Best Practice-based Evaluation of Software Engineering Tool Support: Collaborative Tool Support for Design, Data Modeling, Specification, and Automated Testing of Service Interfaces

Fabian Ohler[1] 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: Software Engineering Collaboration, Service Interface Specification, Service Interface Interaction Design, Tool Support Evaluation, Software Engineering Tool Support.

Abstract: Especially in complex software development projects, involving various actors and interaction interdependencies, the design of service interfaces is crucially important. In this work, a structured approach to support the design, specification and documentation of service interface 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 service interface 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 service interfaces. In a previous work, we performed a requirements analysis to identify important aspects and shortcomings of the current development process and to elicit potential improvements. Starting with a first implementation of collaborative tool support for service interface development, we conducted a best practice-based evaluation with experts of the Association of German Transport Companies (VDV). In this paper, we want to present the results of this focus group-based evaluation and discuss their implications for the envisaged tool support for collaborative service interface development (design, data modeling, specification, and automated testing).

1 INTRODUCTION

Software engineering projects involve highly cooperative tasks during every project phase (Kusumasari et al., 2011; Inayat et al., 2017). Over the years, a lot of approaches have surfaced to support these tasks in different ways ranging from traceability over awareness to assistance during merges of parallel work (Whitehead, 2007). A general trend can be observed, pushing a lot of applications towards support for cooperation (Dalmau et al., 2000). This implicates additional conceptual challenges to guarantee that user expectations regarding usability of the software are met (Garrido et al., 2000).

An intermodal personal mobility platform allows travelers to query, book, use and pay any combination of mobility services (Beutel et al., 2016; Beutel et al., 2018). 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 service interfaces 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 service interfaces, e.g., which phases it is composed of, which tools are used and so on, have been conducted. Furthermore, we identified both benefits and shortcomings to elicit potential areas of improvement. These findings were discussed in (Ohler et al., 2018).

On the basis of these insights, we are developing an information system that supports service interface development with respect to design, data modeling, specification, and automated testing. It focuses on intensified collaboration and promotes a stronger integration to avoid diverging artifacts. Furthermore we aim at implementing features missing in available

Ohler, F. and Krempels, K-H.
Best Practice-based Evaluation of Software Engineering Tool Support: Collaborative Tool Support for Design, Data Modeling, Specification, and Automated Testing of Service Interfaces.
DOI: 10.5220/0007227903480355
In *Proceedings of the 14th International Conference on Web Information Systems and Technologies (WEBIST 2018)*, pages 348-355
ISBN: 978-989-758-324-7

tools that were identified as relevant in this context.

Starting with a first implementation of the developed tool support based on the identified requirements, we conducted a best practice-based evaluation with experts of the Association of German Transport Companies (VDV). In this paper, we want to present the results of this focus group-based evaluation and discuss their implications for the envisaged tool support for collaborative service interface development.

A short overview over the requirements analysis proposed in (Ohler et al., 2018) is given in Section 2. For these requirements, we developed a prototype, whose implementation is discussed in Section 3. In this paper, we discuss the best practice-based evaluation of the usability, feature coverage and fitness of our approach. We introduce the evaluation method in Section 4 and present the results in Section 5. In closing, we then conclude with a discussion in Section 6 and an outlook in Section 7.

2 REQUIREMENTS FOR TOOL SUPPORT FOR SERVICE INTERFACE DEVELOPMENT

The traditional service interface development process leads to the highest rated problem constituting the separation of service interface specification, data models and test tools as identified by (Ohler et al., 2018). Additional problems with high impact include the lacking central availability of latest versions of the artifacts and problems while merging documents after parallel work.

