A Structured Approach to Support Collaborative Design, Specification and Documentation of Communication Protocols

Fabian Ohler[1], Markus C. Beutel[1,2], Sevket Gökay[1,2], Christian Samsel[1,2] and Karl-Heinz Krempels[1,2]

[1]*Fraunhofer Institute for Applied Information Technology FIT, Aachen, Germany*

[2]*Information Systems, RWTH Aachen University, Aachen, Germany*

Keywords: Collaboration, Documentation, Protocols, Requirements Engineering, Specification, Tool Support.

Abstract: Especially in complex software development projects, involving various actors and communication interdependencies, the design of communication protocols is crucially important. In this work, a structured approach to support the design, specification and documentation of communication protocol standards is presented. To do so, we refer to a complex use case, dealing with the integration of multiple mobility services on a single platform. This endeavor requires the development of a large number of independently usable protocol standards which adhere to a multitude of quality aspects. A structured approach is required to speed up and simplify development and also to enable synergies between these protocols. Our requirements analysis methodology consists of interviewing domain experts to identify important aspects and shortcomings of the current development process and to elicit potential improvements. These intermediate results are prioritized and incorporated into a requirements specification for a standardized communication protocol development process. Furthermore, we assess existing software solutions in terms of their applicability.

1 INTRODUCTION

An intermodal personal mobility platform allows travelers to query, book, use and pay any combination of mobility services (Beutel et al., 2016). This might include, but is not limited to public transportation, vehicle rental services, vehicles sharing services, parking, charging (of electric vehicles), ride sharing, etc. To simplify taking part on such a platform and foster innovation, open and standardized access is required (Beutel et al., 2014). This can be achieved by supplying formal and comprehensive design documents, specifications and documentation for the respective communication protocols and application program interfaces (APIs). As a large number of such artifacts is required, we ought to simplify and speed up the creation process by providing integrated support to all involved stakeholders. As a first step, interviews with domain experts to understand the current process for developing communication protocols, e.g., which phases it is composed of, which tools are used and so on, have been conducted. Furthermore, we try to identify both benefits and shortcomings to elicit potential areas of improvement.

The rest of this work is organized as follows: The requirements analysis, including methodology, results and analysis is presented in Section 2. Moreover, Section 3 comprises a survey of current support tools. Related work is listed in Section 4. In Section 5, we critically discuss the experiences so far and give an outlook of the next steps.

2 REQUIREMENTS ANALYSIS

In this section, we present our approach to identify issues in the development process of communication protocols and the respective results.

2.1 Methodology

To better understand the workflow of protocol development projects and identify the problems that arose in this context in the past, the first step of the analysis phase comprised interviewing domain experts. Instead of providing relatively strict response categories in a classical questionnaire, during an interview, experts are able express their ideas and answers freely using their own professional terminology (Mieg and

Ohler, F., Beutel, M., Gökay, S., Samsel, C. and Krempels, K.
A Structured Approach to Support Collaborative Design, Specification and Documentation of Communication Protocols.
DOI: 10.5220/0006787503670375
In *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2018)*, pages 367-375
ISBN: 978-989-758-300-1
Copyright © 2019 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

Näf, 2005). We used the semi-structured approach[1] (Fylan, 2005) to be able to delve into topics of interest, to find out more about the *why* on the one hand and to be able to compare the results between the experts on the other hand. Before the actual interviews, we prepared an interview guide and evaluated it in a pre-test with a senior software engineer with a couple of years of experience in the field of interest. The structure of the interviews contained a general introduction, collecting demographic data and the participants' experience as a starting point. To learn more about the general processes and approaches employed, the experts were asked to identify development phases in past projects. For each of the potentially mentioned phases, general questions regarding activities and their corresponding outcomes were prepared. Furthermore, the supporting software tools utilized for specification, documentation and communication were investigated. In particular, the experts were to identify the weaknesses and deficiencies of the tools employed. An additional goal of the interviews was to determine how the results during the different phases were formalized. We also asked about the responsibility assignment considering tasks and roles during the protocol specification process. Since we expected the experts to identify phases akin to the phases in a classic software development process, the interview guide contained further phase-specific questions to be asked in case the expert identified that particular phase.

