Friedrich-Alexander-Universität Erlangen-Nürnberg
Technische Fakultät, Department Informatik

FLORIAN SCHMITT
DIPLOMARBEIT

INTEGRATING MULTIPLE VIEWS IN A CODE SYSTEM

Submitted on 19 February 2016

Supervisor: Andreas Kaufmann, M. Sc., Prof. Dr. Dirk Riehle, M.B.A.
Professur für Open-Source-Software
Department Informatik, Technische Fakultät
Friedrich-Alexander-Universität Erlangen-Nürnberg

Versicherung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, 19 February 2016

License

This work is licensed under the Creative Commons Attribute 3.0 Unported license (CC-BY 3.0 Unported), see `http://creativecommons.org/licenses/by/3.0/deed.en_US`

Erlangen, 19 February 2016

Abstract

Mature requirements engineering is one of the key success factors of professional software development. This thesis proposes an approach that adapts core principles of qualitative data analysis, which is well established in social science, and transforms them into a holistic method for requirements analysis.

Our method addresses the process steps between the analysis of the target domain and the generation of a requirements specification and different types of domain models. In particular, it aims at inherently providing a high degree of traceability by the institution of an additional artifact, the so-called code system.

We state this code system to introduce significant advantages to the quality of requirements analysis with respect to systematics, documentation, maintainability and efficiency.

The proposed approach features the direct deduction of different types of domain models out of the gathered domain knowledge and a substantial support for the development of a requirements specification.

Contents

1 Introduction

1.1 Original Thesis Goals

This thesis aims at the conception and realization of an approach that implements the elicitation and analysis of requirements as an adaption of the Grounded Theory method. Two key features are intended: The possibility to directly deduct different domain models out of the so-called code system, which provide different views onto the target domain, represented in different types of UML diagrams; and the establishment of a process that supports the systematic development of a requirements specification.

The developed approach is applied to practical execution at the requirements analysis of the QDAcity tool, which is a cloud-based solution for qualitative data analysis (QDA). One of the goals of our research is to generate a theory how social science researchers carry out QDA with the help of so-called computer-assisted qualitative data analysis (CAQDAS) software.

In addition, this project aims at providing a natural language requirements specification for QDAcity.

1.2 Changes to Thesis Goals

The goals of this thesis were not changed.

2 Research

2.1 Introduction

In modern software development, specification and design methods normally advocate a modeling approach where the system to be developed is represented by describing the target domain and the design of the software by the use of the UML notation. This implies the use of different views, with each view presenting a different perspective on the system. Each view is specifically tailored to represent certain aspects.

These views can be subdivided, dependent on whether they describe static or dynamic aspects of the system.

Static aspects describe the main structure and the stable parts of a system. These aspects are represented by the use of structural diagrams. In UML, four types of structural diagrams exist: Class Diagrams, Object Diagrams, Component Diagrams and Deployment Diagrams. The static parts of the system are mainly represented by classes, interfaces, objects, components and nodes.

Dynamic aspects can be described as the changing/moving parts of a system. They are represented by Behavioral Diagrams. UML has the following five types of behavioral diagrams: Use Case Diagrams, Sequence Diagrams, Collaboration Diagrams, State-chart Diagrams and Activity Diagrams.

The UML notation features an extensive set of different views, however, in general, it does not address the issue of mapping and consistency checking among the multiple views. Therefore, multiple-view models often are afflicted with inconsistencies, incompleteness and ambiguities. This is particularly true when you consider that these models, correspondent to the domain analysis and requirements generation where they originate, are subjects of continuous changes.

This thesis propagates an approach that instead of first designing different views and then resolving consistency and quality issues, is intended to directly deduct all desired view representations from one collective artifact, the so-called code system. This code system serves as central information device. All information

about the target domain and the system to be developed is stored here and ordered in a way that allows a direct, automated deduction of views, and multiple types of models, respectively.

2.2 Related Work

2.2.1 Inter-Model Consistency and Multiple Views in UML

The establishment and checking of consistency between interrelated models of the same domain is one of the focuses of current RE research, however, up to now this research is confined to resolving issues between multiple views when they are already established:

(Lange, Chaudron, Muskens, Somers & Dortmans, 2003) focus on the frequency of the multiple types of inconsistencies and researches their impact to the design and implementation of software systems.

(Gomaa & Wijesekera, 2003) propose an approach to implement consistency checks with the help of OCL. Furthermore, this paper includes a detailed and elaborated case study, which illustrates the interconnections between natural-language description, use cases, state charts, class diagrams and behavioral diagrams.

(Simmonds, Van Der Straeten, Ragnhild, Jonckers & Mens, 2003) in particular research consistency of different versions of the same model. In addition, the aspects and differences of horizontal, evolution and vertical consistency are elaborated.

2.2.2 Adaption of QDA Methods for Requirements Engineering

The application of QDA methods to RE is a relatively new approach, and there has not been much research up to now. Yet, a few exceptions are to be associated:

(Würfel, Lutz & Diehl, 2015) propose a holistic approach for data elicitation, data analysis and the deduction of requirements. Similarly to the approach that is proposed in this thesis, they define two process phases: In the first one, GTM methods are employed for data elicitation and analysis, and a second phase aims at transferring the domain descriptions into use cases.

(Kaufmann & Riehle, 2015) focus on the suitability of QDA methods to improve traceability and conclude that the processes of RE and theory building in QDA are

similar and that the adaption of GTM concepts is a suitable method to improve the early phases of software development, particularly in respect of traceability and change management.

(Coleman & O'Connor, 2007) use GTM for the elicitation of requirement analysis in context of software process improvement. They executed an extensive exploratory case study and assess GTM to provide significant advantages in terms of effectiveness and reliability.

2.3 Research Question

This research will address the following questions:

Is a QDA-based approach suitable to provide improvements in terms of systematics, rigorousness, traceability and quality when being applied to requirements engineering purposes via the introduction of an additional artifact, the so-called code system?

Can such a code system be structured in a way that allows a clearly arranged representation of the gathered knowledge, and its direct mapping into output results, like a requirements specification and a domain model, at the same time? Is it possible to automate or semi-automate the direct deduction of the coherently resulting artifacts, specifically domain models and requirement specifications?

How do social scientists perform theory building when applying QDA methods? How is the coding process being performed?

2.4 Research Approach

We adopted concepts from conventional Grounded Theory methodology (GTM) from social sciences to develop a new approach for domain analysis, which aims at the deduction of multiple result artifacts. The new approach was carried out within an exploratory project, which had the following goals:

- to develop a theory on how Social Science Researchers perform theory building when applying QDA methods, specifically on the coding process. The representation of the domain analysis included the illustration of certain views and aspects by the use of domain models (DMs).
- to develop a requirements specification document for the QDACity project.

This thesis proposes an iteratively executed analysis of the domain and the development of the corresponding target artifacts in parallel. The following chapter

introduces the iterative data elicitation and analysis process. Chapter *2.4.2 The Code System and the Extraction of the Resulting Artifacts* will explain the representation of the gathered information within the code system and how the deduction of the target artifacts was implemented.