Thus, the core derived requirements comprise the following two:

1. integrated tool support covering all aspects that were previously scattered over different tools
2. support for cooperative and parallel editing

These requirements also promise to have the highest impact on the architecture of the tool and promise the largest improvements for users of the tool. Many further requirements were identified concerning specification and documentation functionalities, support for artifacts of all project phases, etc. For an early evaluation, we decided to focus on the protocol design phase that includes the actual data type specification and documentation. Therefore, additional requirements were included:

- support for XML / XML Schema for the data type specification
- export functionality to docx format corresponding to VDV formatting styles

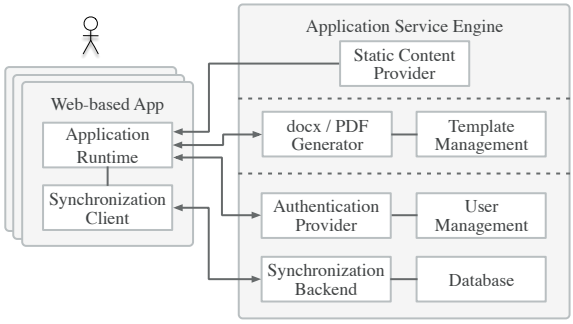

Figure 1: System architecture diagram of the web-based information system providing tool support for service interface development.

- multilingualism
- commenting of elements
- support for rich-text content editing

Another important requirement we are not yet implemented, but considered in the system architecture design approach, is the aspect of supporting testing through automation based on the information specified using the tool.

3 IMPLEMENTATION

The evaluated application is a web application that runs in the browser. The server hosting the website also offers a service to synchronize the data between the clients using Yjs[1] and a service to generate docx and PDF documents. The generated documents contain the entire documentation of the service interface development project and the given data type definitions rendered as tables. A sketch of the system architecture is shown in Figure 1 and a screenshot of the user interface is shown in Figure 2.

The application aims to integrate all aspects of a service interface documentation and specification in one model and generate the artifacts needed based on this information. At the time of the evaluation, it supports the documentation and specification of data types, functions, services and additional project documentation. Data types can be specified and documented using the XML Schema Language. Data types, functions and services can be documented and further chapters of the project documentation can be written using rich text (based on Quill[2]) in multiple languages. Additionally, functions can be grouped into services and their interaction (request and response) messages (defined in the XML

[1] http://y-js.org
[2] https://quilljs.com

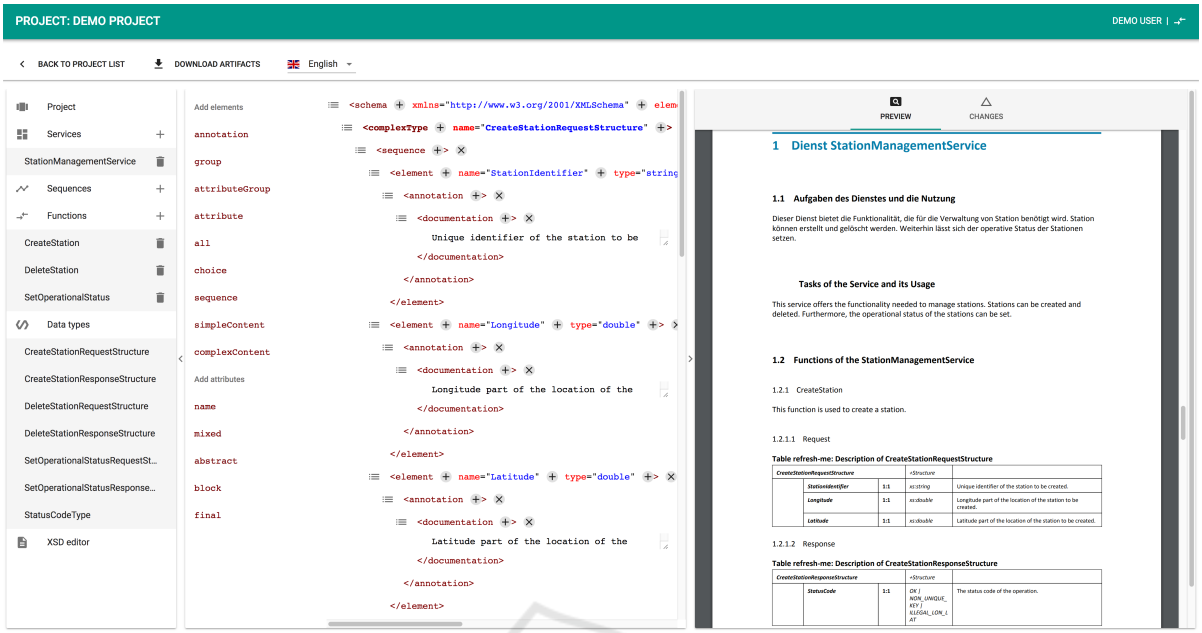

Figure 2: Screenshot of the user interface of the evaluated tool showing a simple demonstration project.