Our interview aimed at being systematizing[2] (in contrast to exploratory or theory-generating interviews, cf. (Bogner and Menz, 2009)). We were able to interview five experts, of which we consider four to be elite according to the remarks in (Littig, 2009), where the author distinguishes between the elite, characterized by formative power, and the experts, characterized by interpretative power, with both groups overlapping strongly. According to other research, this quantity seems acceptable for qualitative studies (van Heek et al., 2017; Karlsson et al., 2002; Neufeldt et al., 1996). Since the experts were spread all over Germany, we performed the interviews mostly by phone and tape-recorded all of them. The duration of the interviews varied between about 35 to 75 minutes.

An overview over the participants is presented in Table 1. All participants have experience in protocol specification and standardization. Participants B and C also have taken part in international standardization bodies for communication protocols.

In a second step, the unified results of the interviews where presented to a larger group of experts ($n = 15$) for a focused discussion. After the presentation of the results, the group was asked to order the identified issues, based on their severity. While not being fully validated, this prioritization of most pressing issues guides the decision process for the further development and forms the assessment metrics for the survey of existing tools (Section 3).

2.2 Qualitative Results

In this section, the results of the aforementioned interviews including the phases distinguished by the experts and the corresponding tasks and artifacts are presented. Furthermore, an overview over the issues experienced and finally derived requirements for adequate tool support is given.

Table 3 at the end of this work lists a selection of key statements by the interviewees that we reference using S01…S13.

Development Phases

The experts identified the following tasks and phases, which resemble a consolidation of steps mentioned in (Nuseibeh and Easterbrook, 2000):

1. **Analysis:** In this phase, a catalog of the roles, personas, scenarios, use cases, and a glossary is created. The results are filed in the form of text and tables.
2. **Architecture design:** In this phase, sequence, component, and deployment diagrams are created. The results are (usually) filed in the form of text and UML (Unified Modeling Language) documents.
3. **Protocol design:** In this phase, data types, interfaces and data flows are specified. The results are (usually) filed in the form of text and XML (Extensible Markup Language) documents.
4. **Validation:** In this phase, reviews, functional and integration tests in a sandbox environment and demonstrations are conducted.

Issues Identified

For these phases, the experts identified the following problems (the solution approaches additionally speci-

[1] "Semi-structured interviews are simply conversations in which you know what you want to find out about – and so have a set of questions to ask and a good idea of what topics will be covered – but the conversation is free to vary, and is likely to change substantially between participants." (Fylan, 2005)

[2] "This kind of expert interview is an attempt to obtain systematic and complete information. The expert enlightens the researcher on objective matters." (Bogner and Menz, 2009)

Table 1: Interview participant overview.

Interviewee	Age	Sex	Position	Area	Experience
A	62	f	Manager	Electronic Ticketing	39y
B	46	m	Division head	Real-time Passenger Information	15y
C	55	m	Department head	Public Transport Information Systems	15y
D	32	f	Research fellow	Media Production	4y
E	45	m	Department head	Public Transport Information Systems	22y

fied were partially already mentioned by the experts during the interviews):

Analysis. The distributed nature of the collaborative work on documents caused problems, which were particularly mentioned by interviewees A and D, cf. S01 to S05 in Table 3. The exchange of documents (mostly via e-mail) was cumbersome and often led to conflicts while merging the changes. These problems can be mitigated by using a central document storage and management. They could further be solved by enabling to collaboratively work on documents. Additionally, the experts found a weak degree of formalization of the results to be problematic, especially interviewees B and D, cf. S09. This could be improved by employing suitable tools and well-established templates for use cases, personas, etc.

Architecture and protocol design. The separated documentation in the form of text and UML / XML documents using different tools led to duplicate maintenance, thus to additional effort and frequently to inconsistencies (stressed by interviewees B and C, cf. S07). By integrating the formal specification and the corresponding documentation in a single tool, editing can be eased significantly (this was already employed in projects of E, but changes made were hard to merge after parallel work, cf. S04).

