Using Template Analysis as Background Reading Technique for Requirements Elicitation

Sebastian Stein, Yves Lauer, Marwane El Kharbili
IDS Scheer AG
Altenkesseler Str. 17
66115 Saarbrücken
Germany
sebastian.stein, yves.lauer, marwane.elkharbili@ids-scheer.com

Abstract: Requirements cannot be collected, but must be elicited from people's tacit and systems' embedded knowledge. It is a proven approach to use existing systems and manuals as a source. Literature suggests using background reading to elicit requirements from domain descriptions and manuals. Besides content analysis, no concrete technique describing how to actually conduct background reading is available in literature. In this paper, we evaluate the usage of template analysis as a technique for background reading. We applied template analysis in a project to extract requirements from 35 success stories about process performance management. We found template analysis to be very useful not just for eliciting requirements and creating a shared understanding of the studied domain, but also for helping new employees to get familiar with the domain. We also formulated competence questions to document and communicate requirements, but this did not prove helpful and hence we would not recommend it.

1 Introduction

Today, it is a well-known fact that carefully defining and managing requirements is a fundamental base for successful software engineering projects [DC06]. In the past decades, requirements engineering has evolved from being a sub-discipline of software engineering into a well established research discipline. Standard requirements engineering literature like [KS04, Dav93] shows that requirements engineering comprises many different aspects such as the requirements engineering process, elicitation and analysis, validation, change management, viewpoint analysis, formal specification, etc. Not surprisingly, research contributions usually focus on a specific aspect of requirements engineering. This paper focuses on requirements elicitation.

Requirements elicitation is an early activity in the requirements engineering process aiming at acquiring and understanding requirements. Usually, this is not a task of just collecting clearly articulated requirements, but instead intensive work is needed to transform embedded knowledge into explicit one. There are many possible knowledge sources like human beings (domain experts, current users, stakeholders, etc.) but also artefacts like manuals, domain descriptions, laws and regulations, legacy system, and competing soft-

ware products [KS04, Dav93]. Depending on the knowledge source, different requirements elicitation techniques are applicable.

Requirements elicitation consists of two phases [Zha07]: problem analysis and product specification. In the problem analysis phase, the requirements engineer creates an understanding of the problem domain and the context, whereas in the product specification phase concrete requirements for the product to be developed are defined. Existing requirement elicitation techniques can be grouped in four categories [Zha07]: conversational methods (e. g. interview), observational methods (e. g. ethnography), analytic methods (e. g. background reading), and synthetic methods (e. g. prototyping).

Byrd et al. [BCZ92] provide a similar categorisation, but they do not just cover requirements elicitation techniques but also include knowledge acquisition techniques used in knowledge engineering. They show that a great similarity of techniques exist in both domains. Zhang [Zha07] extends this work by creating a selection matrix supporting requirements engineers in selecting an appropriate requirements elicitation technique. For example, if no access is given to domain experts, conversational methods cannot be applied. However, the selection matrix by Zhang is not based on empirical findings.

In this paper we present a technique to acquire requirements from existing texts. We use the technique template analysis to analyse texts and we document the acquired requirements by formulating competence questions. We show that template analysis is a promising background reading technique, but that the procedure of formulating competence questions was not very helpful. We applied template analysis in a public funded research project in order to document current requirements of users and customers in process performance management projects.

This paper is structured as follows: In the following section we discuss other background reading techniques and we show why we selected template analysis in our case. Sect. 3 describes the research material used and the research context where we conducted the study. In Sect. 4 we first give a brief overview of template analysis, show how we have done it, how we analysed the results, and how we used competence questions to document the requirements extracted. We provide a comprehensive discussion in Sect. 5 about the lessons learnt. The discussion will be useful for other requirements engineers to judge if template analysis is applicable in their specific context. The paper is shortly summarised at the end.