Schema part) can be linked. All these concepts (including the comments themselves) can be annotated with comments. Every aspect can be edited collaboratively in *near real-time*[3]. The service interface documented and specified this way can be exported to a docx document formatted according to the rules of the Association of German Transport Companies (VDV). This layout template is structured as follows: It starts with introduction chapters describing the goal and involved entities. Subsequently, possible common data types and the services are described in separate chapters. Each of the service chapters contains the corresponding functions, which in turn contain their corresponding data type. However, the layout template can be replaced to match the specifications of other standard setting bodies or the like. The user interface offers a preview functionality showing a PDF export of the same document (with some shortcomings). The XML Schema defined using the tool can also be downloaded.

Aside from UML modeling aspects and further requirements engineering artifacts, the application thus presents at least partial solutions for the core requirements derived in (Ohler et al., 2018).

[3]There are no timing guarantees, since messages are sent through the Internet and the systems involved are not assumed to run real-time operating systems.

4 FOCUS GROUP

An increasingly popular method to gather qualitative feedback is to host one or more focus groups (Greenbaum, 1999). A focus group comprises a small group, often between 6 and 12 people (Longhurst, 2016), hosted by a trained moderator. It is often advisable to include exercises (Colucci, 2007) as proper way to approach the participants and to steer the focus on the target area.

> "Focus groups are useful when it comes to investigating what participants think, but they excel at uncovering why participants think as they do." (Morgan, 1996)

Since we are gathering feedback at an early stage and the software is still in an experimental state, we want to make sure, that we are able to dig deeper in case of unclear answers so that we can draw the right conclusions from the feedback. Thus, we deem the focus group method as appropriate to gather feedback in our current situation.

The applied methodology is now shortly described. The focus group was conducted with seven project members of an ongoing research project in the context of information systems for public transportation. Six of them were male, one of them female. The test persons were experienced in the context of service interface development and public transportation; many of them had more than ten years of experience in this area. The focus group was led by a

moderator and written notes were taken by a separate transcript writer. Additionally, the focus group was recorded on audiotape. It was held in a closed room of a university building and all participants were on site. The event took 110 minutes, starting with a 20 minute introduction, followed by 25 minutes of hands-on exercises, and concluded by a 65 minute group discussion. The introduction presented the methodology and goal of the focus group and demonstrated how to use the application described above. At its end, exercises were given to the test persons that were to be solved using the application. The test persons 5, 6 and 7 decided to work in a group using only one computer, the others worked by themselves. Thus, five different exercises were worked on.

For all exercises, a new function was to be added to an existing service. For this new function, the corresponding data types were to be specified. At least one of the concepts should be documented (multilingually, if applicable). The results were to be contained in the generated document at the correct position.

The expected tasks involved in dealing with such an exercise were the following:

- Users had to create the function and assign it to the correct service.
- They had to create the data types for the request and the response of the function they were creating and link them with the function.
- For that, they had to define the data types using the XML Schema editor.
- They were free to choose whether to document data types within the XML Schema editor (in plain text) or to document their function or service using rich text.
- In case they did not link their function to a service, it would not appear in the generated document.
- Failing to use the data types for the function request or response element would lead to the defined data types being listed in the "common data types" section of the generated document.
- They therefore had to check that their work actually appeared in the document as expected.

During the hands-on exercise the moderator assisted the test persons in case of questions or problems. All test persons succeeded in completing their tasks. The subsequent group discussion was structured along the requirements tackled and an additional part for further feedback. This is also reflected in the structure of the section below.

5 RESULTS

The presentation of results in the context of qualitative content analysis includes categorization of the results (Mayring, 2014). These are given together with an overview over the feedback in Table 1. The citations given below were translated retrospectively.

5.1 Integrated Tool Support

Usability Rating. The usability was rated as intuitive.

The way we structured the information (into services, functions, data types) was deemed challenging by some test persons. The information about a service, its functions and their data types should be represented more closely together. It was seen as a new kind of information fragmentation – exactly the problem we were trying to solve – in the eyes of some test persons.

