software engineering literature, the academic area of knowledge management deals with aspects of what can be considered the context of knowledge of embedded systems. Knowledge management literature covers a broad field: it ranges from more philosophical contemplations (e.g. [Ro07]) to rather technical applications such as data mining [Fa96]. Application of knowledge management approaches to software engineering typically assumes that companies acquire and store knowledge about development projects persistently such that it can be accessed at any time. However, these approaches typically underline the importance of content management systems (e.g. [SB97]), or the use of dedicated knowledge representation models (e.g. mind maps [Bu02]), but do not focus on documenting different types of knowledge with particular consideration of the respective development project.

4 Context of Knowledge: Foundations

As outlined in Section 2, documenting context information has a number of advantages for the engineering of embedded systems. In order to draw upon these benefits, it is necessary to document this context information, their respective knowledge sources, and the nature of their relationship to SUD. An ontology that structures different types of knowledge sources is shown in Fig. 1.

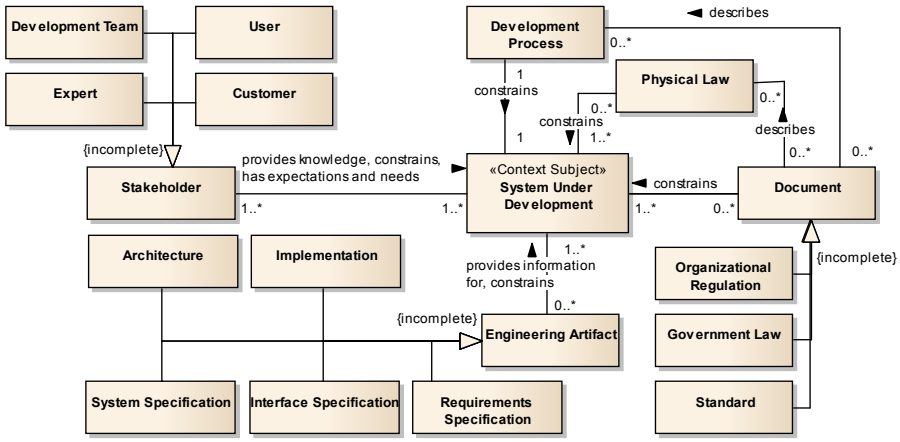

Fig. 1 Core Ontology of the Context of Knowledge

The relations between several kinds of knowledge sources and the SUD are depicted. Knowledge sources possess knowledge that is relevant to the system or the engineering thereof. Knowledge sources comprise documents (e.g. laws, organizational regulations, engineering standards, etc.), systems in operation (e.g. external systems interacting with the SUD), or stakeholders. Stakeholders may be customers, users, members of the development team (e.g. the requirements engineer, the tester, the architect, the business analyst, etc.), and other experts on the application domain that pertains to the SUD. In the example of the automatic window tinting system, relevant experts could be automotive engineers but also the driver. The SUD or its development may be constrained by knowledge that can be found in a number of documents such as government laws, organizational regulations or standards. In the window tinting example, relevant documents could be the aforementioned laws pertaining to road traffic regulations in the United States and Germany, respectively.

Beside knowledge sources pertaining to the SUD directly, knowledge sources pertaining to other systems in the context might pertain to the SUD as well. An SUD might interact with a number of systems that are part of its operational context. To ensure that the SUD can interact with those systems as intended, it is necessary to consider the engineering artifacts, such as the requirements specification, the system specification, the architecture, the interface specification, and the implementation, for each system in the SUD's operational context. Furthermore, when a SUD interacts with its context, knowledge

about a number of physical laws that apply to the SUD's context can have an influence on the SUD itself and are therefore a knowledge source. For example, this could be the laws of thermodynamics that must be considered during production of software components for a steel rolling mill. Furthermore, the development process imposes constraints on the SUD, because its quality affects the SUD's quality. Similarly to organizational regulations, the development process could be constrained, e.g. due to financial or to development time constraints, which may influence certain development decisions.