Validation. The poor tool support for testing and validation was identified to be the main problem in this phase. An automated mechanism to test implementations for specification conformity was lacking. Using the sequence diagrams specified in the previous phase, test templates could be generated. In addition, the diagrams could be augmented in such a way, that actual test cases could be generated from them. Interviewees C and E found this to be an important aspect. Generally speaking, the earlier phases should be adjusted such that the artifacts created there are furnished with as much additional information as possible to be able to use these for testing and validation. Furthermore, the experts noted, that a certification (in terms of a confirmation of a correct protocol implementation), if offered at all, was perceived as very expensive and that there were no options for a self-certification. By means of tools for conformity checks, such a self-certification might be possible.

Across the phases, the following additional problems were addressed:

- The deficient documentation of decisions lead to insufficient traceability of these in the course of the projects (S13). By intensifying the documentation using suitable methods such as issue trackers this can be counteracted.
- The documents edited were versioned often only by convention. By integrating the version control into the tools, suitable software support could be realized.
- A history of the documents providing transparency over which changes were made by whom was mostly non-existent (interviewee A). Integrated version control in the tools could also help here.
- The fact that the working groups mostly documented their results in German whereas the implementation is typically in English again led to duplicate maintenance and the corresponding problems. A multilingual documentation thus seems worthwhile.

Furthermore, interviewee C (cf. Table 1) remarked that cloud-based tools often could not be used due to regulations and only solutions under the complete control of projects partners could be used. He would also like the tool support to be easy to use (without naming tangible metrics such as training time), even for occasional users (S10). Interviewee B emphasized the aspect of reviews in the last development phase, especially in standardization projects. Additionally, E mentioned that, during protocol design, it would be nice to provide for a way to test a server running an implementation w.r.t. whether it is fully functional (i.e. all necessary functional components are ready and working) and not only whether it responds to ping.

Issue Prioritization

Since the described problems entail an extremely high amount of work to address them, they were prioritized with respect to the protocol development process together with experts after the results of the interviews were processed. This prioritization was used by us to plan the next steps regarding how to improve an upcoming development process. Not all of the interviewed experts were present, but other software engineers with experience in the field attended leading to an even higher participation. This different set of participants may explain the low rating of the testing and validation support below. Here, priority values from P1 (very low) to P5 (very high) were used:

P5 – Separated documentation in form of text, UML and XSD (XML Schema Definition) documents

P4 – Lacking central availability / management of documents
 – Problems while merging documents after parallel editing
 – Absence of glossary (including early agreement on definitions between project partners, avoiding to elongate the usage of imprecise terms)

P3 – Lack of decision documentation / traceability

P2 – Lack of templates for use cases, personas, etc.
 – Lack of tools for testing

P1 – Lack of conformity checks (self-certification)

The problems contemplated, especially those with high priority, stem from a low support for collaboration in the different tools employed and the lack of integration between the different tools. We think, that the best way to tackle these problems is to start with a tool supporting collaborative work and to integrate as many aspects of the protocol development process as possible.

Derived Requirements

We derived the following mainly functional requirements for the protocol development tool support from the interviews and the problems discussed above (with order not implying priority):

General requirements regarding the development process workflow:

R1.1 Integrated tool support: The fact that information was more or less unrelated and scattered over different tools was seen as an important problem (see S07 and S08 in Table 3 and the top priority problem above)

R1.2 Cooperative and parallel editing: To mitigate problems arising from parallel work, e.g., merge conflicts (see S01 to S04 and the first two problems with priority 4).

R1.3 Displaying active users with their work focus: Best practice in tools supporting cooperative editing (see S03).

R1.4 A history of changes: Being able to see what changed between versions was perceived as important, see 3rd additionally identified problem.

R1.5 Decision documentation refers to documenting how decisions, e.g., for a specific technology, have been made throughout the development. Based on S13, we assume that not only the outcome of a decision is to be documented, but also the initial problem and potential alternatives.

R1.6 Commenting the elements: Important in review phase and to discuss issues, see S08.

R1.7 Creating snapshots for setting milestones or releases: Carrying over the versioning concepts to the project scope, cf. second additional problem.