Feature Coverage. Test persons wanted to be able to navigate between interconnected aspects more easily, e.g., from a function to its request data type definition.

Approach Fitness. The approach towards the integration of aspects that were previously scattered over several tools was found to be suitable.

5.2 Collaboration

Usability Rating. Some test persons had problems finding their currently relevant aspects in the growing lists of functions, XML elements etc.

The visible portion of the project *scrolled away* in case someone else inserted content further up, which disrupted the interactions.

New content was always appended at the end of lists hindering information locality of related concepts.

For newly created functions, their ownership was unclear, since they were all named "new function" until users renamed them.

Feature Coverage. The test persons wanted to be able to manually set the focus to the concepts they were about to work on, e.g., by filtering or collapsing unrelated concepts.

Approach Fitness. Overall, the group agreed that the way we tackled the collaboration fits the needs and is superior to using git[4] (or similar tools) as a foundation.

[4]https://git-scm.com

Table 1: Feedback Overview.

	Usability Rating	Functional Coverage	Fitness of Approach
Tool Integration	+	+	++
Collaboration	−	+	++
XML Schema Support	−−	−−	+
Docx Export	++	++	++
Multilingual Support	+	++	++
Comments	+	+	+
Rich-Text Editing	++	++	++

++ very positive, + rather positive, − rather negative, −− very negative

> "I consider the problems to be marginal compared to the time and effort necessary to merge two different versions." (Test Person 4)
>
> "I regard the benefits of collaborating in near real-time as more important for the practical use cases than the disadvantages that might occur when two people actually work together in an unsynchronized way." (Test Person 3)

5.3 XML Schema Support

Usability Rating. Test persons, who usually use graphical notations of XML Schema, were not sufficiently familiar with the actual syntax and found the list of suggestions as not fitting for them.
The part of the suggestion list containing attributes was not recognized by all test persons.
Some test persons did not figure out how to delete attributes.

Feature Coverage. Some of the test persons asked for graphical editing of XML Schema documents (as possible e.g. in Altova XMLSpy[5]).
Furthermore, validation of the XML Schema file was requested.
Some test persons wanted for a mechanism to prevent the accidental deletion of a node including its children, such as a confirmation dialog.
Moreover, test persons were missing auto-completion features, e.g., when specifying the type of an element.
The group requested the possibility to import XML Schema files edited with external tools into the XML Schema editor to overcome the aforementioned problems.
Some test persons asked for a way to spread the data types definitions over several XML Schema files to increase lucidity.
Additionally, the possibility of including or importing XML Schema documents from other projects was seen as important.
A further request was made for an experimentation feature, that lets you revert your changes in case the experiment goes wrong while respecting changes from other users.

Approach Fitness. Our support for the creation and editing of XML Schema documents was not regarded as sufficient in its current state.

5.4 Docx Export

Usability Rating. No problems were mentioned regarding the usability.

Feature Coverage. The automatic generation of cross references was seen as "incredible help" (Test Person 3) and as a "huge step forward" (Test Person 3).
Some test persons wanted to be able to reconstruct which parts of the document were changed after downloading it from the tool.

Approach Fitness. The generated docx document containing the documentation and the definitions modeled using the tool was rated positively.

5.5 Multilingual Support

Usability Rating. It was not made sufficiently clear to the test persons, that the layout template of the target standard setting body (VDV) only included the English documentation on function and data type level, while including the German and the English documentation on project and service level.

Feature Coverage. Test persons asked to prevent the user from writing text that will not get included in the generated document.
After an explicit inquiry by the moderator, the group found that allowing a side-by-side editing of texts in two different languages would be superior compared to the current way, where the user

[5] https://www.altova.com/xmlspy-xml-editor

only sees the text in one language and changes the language using a flag icon.

Approach Fitness. Despite the mentioned issues, the approach was rated positively.

5.6 Comments

Usability Rating. Some test persons expected to have their comments appear in the generated document, which was not the case (no request was made to change this behavior).
The fact that comments have to be submitted explicitly was perceived as a break in the usage pattern compared to the rest of the tool.