2.4.1 Outline of the Iterative Data Elicitation and Analysis Process

This chapter is limited to providing an overview of the iterative data elicitation and analysis process steps. A detailed elaboration of our specific experiences with respect to the concrete exploratory project is provided in chapter *3.1 Elaboration of the Data Elicitation within the Exploratory Project*.

The proposed approach is capable of processing a variety of input data, which we consider as a major advantage (see also chapter *3.3.2.6 All is Data*). However, the explications in this chapter, particularly in chapter *2.4.1.1 Execution of Interview* are tailored to data elicitation by the use of expert interviews, since this is the only instrument we used within this thesis. The processing of the elicited data, described in chapter *2.4.1.2 Analysis of the Transcript* works in an analog way for other elicitation techniques.

2.4.1.1 Execution of Interview

The elicitation of domain data via expert interviews takes place in a series of four steps:

2.4.1.1.1 Selection of Interviewee Since our approach proposes an iteratively executed analysis of the target domain, it has to be decided several times which persons to interview next.

We advise to base this decision process on the principle of Theoretical Sampling, which we adapted from GTM (see also chapter *3.3.2.3 Theoretical Sampling*).

2.4.1.1.2 Preparation of Interview We strongly recommend the preparation of an interview outline in advance, as in our experience this significantly improves both the time efficiency during the interview execution and the quality of the elicited information.

The interview questions originate in the data collected up to that point of time, as *figure 2.1* outlines:

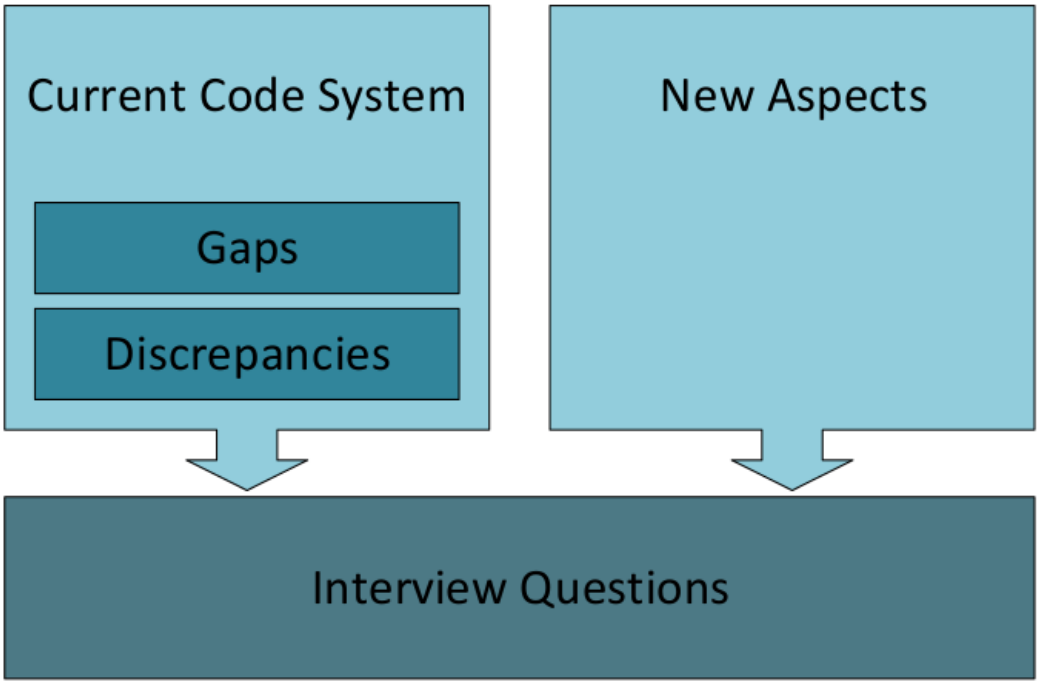


Figure 2.1: Origin of Interview Questions

- **Gaps within the current code system** were one source of questions for the next interview. When we determined open points, the lack of deeper information or untreated aspects within a certain topic during the analysis of current data, this triggered additional investigation in the form of questions in the next interview.
- **Discrepancies within the current code system** typically were originated in contradictions in the statements of different interviewees. Therefore, these aspects were detailed in upcoming interviews and more data was collected, which either confirmed or contrasted the previous statements.
- **New Aspects** In many cases, we discovered new topics and aspects of the target domain during the analysis that have not all been treated up to then. Hence, they were added to the investigation and treated in following interviews.

2.4.1.1.3 Interview We propose to execute interviews as personal interviews if possible, since they provide a higher degree of interaction in our experience. Yet, telephone interviews are also an adequate instrument.

We advertise semi-structured interviews. Open questions create a loose atmo-

sphere and encourage the interviewee to talk about what comes to their mind and also change the topic if they want to.

The interview is audio-recorded and the record serves as input material for the transcription step.

2.4.1.1.4 Transcription of the Audio Record In a next step, the audio records are transformed into transcriptions, which simply represent a written copy of the audio record.

2.4.1.2 Analysis of the Transcript

The generated domain data is analyzed in the next step.

This section confines itself to ranking the analysis step within the iterative process. Yet, it does not provide a detailed description how the code system was designed to embed the information for conceptual modeling in an adequate way. This is provided in chapter *2.4.2 The Code System and the Extraction of the Resulting Artifacts*.

As our approach is based on qualitative data analysis, executing analysis means the processing of the transcript with a CAQDAS. We used MaxQDA within our research, which is one of the standard tools on the market.

The domain data is analyzed by executing a so-called coding process. The basic concepts of this process were adapted from GTM, they are described in chapter *3.3.2.1 The Coding Process*.

The interview is split up into small units of meaning, or statements, respectively. Each statement is applied with a a so-called coding, which represents its content by an outline term.

At this step, the statements from the interviews which were executed in German language (see also chapter *3.1.4 Transcription of the Audio Record*) were assigned with English codings. Hence, all further analysis artifacts, like codes, code system, memos and also the target artifacts like DMs were consistently implemented in English language.

These codings are related to a so-called code, which is an entity within the code system. A code can be related to zero, one or multiple codings; in some cases, codes without associated codings are meaningful (see also section *2.4.2.3.1 Introduction of the Implementation*). A coding represents an instance of a code and is always related to a code. The codes were ordered in a code system and given a hierarchical and logical structure.

The interconnections between the concepts coding, code and code system are illustrated in *figure 2.8*.

The result is a code-system, which contains all relevant statements from the elicited data artifacts, ordered in a logical way, with the hierarchy implementing an ordering into issues and sub-issues. It represents the state of knowledge that the researcher gathered, and in particular, it illustrates gaps, discrepancies and unclear issues.

2.4.1.3 Code System Revision

After running through the coding process, we have a new version of our code system, which we now revise and check for quality:

- Discovered **discrepancies** trigger the further investigation of the related topic in the next interview
- **unification of double occurrences** of codes
- **consistency check of the hierarchical structure** of the code system

Hence, the code system is smoothed and corrected. In addition, the memos are updated afterwards. Arisen thoughts and questions are denoted and often serve as the basis for discussions within the next interview.