5 Diagrammatic Documentation of the Context of Knowledge

The context of knowledge mainly documents knowledge sources and their relationships between one another and their relationships to the SUD. Therefore, any general-purpose modeling language that provides symbols for modeling entities and their relationships, such as Entity Relationship Diagrams, UML Class Diagrams, or SysML Block Diagrams can be used to create context of knowledge models. However, since such modeling languages are often used to model properties of the SUD (e.g. the architecture or design), it can be difficult to distinguish between context of knowledge models and models of the SUD. Therefore, it may be advisable to use dedicated modeling elements for the ontological concepts of the context of knowledge. In Fig. 2, we show a number of dedicated modeling elements for the context of knowledge. These modeling elements can, for example, be integrated into a meta-model of an already existing modeling language. For example, they can be thought of as UML/SysML stereotypes inheriting from the meta-classes SysML::Block and MOF::Association, hence giving the context of knowledge model the semantic of a SysML Block Definition Diagram.

In addition, Fig. 2 shows how the modeling elements can be used to create a concrete context of knowledge model for an example system, i.e. a context of knowledge model for a collision avoidance system (CAS). A CAS prevents airplanes from colliding by warning the pilot about nearby traffic and issuing a resolution advisory when necessary. The CAS will monitor the airplane's environment and alert the pilot. In order for this interaction to occur, the CAS must provide the appropriate interfaces to its context. The information needed to develop these interfaces can be gained from knowledge sources that provide knowledge about the CAS's operational context, such as the engineering artifacts or its users and their relevant properties. Since safety is crucial in the avionics domain, the CAS's context of knowledge contains documents pertaining to safety aspects, such as regulations. Therefore the CAS's context of knowledge contains numerous knowledge sources that must not be neglected during the development of the CAS.

This context of knowledge model shows developers which knowledge source provides what kind of information (solid arrow) and how the development is constrained by knowledge sources (dashed arrow). It also supports the stakeholders in identifying missing knowledge sources. The CAS will interact with the Flight Control System (FCS). Therefore various engineering artifacts of the FCS provide the development team with relevant information about the functional behavior, the physical dimensions, the real-time behavior, needed usage behavior, and early information about needed interfaces. Beside this, laws and regulations constrain the CAS. For example ARP4761 [ARP96] is

an avionics standard that mandates FHA analysis to be conducted by a safety engineer as further stakeholder.