Feature Coverage. The test persons requested that there should be a hint elaborating that comments have to be submitted explicitly.
It should easily be visible whether there are (new or any at all) comments for project elements.
Additionally, it was suggested that one should be able to classify a comment as a bug, problem, praise, information, et cetera.

Approach Fitness. The approach was well received.

5.7 Rich-Text Editing

Usability Rating. No problems were mentioned regarding the usability.

Feature Coverage. Some test persons asked for a mechanism to prevent accidental changes to texts written by other people.

Approach Fitness. The approach was well received.

5.8 General Feedback

Usability Rating. The PDF preview windows refreshed and scrolled to the beginning too often.
Test persons had to search for their own content and thus couldn't benefit from it.

Feature Coverage. It was requested that the PDF preview should jump to the position that the user just edited.
Furthermore, there should be a mode switch between continuously refreshing and manually refreshing the PDF preview.
Test persons asked for the integration of a chat feature.

Employing the usability problem metrics and the rating scale from (Nielsen, 1993), Table 2 gives an overview over the identified usability problems and their severity.

Table 2: Overview over the usability problems and their severity rating.

Usability Problem	Severity
Integrated Tool Support	
Information fragmentation	3
Collaboration	
Finding relevant parts	2
Scrolling away	3
"New function" ownership	1
XML Schema Support	
Insufficiency of suggestion list for users accustomed to graphical notation	3
"Add attributes" not found	2
Could not delete attributes	4
Multilingual Support	
Template specifics	2
Comments	
Comments do not appear in document	1
Comments have to be submitted	2
General Feedback	
PDF preview refreshes too often	1
PDF preview does not scroll to change	1

Rating scale taken from (Nielsen, 1993):

0 I don't agree that this is a usability problem at all

1 Cosmetic problem only: need not be fixed unless extra time is available on project

2 Minor usability problem: fixing this should be given low priority

3 Major usability problem: important to fix, so should be given high priority

4 Usability catastrophe: imperative to fix this before product can be released

6 DISCUSSION

The test persons discovered a lot of relevant issues and asked for reasonable additional features, especially in the context of XML Schema support. One problematic request was to be able to edit the XML Schema with external tools and then merge the results back into the tool. This was mentioned for two reasons: Firstly, as a short term solution to overcome the other missing features of the tool. Secondly, to be able to do experiments and import the changes in case of success or discard otherwise (reverting the information in the tool would possibly also revert changes made by others concurrently). We consider this to be problematic, since the tool would lose a large part of its cooperative aspect. The visibility and accessibility of current versions of the artifacts would then be

dependent on the discipline of the editing person. During the group discussion, this was already considered to be a double-edged sword:

> "How to offer the benefits [of being able to modify project parts externally] without opening the floodgates to abuse, I don't know. I think, at this point we are on the horns of dilemma." (Test Person 3)

It could, however, be possible to implement the experimentation feature inside the tool, such that one can do experiments and, in case of failure, discard own changes up to a specific point without interfering with foreign changes. Yet, this approach would reach its limits as soon as another user either explicitly built upon one's changes or implicitly relied on an artifact in its changed state.

Looking at the feedback for the XML Schema support, the negative scores in Table 1 are comprehensible. However, the test persons did not mention that a lot of problems were implicitly prevented by the DOM-based approach that is compatible with the cooperative near real-time editing aspect. One important benefit of this approach is that the content of the XML Schema editor contains a valid XML document at all times preventing, e.g., that there are opening tags without their closing counter part or only partially written tags such as "<element" missing the trailing ">". This way, it is much easier to parse the document and, e.g., present users a list of elements defined in the XML Schema editor especially in case of parallel work. The interactions with the editor more easily produce meaningful *deltas* (e.g., node added) than in the situation of a text editor with XML content, where deltas would be character insertions or deletions. This promises to have beneficial effects on the synchronization layer. Additionally, combining this approach with a graphical editing mode seems technically feasible. Thus, we consider the approach valid even though there still is a lot of work to do.

To ease the new information fragmentation problem we will try to let the users determine the structure of the result document by mirroring it in the project structure and thus make it easier for users to see the connections between services, functions and data types.