2.4.2 The Code System and the Extraction of the Resulting Artifacts

It is possible to design and structure the code system in multiple ways, and there is a multitude of methodologies in social science, which each define a process how to analyze the input data and how to develop the code system. Grounded Theory, which is the methodology that our approach is deducted from, is one of them.

Up to here, our approach is very similar to GTM: Its iterative character, the layout of elicitation techniques, the coding process with open, axial and selective coding, and the concepts of theoretical sampling, constant comparison and theoretical saturation are all applied within our approach in a very similar way.

The design and structure of the code system is where our approach differs significantly from GTM.

All social science methodologies do have in common that they aim at the development of a theory which answers the research question of the specific research project. Our approach, however, is designed for software engineering purposes

and aims at the analysis of a target domain and at conceptual modeling techniques to describe this domain.

Hence, to fulfill this different purposes, our code system, as the central artifact of the approach, needs to be implemented in a different way.

One of the goals of this thesis is to design the code system in a way that is capable of fulfilling the following purposes:

- to represent a breakdown of the state of knowledge in a clearly arranged way
- the automatic or semi-automatic deduction of one or multiple conceptual models from the code system
- to implement a structure that features the generation of glossary entries for a corresponding code and the automatic consolidation into a central glossary
- to derive a natural-language requirements specification out of the code system, and to establish a back- and forwards traceability between the inputs and the corresponding requirements within the specification

The following chapters will describe in detail how these tasks were solved.

2.4.2.1 Representation of Domain Models and Glossary in the Code System

This chapter is focused on the question how to embed the elicited information within the code system in a way that allows to derive a domain model and a glossary in an automated way.

Up to now there is no software tool on the market which is optimized for our purposes, since the idea to use QDA for RE is relatively new. QDAcity's goal is to fill this gap in near future. At present, we decided to go with MaxQDA as one of the market leaders of CAQDAS systems.

MaxQDA is a sophisticated tool for conventional QDA, but its features are not optimized for our purposes. There is no designated area which is intended to serve for the embedding of additional information atop of codings, codes, a hierarchically structured code system and "conventional" memos, being used for the denotation of thoughts and questions.

Thus, the best option to encode additional information was to modify the use of memos. They provide a free text field where it is possible to store bigger amounts of text, and each memo is directly associated with a code, which facilitates the mapping of an object-oriented structure a lot.

We decided to use the memos for embedding both the information for domain modeling and for a glossary. Since memos can only store plain text, we needed to develop a structure which enables us to decide between DM-information and glossary information, and to represent these informations in a way that makes an automatic or semi-automatic deduction of the target artifacts DM and glossary feasible.

Figure 2.2 gives an overview of the structure we developed. It consists of three major sections:

1. a **switch section** which contains two binary deciders if the memo, and the corresponding code, respectively, will be included in the glossary and in the DMs.

   These two switches are mandatory if the information within the memo is supposed to be used for conceptual modeling. If a memo does not start with the switch section, it is ignored by the modeler.

   This is necessary, since not all codes from the code system are relevant; not all codes contain object-oriented information. In fact, the majority contains either meta-information (which is subsumed in the superordinate glossary-entry), or information which is irrelevant for the deduction of DM and glossary. Nevertheless, these codes are kept in the code system and supplement the data which is collected to the superordinate concept.

2. The **Glossary Section** is an optional section. It is included when the glossary switch is turned to 'yes' and includes a description of the concept, which also may originate in the meta-information from the subordinate codes.

3. The **DM Section** contains all information which is connected to the representation of one or multiple DMs.

   Within this section, it is possible to store information for multiple DMs. This is reasonable, because we want to encode the information for multiple DMs in one and the same code system. The concept which the memo corresponds to may play a role in more than one model.

   The implementation is realized by encapsulating each DM that the concept is related to with <DM> and </DM> -brackets. Each DM-entry is mandatory to be started with two variables that define the ID of the DM which the following information refers to and the type of that DM:

   - **DM_ID** is a simple integer that is unique for each DM.
   - **DM_Type** defines the type of the DM that is identified by the DM_ID. It is necessary to represent this information in each corresponding