2 Background and Related Work

In a recent research effort, we wanted to document typical requirements requested in performance management projects. We had no direct access to domain experts, but we had a corpus of 35 success stories from past projects. Using the selection matrix [Zha07], we decided to use an analytical method, namely background reading. Using existing documents and manuals as a source for requirements elicitation is an accepted approach discussed or at least listed by many authors [KS04, Zha07, BCZ92]. Different terms are used for such techniques like background reading, document analysis or documentation studies. Back-

ground reading is not just used to acquire requirements, but also to understand the overall culture in an organisation or project [Zha07].

Even though many authors point to the value of background reading and document analysis, Rayson et al. [RGS99] show that there are neither specific techniques nor instructions documented or even evaluated in literature. Therefore, they evaluate the usage of natural language processing (NLP) techniques. In NLP, texts are analysed by linguistic algorithms to extract common themes and topics. Ideally, no human interaction is needed. Using NLP techniques to extract knowledge from existing documents is a common approach in other disciplines as well. For example, Aussenac-Gilles et al. [AGBS00] demonstrate the usage in the ontological engineering research domain. Here, it is important not to confuse this with approaches like Fliedl et al. [FKM+07], who are using NLP techniques to analyse requirements documents in contrast to gathering requirements. NLP is a good approach if lengthy texts are available. This was not the case, because we only had 35 success stories. Therefore, we could not apply NLP techniques.

Besides NLP based techniques, we also found publications using content analysis [Kri03] as a reading technique. For example, Shea and Exton [OE04] use content analysis to extract requirements for a software visualisation tool from a set of texts like bug reports. They use predefined categories. They analyse the results through a quantitative analysis measuring how often a certain category is used. Content analysis is a good requirements elicitation technique if a set of categories can be defined prior analysing the actual texts. This requires a certain degree of understanding of the domain to be studied, but it is not applicable in case the texts should be used to get an understanding of the domain and to elicit requirements. We had no prior understanding of the domain covered by the success stories and we could not create a categorisation. Therefore, we could not apply content analysis. We looked for another technique, which is similar to content analysis, but does not require a fixed categorisation. We found template analysis [Kin98], which we are evaluating in this paper.

Our research partners asked us to document the acquired requirements as competence questions. Competence questions [MI96] are either literal questions like "How often was process XYZ executed?" or formal ones like formulating WSML queries. The questions are used during ontology development. The created ontology is checked whether it can be used to answer the competence questions defined before. It was the task of our research partners to create an ontology based on the requirements we gathered from the success stories. Therefore, we decided to try documenting the requirements as competence questions. During this paper, we also reflect on this usage of competence questions for requirements documentation.

3 Use-case and Research Material

The work presented here is part of the research project SUPER[1]. The SUPER research project investigates how semantic technologies like ontologies and reasoners can be used

[1] http://www.ip-super.org/

in business process management (BPM) [SS08]. An important aspect of BPM is monitoring and analysing existing IT systems and executed processes. This is known as process performance management. The SUPER research project investigates how semantic technologies can support process performance management [CAdMZ+07, AdMPvdA+07]. However, just focusing on what might be possible in process performance management using semantics does not automatically guarantee the gain of useful research insights. Therefore, it is important to take into account current user requirements and what is already possible with today's technologies, so that the state of the art can be extended and not just reinvented using another technology.

We, as a vendor of BPM software were asked to provide an overview of today's problems tackled within process performance management projects. Our company is doing such projects since several years all over the world. Fortunately, some of the past projects are documented as marketing oriented success stories. Each success story consists of few presentation slides describing the initial situation, the customer, the customer problem, the applied solution, and the benefits gained. The success stories are created by project leaders or other project members, both having extensive domain knowledge. At the time this research was conducted, we had access to 35 success stories from different industrial domains like chemical/pharma, finance, automotive, logistics, insurance, capital goods, telecommunications, and utilities. The success stories are written by different individuals and as they are meant to be marketing tools, they are written bluntly. Therefore, the material is too subjective and weak to be used in any quantitative study. Also, there is too little text to apply NLP. Still, the success stories contain valuable domain knowledge, which we want to make explicit. The success stories are not available to the public. The following section details how we applied template analysis to extract requirements from the success stories and how we used competence questions to document the requirements.