7 CONCLUSIONS, CURRENT WORK AND OUTLOOK

We presented the results of a focus group evaluating a first implementation of integrated tool support for service interface development projects. This evaluation is part of a development project towards a stronger integration and intensified collaborative tool support for the design, data modeling, specification, and automated testing of service interfaces. The overall feedback, especially concerning the chosen approach, was very positive. This is also reflected in the fact that we are already getting inquiries from project partners, VDV working groups and others showing interest in using the tool. Thus, we are currently remediating the issues identified, many of which are expected to be resolvable short-term. We are looking forward to deploy and evaluate the tool in a real-world service interface development project with the aforementioned partners. Furthermore, we are working on extending the type of renderings we can generate with the tool. These include generated beans, generated communication adapters, generated test templates or even test cases.

A lot of the requirements we derived in (Ohler et al., 2018) are not yet satisfied in the current state of the tool. Additional requirements have emerged during the focus group and are expected to surface during further evaluation steps. Finding solutions to integrate these into the tool will constitute future work.

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).

REFERENCES

Beutel, M. C., Gkay, S., Jakobs, E.-M., Jarke, M., Kasugai, K., Krempels, K.-H., Ohler, F., Samsel, C., Schwinger, F., Terwelp, C., Thulke, D., Vogelsang, S., and Ziefle, M. (2018). Information system development for seamless mobility. In *7th International Conference on Smart Cities and Green ICT Systems, SMARTGREENS 2018, and 4th International Conference on Vehicle Technology and Intelligent Transport Systems, VEHITS 2018, Funchal, Portugal, 2018, Revised Selected Papers*. M. Helfert and C. Klein and B. Donnellan and O. Gusikhin and A. Pascoal.

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.

Colucci, E. (2007). "Focus groups can be fun": The use of activity-oriented questions in focus group discussions. *Qualitative Health Research*, 17(10):1422–1433.

Dalmau, M., Aniorté, P., and Roose, P. (2000). A method for designing cooperative distributed applications. In *Proceedings of the 4th International Conference on Designing Cooperative Systems, COOP 2000, Sophia-Antipolis, France, May 23-26, 2000*, pages 369–384.

Garrido, J. L., Gea, M., Gutiérrez, F. L., and Padilla, N. (2000). Designing co-operative systems for human collaboration. In *Proceedings of the 4th International Conference on Designing Cooperative Systems, COOP 2000, Sophia-Antipolis, France, May 23-26, 2000*, pages 399–410.

Greenbaum, T. L. (1999). *Moderating focus groups: A practical guide for group facilitation*. SAGE Publications.

Inayat, I., Marczak, S., Salim, S. S., and Damian, D. (2017). Patterns of collaboration driven by requirements in agile software development teams. In Grünbacher, P. and Perini, A., editors, *Requirements Engineering: Foundation for Software Quality*, pages 131–147, Cham. Springer International Publishing.

Kusumasari, T. F., Supriana, I., Surendro, K., and Sastramihardja, H. (2011). Collaboration model of software development. In Syaichu-Rohman, A., Hamdani, D., Akbar, S., Adiprawita, W., Razali, R., and Sahari, N., editors, *International Conference on Electrical Engineering and Informatics, ICEEI 2011, Bandung, Indonesia, 17-19 July, 2011*, pages 1–6. IEEE.

Longhurst, R. (2016). Semi-structured interviews and focus groups. In *Key Methods in Geography*, pages 143–156.

Mayring, P. (2014). *Qualitative content analysis: theoretical foundation, basic procedures and software solution*.

Morgan, D. L. (1996). *Focus groups as qualitative research*, volume 16. SAGE Publications.

Nielsen, J. (1993). *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Ohler, F., Beutel, M. C., Gökay, S., Samsel, C., and Krempels, K. (2018). A structured approach to support collaborative design, specification and documentation of communication protocols. In *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering, ENASE 2018, Funchal, Madeira, Portugal, March 23-24, 2018.*, pages 367–375.

Whitehead, J. (2007). Collaboration in software engineering: A roadmap. In Briand, L. C. and Wolf, A. L., editors, *International Conference on Software Engineering, ISCE 2007, Workshop on the Future of Software Engineering, FOSE 2007, May 23-25, 2007, Minneapolis, MN, USA*, pages 214–225. IEEE Computer Society.