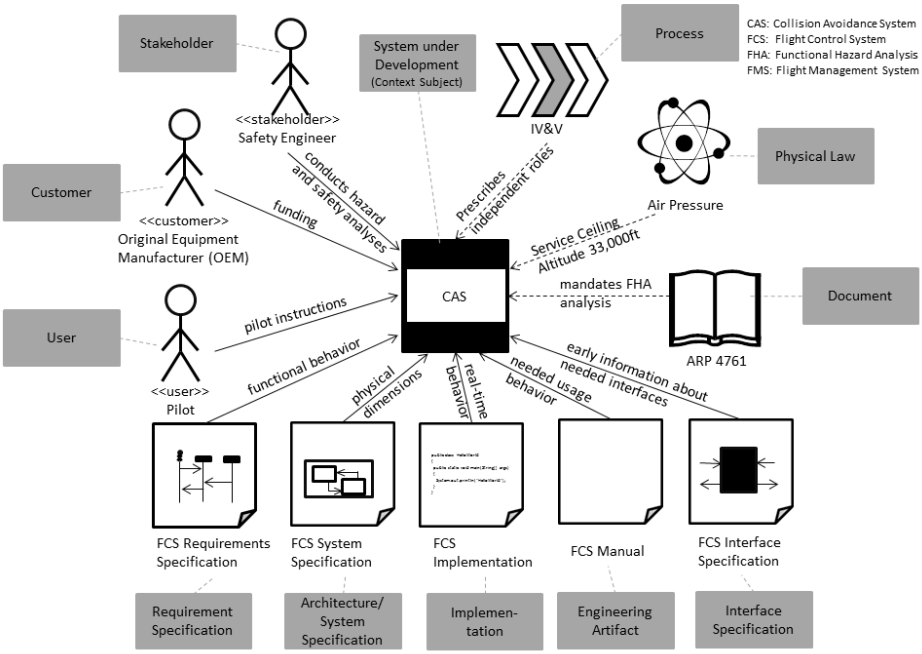

Fig. 2 Excerpt of the Context of Knowledge Model of a Collision Avoidance System

6 Abstracting the Context of Knowledge in a Landscape

One of the core advantages of model-based development is being able to create views onto a model with ease (cf. [Fi92]), particularly when models become too complex. Since the context of knowledge model is prone to accumulate a magnitude of associations between the SUD and the knowledge sources, it may be feasible to generate views which hide less relevant information while keeping the most important information for a particular purpose [Fi92]. In this section, we show two examples of such views. Fig. 3 shows a context of knowledge model for a CAS that includes the relevant knowledge sources from the context of knowledge for two other systems: the Flight Control System (FCS) and the Flight Management System (FMS).

Besides knowledge sources in the SUD's context of knowledge, knowledge sources in the context of knowledge of a related system can impact the SUD as well. In Fig. 3, this is the case because the CAS's ability to avoid collisions depends on how the FCS deals with the aircraft's own inertia. While strictly speaking, inertia is within the context of knowledge of the FCS, but it is also relevant for the CAS, albeit indirectly. In summary,

a context of knowledge model can be extended to include knowledge sources from the context of knowledge of systems in the SUD's context.

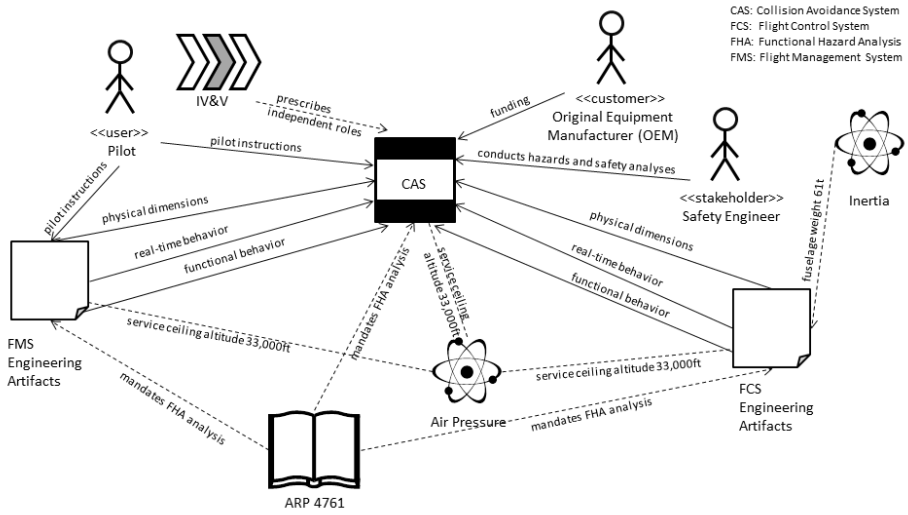

Fig. 3 Landscape of the Context of Knowledge depicting Relations between Context Entities

Like the CAS, the FMS and the FCS are constrained by air pressure and the ARP 4761 standard. The pilot can be found in the CAS's and in the FMS's context of knowledge but not in the FCS's. The only system directly affected by inertia is the FCS, but since the CAS interacts with the FCS, inertia impacts the CAS indirectly. The documentation of these knowledge sources relevant to systems in the context has several benefits. Changes of the ARP 4761 standard, for example, can be traced to needed changes in the FMS, these changes might also affect the CAS. Documenting that not only the subject but further objects are affected by the information contained in ARP 4761 allows for propagating possible changes to all systems in close coordination.