4 Applying Template Analysis

4.1 Overview Template Analysis

This section provides a short introduction to template analysis and demonstrates how we applied template analysis to acquire requirements from the success stories. Template analysis[2] [Kin98] is a qualitative research method for analysing research material given as texts (i. e. the corpus). Codes are terms representing possible themes in the corpus. A code template consists of a set of codes. The person performing template analysis is called coder.

Template analysis is an iterative process. The first step in template analysis is creating the corpus. Afterwards, the coder has to get familiar with the text by reading it. Next, the code template is created by the coder. The first version of the code template should be based on the coder's understanding or perception of the studied domain. Usually, the coder uses background knowledge of theories he expects to be present in the corpus. Another possible

[2] http://www.hud.ac.uk/hhs/research/template_analysis/index.htm

source is existing classifications or taxonomies. A code template can be a plain list of terms or a hierarchy of terms. Afterwards, the corpus is analysed by one or more coders who read the text and assign codes from the code template to segments of the corpus. While coding the corpus, the code template is reworked to better reflect the content. Creating the code template and coding the corpus is an intertwined iterative process. The code template is reworked several times, new codes get introduced, and other codes are removed or split into more detailed codes. This is a major difference compared to content analysis [Kri03], because in content analysis the categories (code template) are fix.

After some time, usually after coding one third of the corpus, the code template stabilises. After having a stable code template, the actual coding starts. Each coder marks segments of the corpus and assigns codes from the code template to the marked segments. It is possible to assign multiple codes to one segment. Segments can be freely selected, so one word can belong to multiple segments. Coding can be done manually using for example a printout of the corpus and marking segments with different colours, where each colour represents a code. There are also different software tools available for this task. Such tools are usually called qualitative data analysis (QDA) tools.

After coding the whole corpus, coders have a better understanding of the content. This understanding can be further extended by sharing the coding results, for example in a group discussion. It is also common to reflect on the coding process itself and how the own perception of the domain changed while coding. In summary, template analysis is a tool for sense-making. It is not meant to cluster the text according to a predefined set of criteria. Instead, template analysis is a tool to work with the text to understand it and gain insights. It is not the aim of analysing the coding results to judge if coding was done correctly. For example, if a coder applied a code to completely different segments than all other coders, it does not mean he did the coding wrong. Instead, it should be discussed why the coder used the code in a different way. Such discussions usually challenge the current individual perception and help to better understand the corpus.

4.2 Application of Template Analysis

We started doing template analysis with two people. We both have profound knowledge in BPM, but only initial insights into process performance management. We have not done any process performance management projects nor used the specific software provided by our company. First, we copied the content of the different success stories into one ASCII text file removing sections like customer descriptions. We created an initial version of the code template using general textbooks about service level agreements (SLA) and key performance indicators (KPI). We expected those topics to be main themes in process performance management projects. We structured the code template around the core terms *costs*, *time*, and *quality*. Beneath those terms, we formulated concrete codes like *cost reduction*, *defect reduction* or *quality improvement*. Both of us created the code template individually. Afterwards, we merged the two code templates after discussing and clarifying the scope of each code. We learnt that we both had different ideas and expectations about process performance management projects.