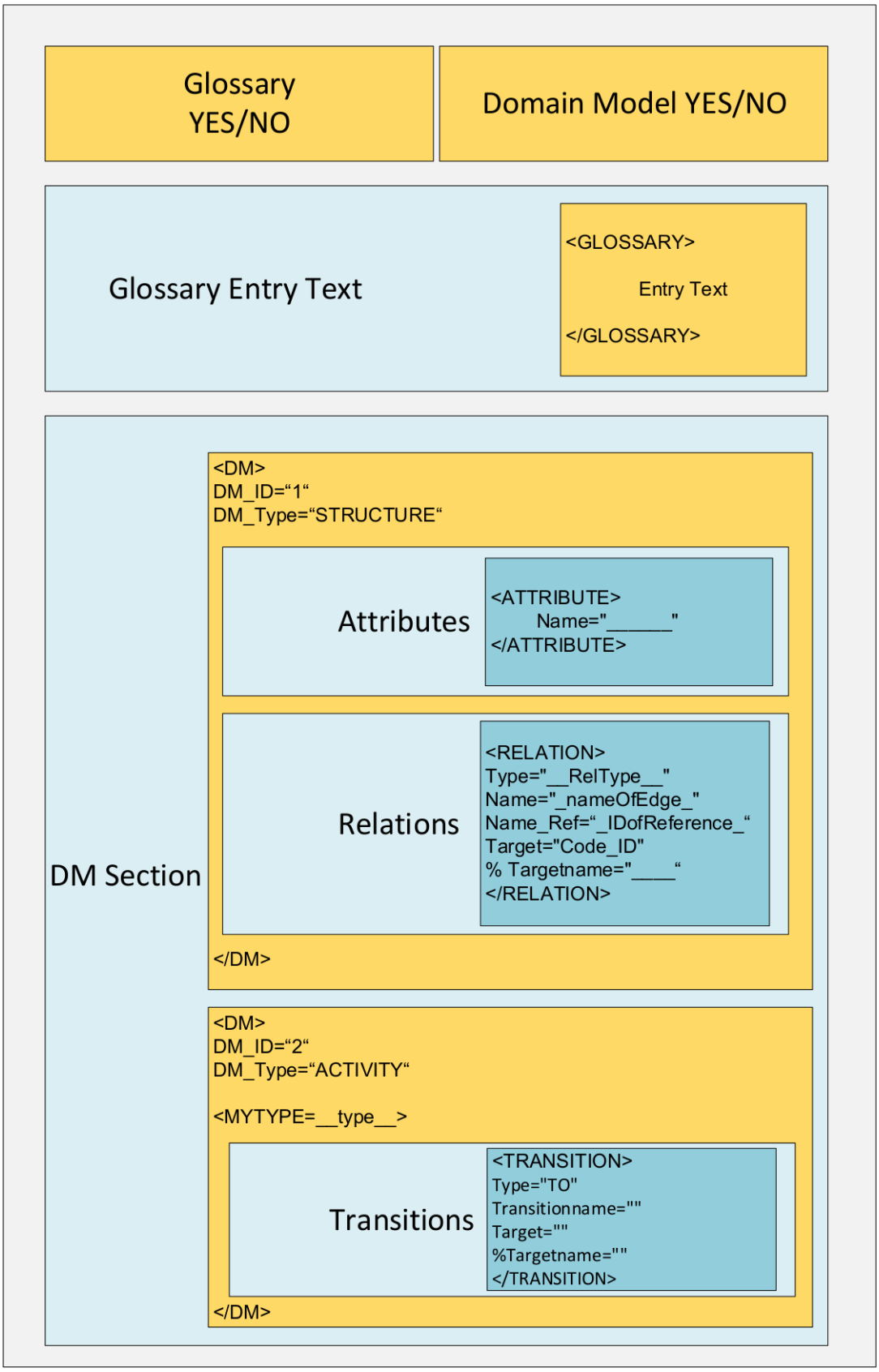


Figure 2.2: The Design of Memos for Conceptual Modeling

memo, since the target DM is not stored anywhere centrally, but is composed out of all related memo entries.

Within this thesis, we implemented **two simple DM types**:

- simplified class diagrams as **structural DMs**, for which the DM_Type is "STRUCTURE"
- simplified activity diagrams as **behavioral DMs**, for which the DM_Type is "ACTIVITY"

These two DM types are introduced in the following two sections. If DM information for more than one DM is encoded in one memo, either multiple DMs of the same type or different types, each part is separated by the encapsulation with <DM> and </DM>.

2.4.2.1.1 Structural Domain Model *Figure 2.2* provides an outline of the structure of an DM-section for a structural DM in the upper part of the DM section of the figure.

As DM-sections of all types, the section is embraced by <DM> and </DM>, followed by two identifiers, which are explained in section *2.4.2.1 Representation of Domain Models and Glossary.*

There are two sections:

1. The **attribute section** contains the attributes of the code. Each attribute is stored as <ATTRIBUTE>Name=" "</ATTRIBUTE>. We decided that for this approach the only relevant information to be stored in an attribute is its name and that all additional attribute-values defined in UML (initial value, property value and assurances) will be ignored.
2. Each relation within the **relations section** is encapsulated again with <RELATION> and </RELATION>.

   It contains five entries:

   - **Type** defines the type of relation. Applicable values were decided to be association, inheritance and realization. We decided to go with this small subset of types, since implementing some of the further possible ones would have required more complicated structures. Furthermore, we decided that as a proof of concept these would be sufficient.
   - **Name**: The name of the relation, respectively the inscription of the edge to the target concept.
   - **Name_Ref**: The ID of the code that the naming of the edge originates in. The ID is taken from the exported XML file as described in section

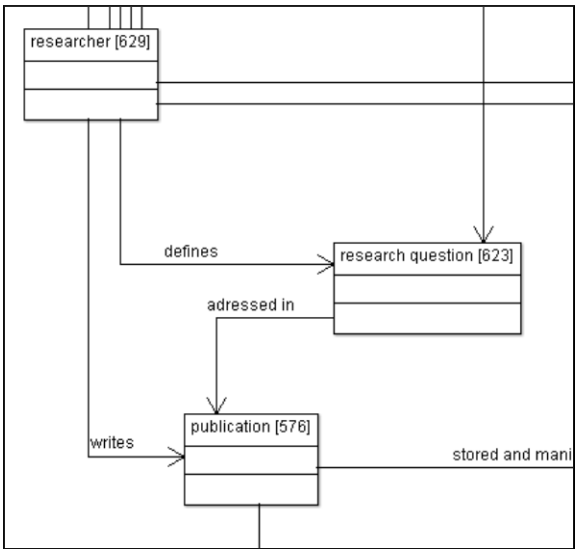


Figure 2.3: An exemplary excerpt from the resulting structural model

2.4.2.2 The Mapping of Memos into Artifacts and the MaxQDA File System.

- **Target** The code-ID of the target code. This ID is taken from the exported XML file as described in section *2.4.2.2 The Mapping of Memos into Artifacts and the MaxQDA File System.*
- **Target-Name** includes the name of the target code. We decided to include it, although it is not used for the parsing, since the name of a code could be changed but its ID stays the same. This entry solely serves to facilitate the analyst's work and is more a comment, which we want to emphasize with the "%"-sign.

We consider this design to represent a compromise between usability for the analyst and parse-ability for the software which maps the memos into the DM. Figure *2.3* provides an exemplary excerpt from the DM we developed. The complete DM can be found as figure *3.2* in the appendix of this thesis .

2.4.2.1.2 Activity Domain Model Figure 2.2 provides an outline of the structure of an DM-section for a behavioral DM in the lower part of the DM section of the figure.

We implemented our behavioral model as a small subset of UML 2.3 activity diagrams. The reasons for limiting the functionality are the same as for structural diagrams.

As DM-sections of all types, the section is embraced by <DM> and </DM>,

followed by two identifiers, which are explained in section *2.4.2.1 Representation of Domain Models and Glossary*.

The identifiers are mandatory to be followed by the variable **MYTYPE** that has to be set once for each activity-DM-memo. It defines whether the code corresponding to the memo is represented as an **activity state** (<MYTYPE= ACTIVITYSTATE>) or as a **decider** (<MYTYPE=DECIDER>). Only these two options are valid.

The MYTYPE-identifier is followed by zero, one or multiple transition-entries. Each transition within the **transitions section** is encapsulated with <TRANSITION> and </TRANSITION> and contains four entries:

- **Type** defines whether this transition goes to the target activity state or if it comes from there. The default case is that it goes out, there is one and only one exception, which is a start node. These are not represented by a code in our implementation, therefore it is necessary to define these transitions within the target activity state.
- **Transitionname** is an optional entry that can assign the transition edge in the model with a name.
- **Target** is the code-ID of the target code. This ID is taken from the exported XML file as described in section *2.4.2.2 The Mapping of Memos into Artifacts and the MaxQDA File System*.
- **Targetname** includes the name of the target code. Corresponding to structural DMs, we decided to include it, although it is not necessary; it serves to facilitate the analyst's work and is more a comment, which we want to emphasize with the "%"-sign.

We implemented an activity diagram within our code system which outlines the phases of a social science research project. It can be found in *figure 2.4*.

2.4.2.1.3 The Traffic Light System for the Memos To ease clarity within the code system, we established a traffic light system for the memos. MaxQDA features to assign multiple colors and symbols to a memo. Figure 2.5 illustrates how we decided to use them:

1. **conceptual modeling memos** were marked with four colors:
    - **red, yellow and green** indicate the status of memos: Green memos contain conceptual modeling information and are stated to be completely implemented. Yellow ones are not finished yet and are under development, and red ones are those who are evaluated to be relevant for conceptual modeling, but not yet implemented.