R1.8 Tool support should be easy to learn (within a few hours): Statement S10 emphasizes that tool support should also be easy to learn and use for occasional users. This has to be refined to a metric usable for validation.

Requirements regarding the documentation scope:

R2.1 Multilingualism: As mentioned in S12 as well as via the 4th additional problem, protocols often have to be documented in more than one language (e.g. VDV 301-2-13[3]).

R2.2 Templates, styles, etc. for personas, scenarios, requirements, etc.: Integrating well established templates for the analysis phase artifacts may help mitigate the problems mentioned in S09 and the first problem with priority 2.

R2.3 Glossary for the domain terms used (including a blacklist): Maintaining a glossary for the relevant domain terms is critical for unambiguous communication (cf. third problem with priority 4, S05 and S06).

R2.4 Glossary for the concepts used (e.g. use case, role, persona): Maintaining a glossary for the relevant concepts helps clarify tasks and results (cf. third problem with priority 4, S05 and S06).

Requirements regarding the documentation functionality:

[3]https://www.vdv.de/301-2-13-sds.PDFx

R3.1 Import functionality for resources such as figures, logos, etc.: This requirement was derived from outcomes of previous standardization projects.

R3.2 Rich text editing including enumerations, tables, comments, and figures: This requirement was derived from outcomes of previous standardization projects.

R3.3 UML diagrams (at least a subset including use case and sequence diagrams): This requirement was mentioned as an artifact.

R3.4 An export functionality of the documentation in Microsoft Word format: This requirement was mentioned as an artifact and as a prerequisite for standardization bodies, cf. S07.

Technical requirements regarding the implementation workflow:

R4.1 XML / XSD to specify data types: This requirement was mentioned as an artifact.

R4.2 Importing existing schema definitions (e.g., from previous projects or in case a project consists of sub-projects): This requirement was derived from outcomes of previous standardization projects.

3 EVALUATION OF EXISTING TOOLS AND APPROACHES

By means of the requirements derived above, we evaluated several existing open source tools (such as Papyrus[4] and UML Designer[5], which are based on Eclipse Modeling Framework) and commercial tools (in particular those tools that came up in the expert interviews: Microsoft Word[6], Sparx Systems Enterprise Architect[7], Altova XMLSpy[8]). An overview over these products is given in Table 2.

Across all of them, the following commonalities can be observed:

- All of them are desktop applications.
- There is no support for multilingual documentation.
- The user experience could be improved as using the tools is non-intuitive or the usability suffers from the great functional extent.
- Collaborative editing of documents is only possible using additional systems or support for cooperative development is very limited.
- The functional coverage with respect to testing and validation is insufficient.

Additionally, the tools with the highest requirements coverage are commercial products and can thus not be modified or extended beyond their current features set. On the other hand, the requirements coverage of the open source tools is relatively low – especially in the context of the high priority of cooperative editing of documents, which is tackled by none of the tools in a sufficient way. While Microsoft Word has a high degree of collaborative support, the near-real time collaboration relies on a proprietary cloud-based solution, which is often in conflict with (company) regulations and thus cannot be taken advantage of.

4 RELATED WORK

Most software engineering projects include highly collaborative tasks during all project phases from requirements analysis to validation. Therefore, tools supporting such projects should facilitate collaborative work in an adequate way, i.a. through the use of social media (Storey et al., 2010). In this paper, we focused on framework, library and protocol development, where the target audience needs to be supplied with a formal specification of the API / protocol and a documentation, e.g., of the data types, the interrelationships, and the big picture. Thus, maintaining the documentation in accordance with changes in the specification is of particular importance and poses additional challenges (Dagenais and Robillard, 2010).

Yet, over the years, collaboration found its way into many aspects of professional life. Therefore, general purpose collaborative tool support (often referred to as groupware) such as BSCW (Bentley et al., 1997), supporting, e.g., file exchange, version control and project management tasks, has been around for some time now.