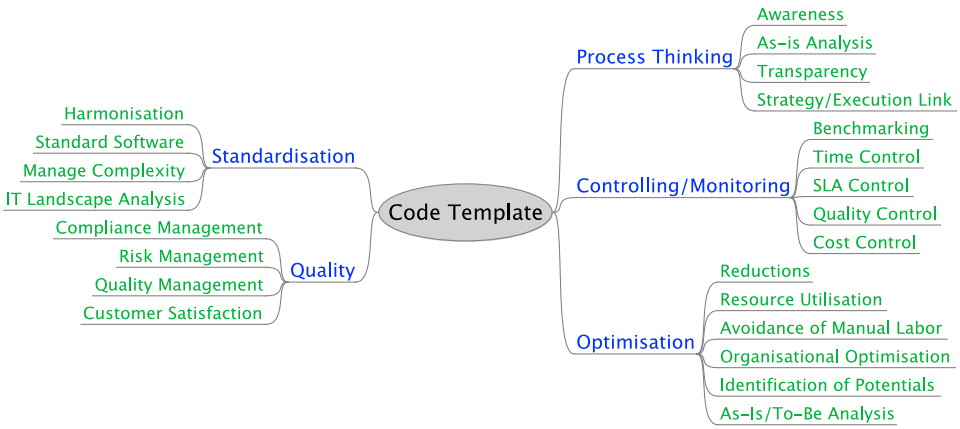

Figure 1: Final code template

We did a test coding with the initial version of the code template. Testing the code template revealed that the initial code template was not reflecting very well the content of the corpus. For example, many customers wanted to stimulate process awareness among their employees by mining and documenting existing processes. Therefore, we completely restructured the code template. The final version of the code template is shown in Fig. 1. We only used leafs for coding. Before we started coding, we created a textual description for each category and each code. We discussed this description to ensure that we have a similar understanding of the codes' meanings. We can confirm that the code template is not just defined at the beginning but revised several times as the coder gets a better understanding of the studied texts.

After having a stable code template, both of us coded the whole text. Coding the whole text took one day for each of us. We used Weft QDA[3] for coding, because it is freely available. Unfortunately, Weft QDA does not support multiple coders. Therefore, each coder had to copy the database containing the text and code template. Having several databases at the end makes comparing the results much harder, because inter-coder analysis is not supported by the tool but instead must be done manually outside of Weft QDA. However, this is not an inherent problem of template analysis, but just related to the tool we used.

A third researcher joined after both of us had finished coding. We wanted to test what happens if a coder has to use an existing code template. First, the new coder read the code template with the corresponding textual descriptions. Afterwards, we discussed with the coder the meaning of the different codes and how he understood the code template. This already revealed that he had thought of additional things, which were not present in the code template. Afterwards, he performed the coding using the unchanged code template. We did not change the code template, so that the two of us, who already finished coding, do not had to do the coding again. We included the results of the new coder in the overall analysis process. The following subsection describes how we analysed the coding results.

[3]http://www.pressure.to/qda/

4.3 Results of Template Analysis

After all coders had coded the whole text, we had three Weft QDA databases containing the results. Weft QDA has some basic analysis and reporting functionality, but which was not sufficient in our case. Therefore, we did the actual analysis outside of Weft QDA.

We created a list for each code containing the text segments marked with this code by each coder. We used those lists to discuss the outcome. For example, the discussion revealed that we had a different interpretation of some of the codes. This was not avoidable, even though we had one common textual description for each code. We were able to learn about why a certain coder applied a code. Having coded the text differently was therefore a good way to initiate a discussion between the participants and to strengthen the understanding. We also learnt that some codes were not applied very often, because the software provided by our company only supports those use-cases partially and therefore such use-cases are not within the focus of customers buying our software and services. It is important to understand that template analysis is not finished when the coding is complete. Instead, discussing the results between the different coders is an important part of template analysis. Template analysis is done to work with the text, but not to manually cluster the text according to the code template. This also means that if a certain code is only used a few times, this code can still represent a major aspect.

4.4 Documenting Requirements as Competence Questions

As discussed before in Sect. 2, formulating competence questions is a common approach in ontology engineering. To support our research partners and to communicate the insights we gained by analysing the success stories, we formulated a set of competence questions. For each code we looked at the text segments marked with the code and transformed the text segments into questions. For example, we created the following five competence questions for the code *as-is analysis*:

- How is process XYZ looking like?
- Can we extract the executed processes from our running IT systems?
- Where do we have weak spots and bottlenecks?
- Which processes are used/executed very often?
- What is our organisational structure?