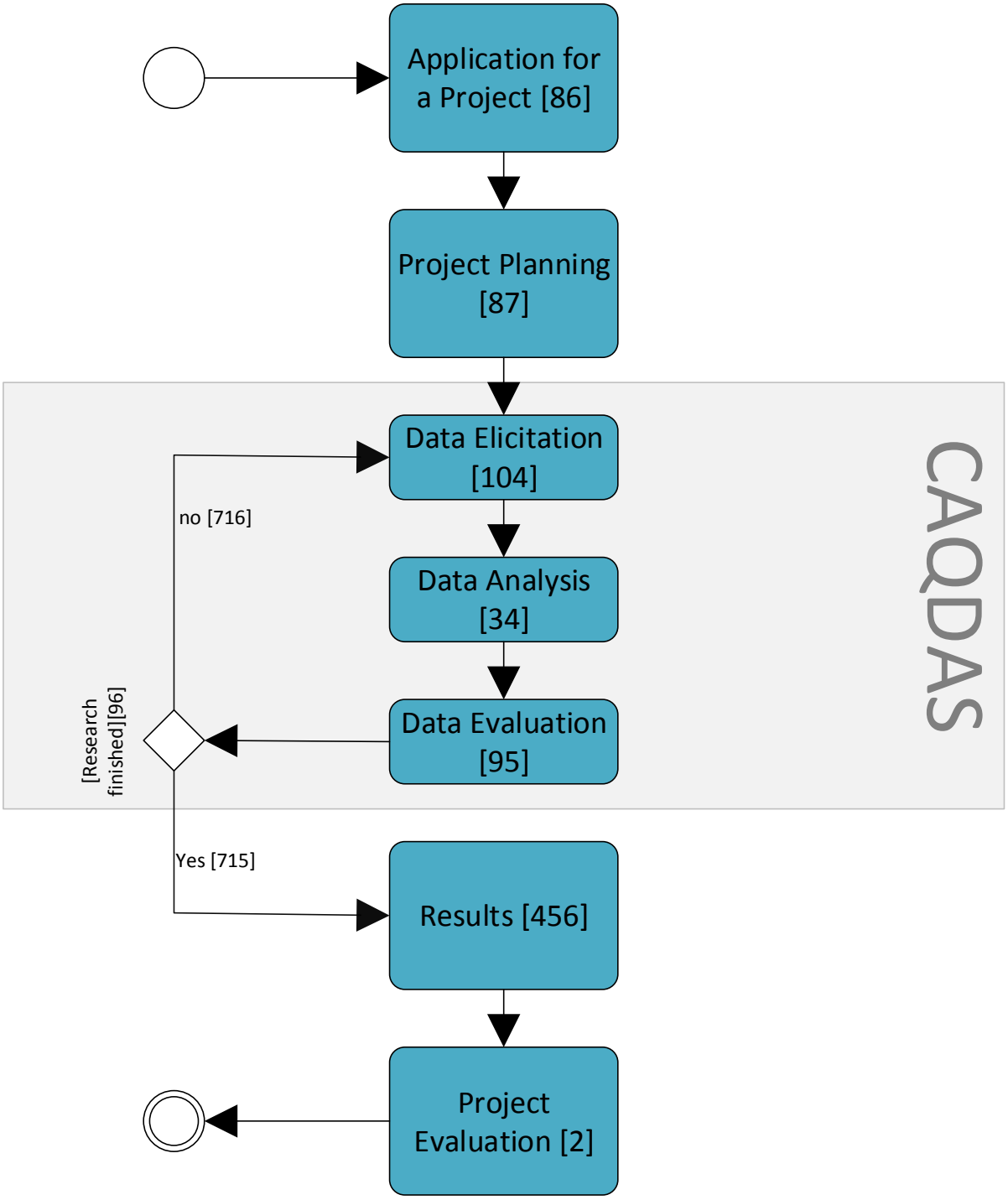


Figure 2.4: Outline of a Social Science Research Project, represented as a simple activity diagram

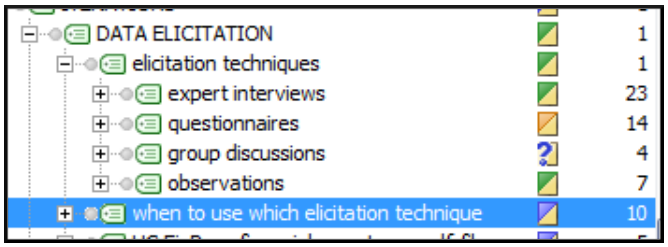


Figure 2.5: Example for the Traffic Light System for the Memos

- **blue** memos indicate codes who are evaluated to be irrelevant for conceptual modeling. They were solely instanced to support a better overview and do not contain any information at all.

2. **conventional memos** were used in parallel in a conventional way, the researcher denoted thoughts, questions and tasks here. They were marked with a question mark.

   Conventional memos proved to be a valuable help during the whole analysis process, since it was possible to get thinkings denoted there without the necessity to do this very structured or in mature language. In addition, the direct interconnection from the memo to the coding and its placement within the code system was often very helpful.

2.4.2.2 The Mapping of Memos into Artifacts and the MaxQDA File System

Figure 2.6 gives an outline of the mapping process and the affected artifacts:

1. From MaxQDA, you can export the constituent parts of the MaxQDA project as XML files. What you receive then is a folder which consists of three parts:

   - A sub-folder that includes all the documents which you imported into the MaxQDA project. The data format depends on the format of the files when they were imported: .rtf-files if you imported Microsoft Word documents or .pdfs if you imported pdfs.
   - A sub-folder with all memos from the code system in it, with one .rtf-file for each memo.
   - An XML-file, named like the project, which consists of three sections:
     - a "codings"-section for each imported document. It includes a description for each coding with the definition which text has been coded and its assignment to a code.