Fig. 4 shows a context of knowledge landscape for a CAS with particular regard to the FCS. Here the FCS is represented by three specific engineering artifacts: the implementation, the system specification and the requirements specification. Inertia, air pressure, and the ARP 4761 standard constrain all three engineering artifacts. This kind of knowledge landscape provides a more detailed view on the relationship between the SUD and a single system in the context. Representing a system in the context by its various engineering artifacts is especially useful if the system in the context is being developed at the same time because the system in the context and its engineering artifacts are likely to change frequently during this period.

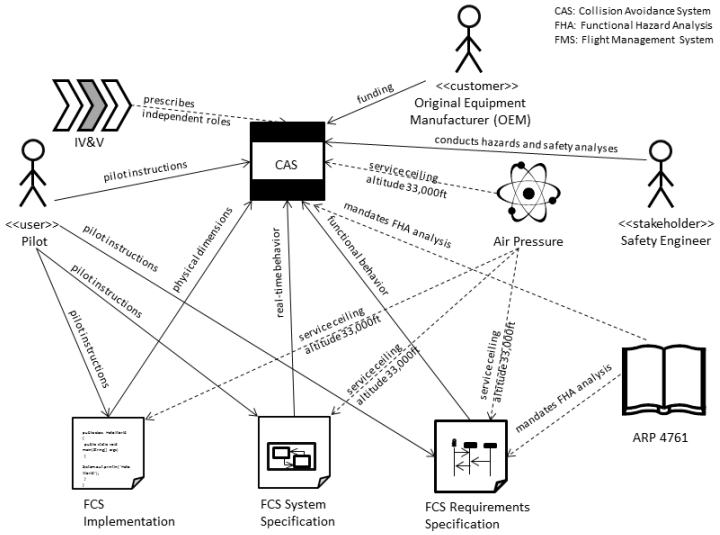

Fig. 4 Context of Knowledge Focusing on Engineering Artifacts

7 Conclusion

In this paper, we have made an argument for the necessity to document the context of knowledge of embedded systems. The context of knowledge comprises knowledge sources that provide information that constrains the system or its development. Documenting knowledge sources, their relationships to one another, and their relationships with the system under development is necessary to be able to clearly define the scope of development and is an important prerequisite for the definition of valid requirements. By making knowledge sources comprehensively persistent, the decision making process can be supported, and conflicts can be resolved. Furthermore, using context information, the risk of inconsistencies of subsystems that are developed in parallel on multiple layers of abstraction can be substantially reduced, as each subsystem can be understood as being part of the context of each other subsystem. Thereby, other subsystems that interact with one particular subsystem can be identified, documented, and monitored. We have outlined an approach to document the context of knowledge by means of graphical models and substantiated our approach ontologically. Furthermore, we have shown the application of our approach by means of a simplified case example from the avionics domain and showed how views onto the context of knowledge can be generated. In future work, our approach could be combined with specific techniques from the area of knowledge management to apply, e.g. information mining to embedded systems development.

Acknowledgements

This paper was funded in part by the *German Federal Ministry of Education and Research* under grant number 01IS12005C.

References

[AH01] Alfaro, L.; Henzinger, T.: Interface automata. In: Proc. of the FSE, 2001; pp. 109–120.

[ARP96] SAE: ARP4761 – Guidelines and Methods for Conducting The Safety Assessment Process on Civil Airborne Systems and Equipment, 1996.

[BB12] Beetz, K.; Böhm, W.: Challenges in Engineering for Software-Intensive Embedded Systems. In: Model-Based Engineering of Embedded Systems. Springer, 2012; pp. 3–14.

[BC06] Bergh, J.; Coninx, K.: CUP 2.0: High-Level Modeling of Context-Sensitive Interactive Applications. In: Proc. of the MoDELS, 2006; pp. 140–154.