We formulated the questions from the viewpoint of the users based on the corpus. We did not create one question for each marked text segment, but tried to summarise several coded text segments in one question. For example, the first question above is based on the following three original text segments marked with the code *as-is analysis*:

- “Process analysis of as is HR-processes”

- "visualize the as-is-processes in different regions"
- "Documentation of lived SAP-SD processes"

The example competence questions given above show that it is impossible to simply answer them using some kind of software. It was an important outcome to demonstrate to our research partners that many questions raised in process performance management projects require intensive consulting services and not just software. However, the initially formulated competence questions were too abstract to be useful for our research partners, because they it was their task to create a formal description of the domain by means of an ontology. The questions provoked a discussion among the partners about the nature and scope of process performance management projects, but they could not be used to judge how semantic technologies can be applied in this domain. Therefore, we refined the questions further to provide a more detailed view by interpreting the statements given in the success stories. The limitations of this refinement are discussed in Sect. 5. The following list shows some of them for the code *as-is analysis*:

- What data do we use in our business processes?
- What processes use what data?
- What is the configuration of my business process execution system?
- Can we have a global schematic map of the organisations/roles involved in processes/activities?
- Can we view a dashboard of the number of instances of each process over a period of time?
- Can we view a dashboard of the frequency of execution of each process over a period of time?

5 Discussion

5.1 Template Analysis as Requirements Elicitation Technique

In the past sections, we presented how we used template analysis as a background reading technique for requirements elicitation. In this section, we reflect on this usage of template analysis. This discussion also includes the usage of competence questions to define and communicate the requirements extracted.

We used template analysis as a tool to analyse an existing comprehensive set of texts. The analysis was done in a small team of three people. We noticed that applying template analysis requires a certain degree of open-minded thinking and a high level of collaborative disposition. It is not the aim of template analysis to prove or disprove a certain

expectation, but instead to develop the perception of the participants. Reworking and discussing the code template is an important element of template analysis enabling creative thinking. For example, in our case study a third coder joined at a later point. Adding his view to the analysis showed us some shortcomings in our code template. The resulting discussions helped all participants to better understand the corpus. We also used basic statistical analysis to visualise different understandings and perceptions.

Using a tool like Weft QDA for coding the text allows having a direct link between the codes and the coded segments. This ensures that one can easily look up to which text segments a code was applied to. Even though this allows an easy navigation between source and result, it is unclear how to integrate such a qualitative data analysis tool with existing requirements management tools.

If we compare our usage of template analysis to previous usages by other researchers, we can see that there are different implementations of the approach. For example, Carey et al. [CMO96] also had several coders, but they were trying to do template analysis in a much stricter way. They quantitatively tested the code template and also made a detailed statistical assessment about the application of the code template in the coding process. We did not have such a formal approach to template analysis. Instead, we tried to use it more as a creative thinking technique to provoke discussions and disagreements between the participants.

5.2 Template Analysis, NLP and Content Analysis

Template analysis and NLP are very different approaches to do requirements elicitation on existing texts. NLP is meant to be an automated approach creating a knowledge base, which can be used by a requirements engineer. In contrast, template analysis is a manual approach, where requirements engineers create a knowledge base according to their own needs. As a result, requirements engineers gain a better understanding of the domain. Template analysis enables collaboration between several people. We expect template analysis to be easier to use, because no complicated installation and setup of software is required. We also expect that it is easier to understand the results and how they were created, because explaining and understanding NLP is not an easy task. On the other hand, template analysis does not scale very well with the corpus length and number of coders involved. Each coder has to be familiar with the whole text. In case of very long texts, this can be almost impossible and it may take too much time and resources. As both techniques are very different, we do not expect a competition between both. Instead, both techniques might be combined. For example, NLP can be used to create an initial version of the code template by analysing the text for common terms. Such a generated code template might be more accurate in reflecting the content of the text and hence provide a better starting point for the coders continuing with template analysis.