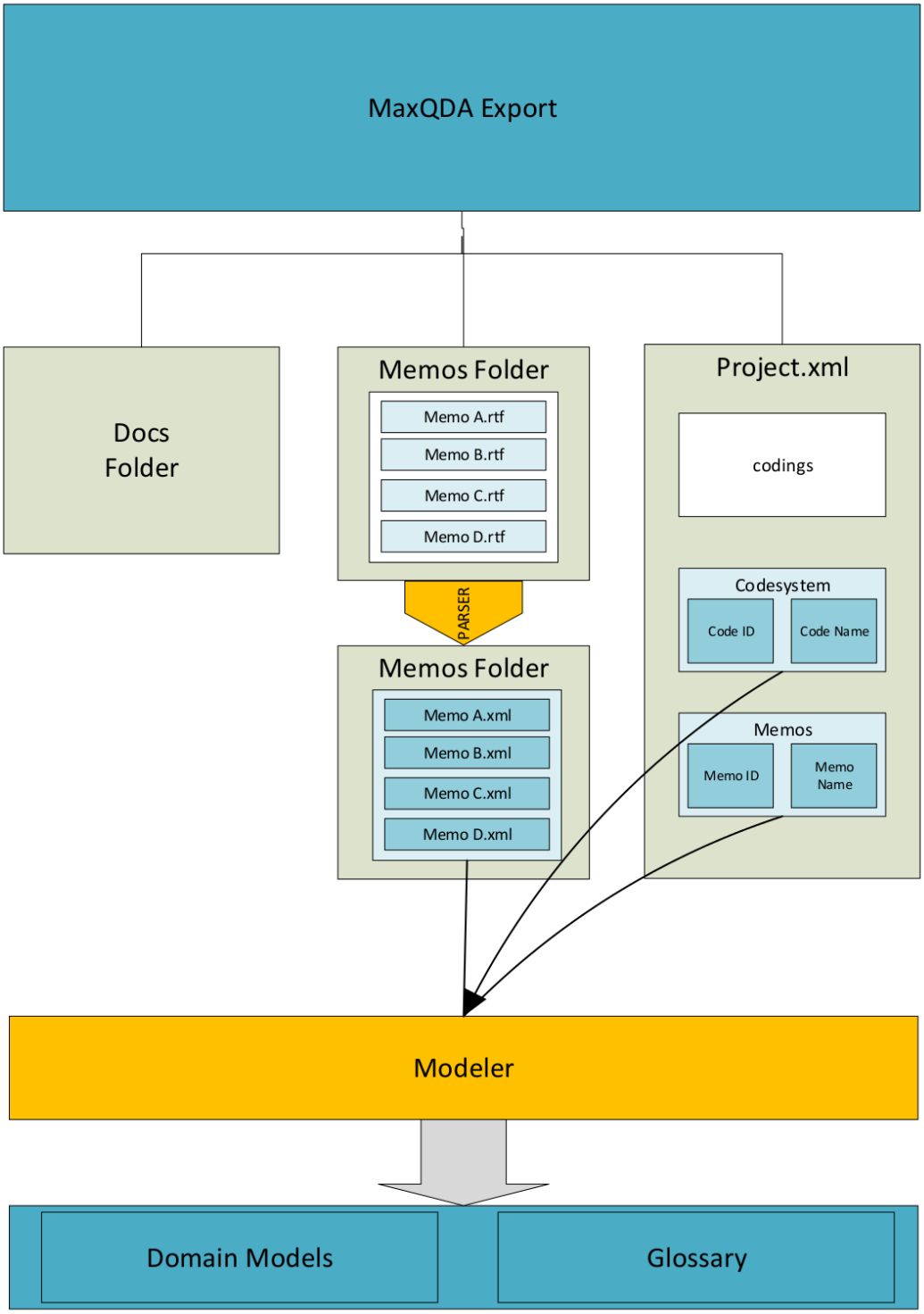


Figure 2.6: The mapping process from MaxQDA to the Domain Model

- a "codesystem"-section. Each code definition includes an ID, the name of the code as a string, its color, the author who made the code and a time stamp when it was made.
- a "memos"-section which defines each memo by an ID, its title as a string, the author of the memo, a time stamp when it was made, and a relative path to the .rtf-file where the memo is saved.

2. The memos, which are stored as .rtf-files in MaxQDA, need to be transformed into XML-files, since .rtf-files cannot serve as input for a model generator. This can be done with the help of a tool that was already developed for related research, which can be found at `https://github.com/maclomork/qda-parser`
3. The relevant pieces of information (also highlighted in *figure 2.6*) are the memos from the memo-folder, and two types of relations, which can be extracted from the XML-file: the connection from code-ID to code, and the connection from a memo to the particular .rtf-file where it is stored.

   With the help of this information, it is feasible to derive a domain model.

   Unfortunately, it was not possible within the scope of this thesis to program a modeling tool which automates the model generation. Nevertheless, we believe this should be unproblematic.

2.4.2.3 Deduction of a Requirements Specification

Besides establishing processes for the generation of domain models and a glossary, this thesis aims the implementation of a process that supports the deduction of a requirements specification out of the code system.

It is necessary to tread another path for this third artifact. The following chapter will introduce the reader to our approach of specification deduction. The rationale for this implementation will follow in chapter *2.4.2.3.2 Rationale of the Implementation*.

2.4.2.3.1 Introduction of the Implementation The basic idea of our approach is to use MaxQDA not only for the processing of the input data and the management of codes, but also as the tool where the specification is written. Text sections from the specification can then be assigned as codings of the codes which they relate to. This principle offers significant benefits, since it features a very high grade of traceability in combination with good usability.

An additional basic design principle is to create an extra section within the code system, where **structural codes** are inserted, which are named and ordered in a

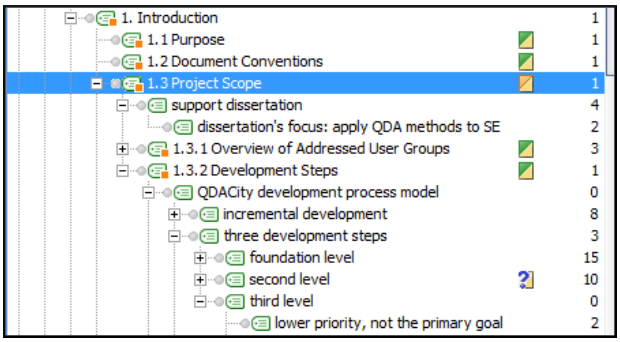


Figure 2.7: Example for structural codes within the code system

hierarchical structure corresponding to the chapter structure of the specification. These structural codes are flagged as such by assigning them with a different, contrasting code color.

Figure 2.7 shows an excerpt of the code system with these codes: The orange codes are structural codes. They are named like the specification chapter which they correspond to.

These structural codes have multiple tasks:

- It is possible to subsume all codes that are clearly related to this topic under the structural code and therefore get a **clear arrangement** of topics and aspects, corresponding to the specification's outline.

  However, **flexibility** is offered: It is not mandatory to arrange all codes within this structure. All codes which cannot be clearly assigned to one particular specification chapter, or relate to multiple chapters, are simply further kept outside of this structural code hierarchy. They are still represented the same way as they were after the coding process: Hierarchically and logically structured.

  In *figure 2.8*, these two sections within the code system are shown.

- Each structural code is assigned with exactly one coding, which is the caption of the corresponding chapter in the specification. *Figure 2.8* illustrates this with the black connections between the specification and the code system. This eases the jumping to this section within the specification, and it offers an **implicit structural error checking**: Each structural code has to have exactly one coding, and each headline within the specification is coded with exactly one orange coding.