With special focus on collaborative software development, various tools emerged over time supporting different aspects of the development process. Examples include version control systems, issue trackers, or collaborative development environments (such as GitHub[9]). An overview over the broad spectrum in this area is given by (Lanubile et al., 2010). Complementing these tools with a new approach to communication are *team communication platforms* (such

[4]https://www.eclipse.org/papyrus
[5]https://www.umldesigner.org
[6]https://www.office.com
[7]https://www.sparxsystems.com/products/ea
[8]https://www.altova.com/xmlspy-xml-editor

[9]https://github.com

Table 2: Overview over the tools and their support for specific aspects of protocol development projects.

	Papyrus	UML Designer	Microsoft Word	Sparx Systems Enterprise Architect	Altova XMLSpy	Altova UModel
1. Analysis						
Personas, scenarios, use cases	yes	yes	no	yes	no	yes
Requirements	yes	no	no	yes	no	yes
Glossary	no	no	no	yes	no	no
2. Architecture Design						
Architecture, component, deployment diagrams	yes	yes	no	yes	no	yes
Sequence diagrams	yes	yes	no	yes	no	yes
3. Protocol Design						
Function and protocol definition	yes	yes	no	yes	yes	yes
Data type definition	yes	yes	no	yes	yes	yes
4. Validation						
Generating documentation	no	no	no	yes	yes	yes
Reviews	no	no	yes	yes	no	no
Functional and integration testing	yes	no	no	yes	no	no
Across the phases						
Decision documentation	no	no	no	no	no	no
Multilingual documentation	no	no	no	no	no	no
Cooperative work support						
File format compatible with external version control	yes	yes	no	no	no	no
Conflict resolution and merging support	no	no	yes	yes	yes	yes
Traceability of changes	no	no	yes	yes	yes	yes
Near-real time collaboration	no	no	yes	no	no	no

as Slack[10]). The potential benefits and use cases of platforms like these are evaluated in (Anders, 2016).

Moreover, there are scientific approaches with a more top level focus on tool support for collaborative design. Examples here include the approach presented in (Thomas and Scheer, 2006) for the area of computer-aided management of reference models in business engineering projects or for the collaborative construction of ontologies (Farquhar et al., 1997). Currently, works in the field of cooperative tool support take that concept to the next level incorporating support for shared near-real time editing not only of text, but of models. The *Community Application Editor* presented in (de Lange et al., 2017) supports model-driven web engineering in a collaborative manner. In (Nicolaescu et al., 2016), users are presented as being able to collaboratively design a metamodel used to generate collaboration-enabled editors for model instances.

[10]https://slack.com

5 DISCUSSION

This work describes initial and fundamental steps on a path towards a structured approach for collaborative design, specification and documentation of protocol standards. By the help of systematized expert interviews, we characterized the protocol development process. Moreover, core requirements for a software tool, that supports such collaborative processes, were identified and existing solutions in the area of interest were analyzed.

According to the importance of suitable protocol specifications in software development, a structured collaborative approach, consisting of a process as well as tool support, seems valuable in various areas. An approach that considers the requirements presented beforehand, might foster efficient and high-quality protocol design and specification, not only in the area of personal mobility. In particular, the collaborative design aspect plays a major role in, but not limited to, complex development projects and offers interesting potentials. Considering information interdependencies and exchanges among system boundaries, collaboration support seems very fruitful. In the same step with fostering collaboration, progress and results become more transparent to participants. This creates comprehensibility that might be beneficial or even adversely in specific cases.

Some of the above identified requirements seem demanding towards a tool implementation. Hence, there is a need to strike a balance between adequate functionality and contemporary realization. Given the fact, that there are also individual best practices and tools, the willingness to use a collaborative tool across organization boundaries might be an obstacle in the future. Another drawback might be, that the sustainability of newly developed tool support is unknown and potentially inferior to established commercial products.

While the focus of this paper is on the development processes for communication protocols, many aspect might also be valid for general software development projects. Even though many organizations already use tools like GitHub and Google Docs[11], there are still a lot of companies that do not rely on services hosted by third parties for various reasons including data privacy concerns, sustainability doubts, or to avoid being dependent on third party decisions. Self-hosted solutions are used to fill some of the blanks, but are then often only available for internal use and can not be employed in inter-organizational endeavors such as the protocol development projects considered here for security and administrative reasons. We are thus looking forward to enable the standardization bodies to provide projects aiming at submitting drafts with a collaborative development platform by making the tool envisioned by us available to them.