Content analysis is more similar to template analysis than NLP. Content analysis is usually used to analyse communication like interview transcripts, but it can also be applied to other texts. In contrast to template analysis, content analysis uses a fixed set of categories.

Content analysis has a very strong focus on providing quantifiable results through statistical analysis. We expect content analysis to be a better background reading technique than template analysis if a) the requirements engineers are already familiar with the domain and b) quantifiable results are needed. Both conditions were not given in our case.

5.3 Additional Advantages and Disadvantages of Template Analysis

A big advantage of using template analysis is that it is easy to learn and teach. After a third coder joined, we gave him a short introduction how to do template analysis and provided the available literature. This small training was enough that he could directly start coding the corpus and discuss with us the results.

Creating and changing the code template is easy for someone able to do requirements analysis. A high acquaintance with conceptual and abstract thinking is required. On the other hand, no special knowledge like computer science or qualitative research background is needed. This ensures that very different people can do template analysis. Having people with different know-how will even strengthen the result, because more perceptions of the domain are included and mixed. All of us had no specific background in the domain of process performance projects. Still, we were able to perform the analysis and we gained a valuable overview. All of us confirmed that template analysis and especially the coding step is an exhaustive task. One has to concentrate very much to use the codes always in the same way and not to change the meaning of codes while coding. This may happen, because while coding the own view on the domain changes. Here, we found a glossary to look up the meaning of each code to be very useful.

Maybe one of the biggest disadvantages is the time needed to carefully do template analysis. One has to read the corpus several times to develop the code template. Also, coding a text takes time and splitting this time over several days is not easy. During coding it is important to have a quiet surrounding. The best results are achieved if several people participate. However, this can be a problem, since people tend to be busy and thus booking meetings for group discussions can be a tricky task. If such constraints exist, template analysis might not be the technique to choose.

5.4 Competence Questions as Requirements Documentation

Besides using template analysis, we also evaluated the idea of formulating competence questions to document and communicate the results to our research partners. Formulating the competence questions was challenging for us. We felt it was an artificial approach. We often just turned a customer statement like “I want to extract my running processes!” around into questions like “Can I extract my running processes?” It is unclear to us if we really gained anything by doing that. Transforming statements found in the corpus into questions involves a reinterpretation of the person doing the transformation. We are not convinced that this helps in clearly defining requirements or if it just adds another level of

indirection.

In the end, the competence questions provided to the other research partners were not detailed enough to be useful for them. We had to refine the questions by providing more detailed ones. This step is hard to justify, because such detailed requirements are not supported anymore by the content of the success stories. Therefore, we would not suggest using competence questions to document the understanding gained by template analysis, unless the competence questions are defined by domain experts. If we take the requirement abstraction model [GW06] into account, we can say that template analysis is a valuable technique to deliver requirements on the product or feature level, but not to provide requirements needed on the function or component level. However, this might also depend on the source material like how detailed the textual descriptions are. A final judgement is not possible at this point and further evaluations are needed.

6 Summary

In this paper, we evaluated template analysis as a background reading technique for requirements elicitation. We also formulated competence questions to communicate the requirements found. Template analysis proved to be successful in our case study, but formulating competence questions was not helpful. To better understand the conditions under which template analysis can be used as a background reading technique, further case studies and experiments are needed. It is also important to compare template analysis to other requirements elicitation techniques. If possible, we will use template analysis again and evaluate it. However, we also encourage other practitioners and researchers to use template analysis in future requirements elicitation efforts and to report on their experiences.