- Similar to the DM-related memos described in chapter *2.4.2.1.3 The Traffic Light System for the Memos*, each structural code is assigned with a **memo** that is marked with a **traffic light color**, which expresses the status of the

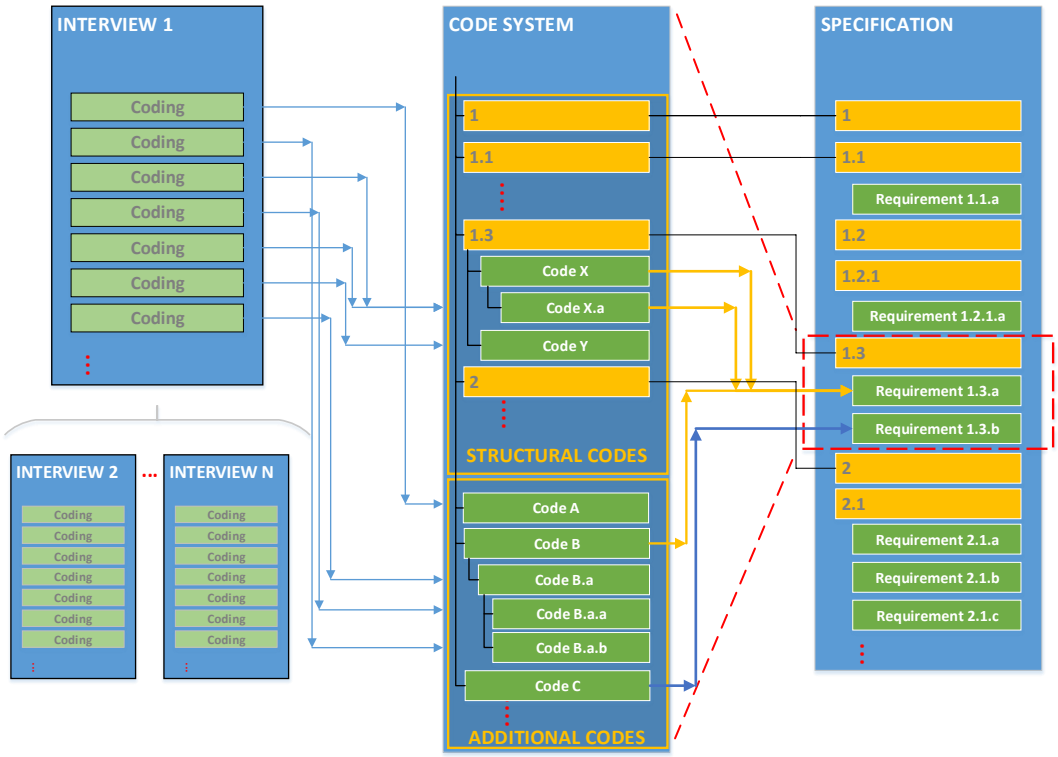


Figure 2.8: Outline of the Traceability between Codings, Codes and Requirements

code section and the corresponding specification chapter:

- *red*: huge gaps of information, lots of data missing.
- *yellow*: contradictions within related codes, information gaps, open tasks for this chapter
- *green*: chapter is complete, no questions, requirements are written down, tasks are done

The researcher denotes the specifics of the gaps, problems, contradictions, thoughts or questions as the content of these memos.

Hence, they provide both a quick overview of status and a flexible way to denote issues.

These memos can never collide with the memos which are used for conceptual modeling purposes, since they are strictly assigned to the structural codes, which are artificially added for their particular purpose.

In total, this approach design supports the development of the specification in parallel to the iterative data elicitation and analysis process:

- It is possible to adapt the structure of the specification and in the corresponding code structure. Yet, this has to be done manually (see also chapter *3.2 Limitations of the Approach in Practical Execution*).
- A statement or requirement from the specification can also reference codes which are not arranged into the section of the corresponding structural code; and by the coding within the specification, a direct link to this code is established and it can easily be found.

2.4.2.3.2 Rationale of the Implementation The generation of a specification is a process significantly different to the deriving of DMs or a glossary:

The generation of DMs and glossary is about directly modeling the concepts of the codes; the information to be encoded in the code system can be embedded in correlating memos. The single information units can be either mapped 1:1 (like i.e. edges within a DM) or summed up into superordinate units (like glossary entries, which typically contain the meta-information from subordinate codes).

The generation of a specification requires a different view onto the code system. The formulation and structuring of requirements is a complex intellectual procedure of the analyst. It includes activities like i.e. the finding of unambiguous wording or the fragmentation and transformation of domain information into functional and non-functional requirements.

From our point of view this process cannot be automated or semi-automated, since it requires a high level of skill and experience. We think that even if an implementation of structures that aim at a semi-automation would be attempted, this would result in a highly-complex, very formal and inflexible framework that accounts for very poor usability and deficient results.

Our approach aims at supporting the analyst in his work and promote efficiency, quality and usability:

- The clearly arranged state of knowledge, and the direct linking of all information to a certain aspect help the analyst to include all relevant pieces of information and to not miss aspects.
- The explicit indication of gaps and contradictions ensures efficiency in domain analysis.
- It is easily possible to arrange newly gained additional information into the present code system.
- It provides a high degree of traceability:
  - **backwards:** Requirements can be traced back to codes, codes can be traced back to codings, codings can be evaluated in context.
  - **forwards:** The analyst can follow a coding to the correspondent code and can retrace which passages of the specification are linked with this code, the superordinate code, or codes that are also subordinated to this code.
- Therefore, it also provides significant advantages in terms of maintainability and the facilitation of change management, which we consider a major plus, particularly for capacious software development projects.

Nevertheless, the use of a CAQDAS for writing a specification does also come with several shortcomings in practical execution, since it is not targeted on this purpose. We described our experiences in chapter *3.2 Limitations of the Approach in Practical Execution.*

We assess the benefits of our approach to clearly prevail these shortcomings. Furthermore, we expect these practical problems to significantly decline when the approach is executed with a tool which is designed to meet the needs of conceptual modeling.

2.5 Used Data Sources

The major data source for this thesis was the execution of a series of expert interviews. In total, five interviews with five stakeholders were carried out.

These stakeholders can be divided up into two groups:

Four of them are professional researchers from social sciences. The goal of these interviews was to generate a theory on how Social Science researchers perform theory building, with a focus on Qualitative Data analysis and specifically on how coding is performed there. These interviewees are considered to be stakeholders in the QDACity project since they represent one of the major target groups of QDACity. Hence, these interviews served both the purposes of collecting profound knowledge about how Social Scientists work and take their needs and requirements into account when developing a specification for QDACity.

Three of the four researchers we talked to do conventional social science research and use qualitative data analysis methods. All of them are experienced researchers and employed QDA methods in multiple research projects. One of them holds a doctor's degree, the other two are PhD students and hold master's degrees or equivalent titles.

The fourth researcher from this first group is appendant to a university chair for economic research. During her dissertation, she extensively employed MaxQDA. Her experiences were of interest for us, since her use of a CAQDAS did not aim at deriving a theory, but rather categorize and structure the content of financial reports.

The second stakeholder group consists of one expert, the lead developer of QDACity. As one of the goals of this thesis was the creation of a specification for QDACity, we considered it crucial to involve this stakeholder and executed an interview with him. This interview aimed at gathering information about the QDACity project's specifics, like i.e. its scope or constraints.

The fact of only having one person in this group of stakeholders is founded in the characteristics of the QDACity project: It is basically a one-man-project, which itself is part of an academic research project: It is the core aspect of a dissertation. The PhD student is basically responsible for design decisions, therefore this person represented the most adequate interview partner to research the project and derive a requirements specification. The professor who is the supervisor of the dissertation would have represented an important additional information source, however we had to resign that because of time limitations.

Furthermore, we would have liked to execute more interviews with the lead developer, since lots of aspects could not be adequately covered in the one interview.

However, this was not possible any more because of time limitations.