Current Work and Outlook

Based on our findings, we concluded that currently available tools only inadequately support our recommended workflow and its phases. According to our set requirements and prioritization, we started developing a dedicated tool set to support the design, specification and documentation of communication protocols. Special attention is given to improve inter-organizational collaboration, as this requirement promises a large improvement over current solutions. We surveyed current existing libraries and software components for the cooperative editing of documents. Most of these are based on web technologies (XHTML, CSS, JavaScript), which lead us to decide that a web-based tool is the best option. Furthermore, existing JavaScript libraries were examined w.r.t. requirements coverage, which could be integrated into the targeted tool support, in the areas of UML, XML and rich text editors. These existing libraries and software components were evaluated w.r.t. their compatibility with the shared near real-time editing approach. In doing so, we identified a set of libraries as starting point for the actual development of our tool.

As soon as the to-be-developed tool reaches a usable state, we aim to apply it in our project to improve the development process. Based on potential feedback of future users, we will iteratively change and evaluate both the development process as well as the supporting tool. We will apply both qualitative, e.g. more interviews, as well as quantitative methods, e.g. a usability study, to do so.

[11]https://www.google.de/intl/en/docs/about/

ACKNOWLEDGEMENTS

This work was partially funded by the German Federal Ministry of Transport and Digital Infrastructure (BMVI) for the project "Digitalisierte Mobilität – die Offene Mobilitätsplattform" (19E16007B). The authors want to thank Felix Schwinger and Andreas Heuvels for their support in conducting the interviews.

REFERENCES

Anders, A. (2016). Team communication platforms and emergent social collaboration practices. *International Journal of Business Communication*, 53(2):224–261.

Bentley, R., Horstmann, T. C., and Trevor, J. (1997). The world wide web as enabling technology for CSCW: the case of BSCW. *Computer Supported Cooperative Work*, 6(2/3):111–134.

Beutel, M. C., Gökay, S., Kluth, W., Krempels, K., Samsel, C., and Terwelp, C. (2014). Product oriented integration of heterogeneous mobility services. In *17th International IEEE Conference on Intelligent Transportation Systems, ITSC 2014, Qingdao, China, October 8-11, 2014*, pages 1529–1534.

Beutel, M. C., Gökay, S., Kluth, W., Krempels, K.-H., Ohler, F., Samsel, C., Terwelp, C., and Wiederhold, M. (2016). Information integration for advanced travel information systems. *Journal of Traffic and Transportation Engineering*, 4(4):177–185.

Bogner, A. and Menz, W. (2009). The Theory-Generating Expert Interview: Epistemological Interest, Forms of

Knowledge, Interaction. In Bogner, A., Littig, B., and Menz, W., editors, *Interviewing Experts*, chapter 2, pages 43–80. Palgrave Macmillan.

Dagenais, B. and Robillard, M. P. (2010). Creating and evolving developer documentation. In *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering - FSE '10*. ACM Press.

de Lange, P., Nicolaescu, P., Klamma, R., and Jarke, M. (2017). Engineering web applications using real-time collaborative modeling. In Gutwin, C., Ochoa, S. F., Vassileva, J., and Inoue, T., editors, *Collaboration and Technology - 23rd International Conference, CRIWG 2017, Saskatoon, SK, Canada, August 9-11, 2017, Proceedings*, volume 10391 of *Lecture Notes in Computer Science*, pages 213–228. Springer.

Farquhar, A., Fikes, R., and Rice, J. (1997). The ontolingua server: a tool for collaborative ontology construction. *International Journal of Human-Computer Studies*, 46(6):707 – 727.

Fylan, F. (2005). Semi-structured interviewing. In Miles, J. and Gilbert, P., editors, *A Handbook of Research Methods for Clinical and Health Psychology*, chapter 6, pages 65–78. Oxford University Press Oxford.

Karlsson, L., Dahlstedt, Å. G., Dag, J. N. O., Regnell, B., and Persson, A. (2002). Challenges in market-driven requirements engineering - an industrial interview study. In *Proceedings of the Eighth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'02)*.