[Bo81] Boehm, B.: Software Engineering Economics. Prentice Hall, 1981.

[Bu02] Buzan, T.: How to mind map. Thorsons, London, 2002.

[Da93] Davis, A.: Software requirements. Objects, functions, and states. Prentice Hall,, 1993.

[Dh09] Dhaussy, P. et al.: Evaluating Context Descriptions and Property Definition Patterns for Software Formal Validation. Proc. of the MoDELS, 2009; pp. 438–452.

[DTW12] Daun, M.; Tenbergen, B.; Weyer, T.: Requirements Viewpoint. In: Model-Based Engineering of Embedded Systems. Springer Berlin Heidelberg, 2012; pp. 51–68.

[Fa96] Fayyad, U.; Piatetsky-Shapiro, G.; Smyth, P.; Uthurusamy, R.: Advances in Knowledge Discovery and Data Mining. The MIT Press, 1996.

[Fi92] Finkelstein, A.; Kramer, J.; Finkelstein, L.; Goedicke, M.: Viewpoints: A Framework for Integrating Multiple Perspectives in System Development. In: Int. J Soft. Eng. Knowl. Eng. 1992, 2, pp. 31-59.

[Georg] State of Georgia: Motor Vehicles and Traffic - Horns, Exhaust Systems, Mirrors, Windshields, Tires, Safety Belts, Energy Absorption Systems. Official Code of Georgia: Title 40, Ch. 8, Art. 1, Part 4, 2013.

[HRH01] Hammond, J.; Rawlings, R.; Hall, A.: Will it work? In: Proc. IEEE Intl. Symp RE, 2001.

[ISO10] ISO/IEC: ISO/IEC 25010 - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models, 2010.

[Ja95] Jackson, M.: The World and the Machine: Proc. of the ICSE, 1995; pp. 283–292.

[KW93] Ketler, K.; Walstrom, J.: The outsourcing decision. In: Intl. J of Inf. Mgmt., 1993, 13; pp. 449–459.

[LKK04] Lehtola, L.; Kauppinen, M.; Kujala, S.: Requirements Prioritization Challenges in Practice. In: Product Focused Software Process Improvement. Springer, 2004; pp. 497–508.

[MP84] McMenamin, S.; Palmer, J.: Essential systems analysis. Yourdon, New York, 1984.

[Po10] Pohl, K.: Requirements Engineering. Fundamentals, Principles, and Techniques. Springer, 2010.

[PU13] Pohl, K.; Ulfat-Bunyadi, N.: The Three Dimensions of Requirements Engineering: 20 Years Later. In: Seminal Contributions to Information Systems Engineering. Springer, 2013; pp. 81–87.

[Ro07] Rowley, J.: The Wisdom Hierarchy: Representations of the DIKW Hierarchy. In: J of Inf. Sc. 2007, 33, pp. 163-180.

[RR13] Robertson, S.; Robertson, J.: Mastering the Requirements Process. Getting Requirements Right. Addison-Wesley, 2013.

[SB97] Sawy, A.; Bowles, G.: Redesigning the Customer Support Process for the Electronic Economy: Insights from Storage Dimensions. In: MIS Quarterly, 1997, 21; pp. 457-483.

[SDP12] Sangiovanni-Vincentelli, A.; Damm, W.; Passerone, R.: Taming Dr. Frankenstein: Contract-Based Design for Cyber-Physical Systems. European J of Control 18(3), 2012.

[St12] The Standish Group: Chaos Manifesto 2012.

[StVZO] Federal Republic of Germany: Scheiben, Scheibenwischer, Scheibenwascher, Entfrostungs- und Trocknungsanlagen für Scheiben. StVZO §40 Absatz 1, 2013.

[We10] Weyer, T.: Kohärenzprüfung von Verhaltensspezifikationen gegen spezifische Eigenschaften des operationellen Kontexts. Dissertation, University of Duisburg-Essen, Essen, 2010.