References

[AdMPvdA+07] A. K. Alves de Medeiros, C. Pedrinaci, W. M. P. van der Aalst, J. Domingue, M. Song, A. Rozinat, B. Norton, and L. Cabral. An outlook on semantic business process mining and monitoring. In *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops*, volume 4806 of *LNCS*, pages 1244–1255, Vilamoura, Portugal, November 2007.

[AGBS00] N. Aussenac-Gilles, B. Biébow, and S. Szulman. Revisiting Ontology Design: A Method Based on Corpus Analysis. In *Knowledge Engineering and Knowledge Management. Methods, Models, and Tools: 12th International Conference (EKAW)*, volume 1937 of *LNCS*, pages 27–66, Juan-les-Pins, France, October 2000.

[BCZ92] T. A. Byrd, K. L. Cossick, and R. W. Zmud. A Synthesis of Research on Requirements Analysis and Knowledge Acquisition Techniques. *MIS Quarterly*, 16(1):117–139, 1992.

[CAdMZ+07] I. Celino, A. K. Alves de Medeiros, G. Zeissler, M. Oppitz, F. Facca, and S. Zöller. Semantic Business Process Analysis. In *Workshop on Semantic Business Process*

and Product Lifecycle Management (SBPM), volume 251 of *CEUR Workshop Proceedings*, pages 44–47, Innsbruck, Austria, June 2007.

[CMO96] J. W. Carey, M. Morgan, and M. J. Oxtoby. Intercoder agreement in analysis of responses to open-ended interview questions: Examples from tuberculosis research. *Cultural Anthropology Methods Journal*, 8(3):1–5, 1996.

[Dav93] A. M. Davis. *Software Requirements: objects, functions and states*. Prentice Hall, New Jersey, USA, 1993.

[DC06] D. Damian and J. Chisan. An Empirical Study of the Complex Relationships between Requirements Engineering Processes and Other Processes that Lead to Payoffs in Productivity, Quality, and Risk Management. *IEEE Transactions on Software Engineering*, 32(7):433–453, 2006.

[FKM+07] G. Fliedl, C. Kop, H. C. Mayr, A. Salbrechter, J. Vöhringer, G. Weber, and C. Winkler. Deriving static and dynamic concepts from software requirements using sophisticated taggingstar, open. *Data & Knowledge Engineering*, 61(3):433–448, 2007.

[GW06] T. Gorschek and C. Wohlin. Requirements Abstraction Model. *Requirements Engineering*, 11(1):79–101, 2006.

[Kin98] N. King. Template Analysis. In Gillian Symon and Catherine Cassel, editors, *Qualitative Methods and Analysis in Organizational Research: A Practical Guide*, pages 118–134. Sage Publications, London, UK, 1998.

[Kri03] K. Krippendorff. *Content Analysis: An Introduction to Its Methodology*. Sage Publications, London, UK, 2nd edition, 2003.

[KS04] G. Kotonya and I. Sommerville. *Requirements Engineering: Processes and Techniques*. Wiley, Chichester, UK, 2004.

[MI96] R. Mizoguchi and M. Ikeda. Towards Ontology Engineering. Technical Report AI-TR-96-1, I.S.I.R., Osaka University, 1996.

[OE04] P. O'Shea and C. Exton. The Application of Content Analysis to Programmer Mailing Lists as a Requirements Method for a Software Visualisation Tool. In *12th International Workshop on Software Technology Practice (STEP)*, pages 30–39, Los Alamitos, CA, USA, 2004. IEEE CS.

[RGS99] P. Rayson, R. Garside, and P. Sawyer. Language engineering for the recovery of requirements from legacy documents. Technical report, REVERVE project report, Lancaster University, May 1999.

[SS08] H. J. Schmelzer and W. Sesselmann. *Geschäftsprozessmanagement in der Praxis*. Carl Hanser Verlag, München, Germany, 6th revised edition, 2008.

[Zha07] Z. Zhang. Effective Requirements Development – A Comparison of Requirements Elicitation techniques. In *INSPIRE2007*, Tampere, Finland, 2007.