Lanubile, F., Ebert, C., Prikladnicki, R., and Vizcaíno, A. (2010). Collaboration tools for global software engineering. *IEEE Software*, 27(2):52–55.

Littig, B. (2009). Interviewing the Elite Interviewing Experts: Is There a Difference? In Bogner, A., Littig, B., and Menz, W., editors, *Interviewing Experts*, chapter 4, pages 98–113. Palgrave Macmillan.

Mieg, H. A. and Näf, M. (2005). *Experteninterviews (2.Aufl.)*. Institut für Mensch-Umwelt-Systeme (HES), ETH Zürich.

Neufeldt, S. A., Karno, M. P., and Nelson, M. L. (1996). A qualitative study of experts' conceptualizations of supervisee reflectivity. *Journal of Counseling Psychology*, 43(1):3–9.

Nicolaescu, P., Rosenstengel, M., Derntl, M., Klamma, R., and Jarke, M. (2016). View-based near real-time collaborative modeling for information systems engineering. In *Advanced Information Systems Engineering - 28th International Conference, CAiSE 2016, Ljubljana, Slovenia, June 13-17, 2016. Proceedings*, pages 3–17.

Nuseibeh, B. and Easterbrook, S. (2000). Requirements engineering: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 35–46. ACM.

Storey, M.-A., Treude, C., van Deursen, A., and Cheng, L.-T. (2010). The impact of social media on software engineering practices and tools. In *Proceedings of the FSE/SDP workshop on Future of software engineering research - FoSER '10*. ACM Press.

Thomas, O. and Scheer, A.-W. (2006). Tool support for the collaborative design of reference models - a business engineering perspective. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences 2006 (HICSS'06)*.

van Heek, J., Arning, K., and Ziefle, M. (2017). Differences between laypersons and experts in perceptions and acceptance of CO2-utilization for plastics production. *Energy Procedia*, 114:7212–7223.

Table 3: A selection of key statements made by the interviewees.

No.	Interviewee	Statement
S01	D	"I consider cooperative activities as the greatest shortcoming of both tools we used."
S02	D	"In our case [in the context of parallel work], there were a lot of single documents, which needed to be merged in a relatively time-consuming way."
S03	D	"It would make sense to have an exchange platform which everyone can consistently access at all times and where you can work on documents together, to transparently give everyone insight into the current activities, to give everyone the possibility to inspect the results and to enable parallel work on the documents."
S04	E	"When several people worked collaboratively on the same, common ontology[a], we had no way of merging that."
S05	E	"We found out that we repeatedly interviewed [domain] experts, where we used the same wording, but meant different things. We write down, what we understood. As soon as this goes back to him [the expert], he sees and clarifies it. [...] [cooperative tool support] would allow us to invite several domain experts to work on a record [of an interview]."
S06	B	"Even when you only leave your own system vendor and talk to other partners, you often use a different terminology. As soon as you start using a foreign language, it becomes hard for the partners involved to be precise in the terminology such that everyone understands exactly what others mean."
S07	B	"Currently, there is a duplicate maintenance in the tools, e.g., in Enterprise Architect and in the Word documentation. If you would manage to do both with one tool, the documentation as well as the actual specification activities [...] that would be a goal."
S08	D	"One could definitely improve the structured reviews of the architecture design since the links between the architecture and the requirements were rather scattered. For there were two completely separated – let's say – worlds in terms of documentation, I personally saw this as a major obstacle that needed to be overcome."
S09	B	"A more consistent format would definitely be desirable. Such a more formal approach would generally help to guarantee portability of the results, but also to ensure that all aspects of a use case were considered and nothing was forgotten."
S10	C	"Another problem here is not just maintenance of the tools, but also keeping the know-how. At least for us that was the case, as when you work with it very rarely, the usage it is rather hard."
S11	A	"[The tools] still produce documents one has to rework; which don't have the form one imagined. At this point there is still a lot to do, I think."
S12	C	"Keep in mind that various documentations need to be bilingual."
S13	A	"Many things were discussed that were discarded over the time. One should have documented this even more."

[a] a set of concepts and categories in a subject area or domain that shows their properties and the relations between them.