2.6 Research Results

2.6.1 Abstraction Levels

The proposed method is capable to process information on all levels of abstraction, and it facilitates its hierarchical and logical ordering. Umbrella terms will be found on a high level of the code system, since they represent an abstract concept. Details to concepts will be stated on the subordinate levels of the code system.

Typically, the codes that were mapped into DM-concepts could be found on the middle levels of hierarchy. Higher levels represent abstract, superordinate core-concepts, whereas the lowest levels mostly detailed concepts of the middle layer.

2.6.2 The Degree of Freedom in Coding

We think that our method gives the analyst a high latitude how to develop the code system, since it provides procedures for conceptual codes, but also codes which represent meta-information. We consider this a significant advantage of our method, since hereby also non-technical aspects like i.e. social considerations can be constantly recorded and documented. Moreover, alternative development paths are documented throughout the project, plus the rationale not to follow them.

2.6.3 Views and Perspectives

When we started the domain analysis process, we let the code system structure emerge out of the data. Hence, we initially kept very close to the concepts of Grounded Theory.

However, during the first iterations' axial and selective coding steps, the code system developed into a direction that might be considered to be the expected outcome of a conventional analysis as well: The structure split up into separate parts for the social science domain analysis and for the requirements analysis of QDAcity. Issues like i.e. the project steps for the former and development constraints for the latter emerged out of the data as core concepts and hence became superordinate codes in the code system. This may partly result from the interview outlines of the initial interviews, since they addressed questions

like "What project phases are there in a research project when you apply QDA methods?", simply because it made sense.

As the concepts of our approach were partly developed during the practical execution of the exploratory project, there was a point in development where we decided to introduce the structural codes for supporting a mapping to the structure of a requirements specification (see also section *2.4.2.3.1 Introduction of the Implementation*). Consequently, the RA-related part of the code-system was resorted and arranged subordinate to the structural codes, corresponding to topics and perspectives. In our experience, this step was rather easy, in fact, it was often a 1:1-mapping.

We think that this similarity between our code system's categories which developed uninfluenced development and the topical structure of the specification template we started with speaks in favor of our approach: The structure, categories and topics which are represented in requirements specification templates were established over time and out of practical experience. A mature domain analysis approach should inherently result in a similar structure.

With the experience from this project, we propose a process for the future that plans a first phase of domain analysis, where the structure of the specification is being developed out of the well-established templates, but also tailored in respect to the specific project. This phase corresponds to the phase of data analysis within our approach, where the degree of theoretical saturation is not yet very high.

Subsequently, the aspects and views of the domain are elaborated in detail. We evaluate particularly the generation of fine-grained and highly-detailed requirements to be developed most efficiently, when the iterative elicitation of additional data to complement the state of knowledge is well-tailored to the targeted results. This phase was only poorly entered in our RA of QDACity because of time restrictions.

2.6.4 Validation of Results

2.6.4.1 Domain Models and Glossary

We provided our results to the social science domain experts and asked them for feedback with a written survey. The original can be found in figure *3.3*.

Two of the experts provided feedback, which basically was very positive. One of them pointed to a missing detail, which we complemented. Beyond, they mainly agreed to our results (see also figure *2.9*).

Question	Disagree	Rather disagree	Undecided	Rather Agree	Agree	Can't be assessed
It was easy for me to understand what the artifacts were trying to model.				1	1	
The artifacts represent the domain correctly.				1		1
The artifacts are a realistic representation of the domain.				1		1
The artifacts contain contradicting elements.	1			1		
The artifacts contain the following inconsistencies or errors:	Other elicitation techniques than interviews are also prepared in advance					
The models give a complete representation of the domain (particularly structure model).				1		
The following elements are missing from the structural domain model:	Type-building content analysis from Kuckartz					

Figure 2.9: The received Feedback from the Domain Experts

2.6.4.2 Requirements Specification

A validation of the received specification was not possible within the scope of this thesis, since no QDAcity domain expert was available for assessing the result, and competing specification documents that were developed by independent researchers were not accessible for us.

However, in our opinion, the evaluation of the suitability of the proposed method is more important than the received result, which in our belief is definitely not complete and well elaborated. We ascribe that to the limited time.

Therefore, a substantial assessment of the proposed method will require its application to additional analysis projects.

2.7 Results Discussion

This thesis aims at evaluating the adaption of core concepts of QDA methodology for conceptual modeling purposes. We developed processes for the representation and semi-automatic deduction of conceptual models, and for the support of deducting a requirements specification from the domain analysis.

Basically, this research is intended to serve as a proof of concept. Hence, the practical application to an exploratory project could not be satisfactorily finalized:

- The results of our domain analysis of social science research projects with focus on QDA methods are stated to provide a significant grade of maturity. The progress of the execution of interviews clearly indicated an increasing grade of theoretical saturation, and towards the end of the series, barely new aspects were brought up any more.
- The requirements analysis of the QDAcity project, however, could not be fully applied. Only one interview with one stakeholder could be executed within the scope of the thesis. Consequently, it was not possible to deduct a complete, mature specification. In fact, we evaluate the resulting specification not to be anywhere close to high quality.

  However, we ascribe this mainly to the restricted time at hand. We are sure that the developed approach is suited to also provide high-quality results in respect of the deduction of a requirements specification when its appliance is fully performed.

  The achieving of a complete and high-quality specification was not the primary goal of this thesis, but rather to develop a methodology and a process that is capable of leading to such a specification. The goal of applying the approach to QDAcity was a proof of concept, and in our opinion the approach has proven his suitability for RE purposes.

To verify our findings, future research will have to include the appliance of our methodology to further projects. Nevertheless, we believe that the employment of a QDA-based approach represents a suitable method to introduce multiple improvements for RE purposes, since it provides a series of benefits.

To further investigate the aptitude of the proposed methodology, the following aspects will have to be implied:

- How can the operating expense be ranked, compared to conventional domain analysis?
- How much does the maturity of results differ when the proposed and the conventional methodology are directly competing?

Hence, we suggest that the proposed method should be applied to one or multiple projects in direct competition to conventional RE methods, and the operating expense and the results should be comparatively evaluated.

If this comparison should speak in favor of our method, we propose the training of conventional RE analysts how to apply our approach as a next step. In our

opinion, the question how much initial effort it is for an analyst to learn our method is an important aspect.

2.8 Conclusions

This thesis proposes an approach for requirements engineering purposes that is based on QDA principles and features the semi-automatic deduction of conceptual models out of the domain analysis data. In addition, it supports the development of requirement specifications and provides significant improvements in terms of efficiency, traceability, systematics, maintainability and quality of results.

3 Elaboration of Research

3.1 Elaboration of the Data Elicitation within the Exploratory Project

During this research, our sole instrument of data elicitation was the execution of expert interviews. They were decided to be the appropriate instrument, since the number of suitable domain experts at hand was commensurately limited, but the projected amount of information and level of detail were high.

We consider expert interviews to be time- and labor-extensive, however they provide several advantages in comparison to other data elicitation techniques: It is possible to individually fit the course of an interview and the researcher can directly ask questions at topics where he would like to go more into detail. Therefore the likeliness of misunderstandings and ambiguities is much lower than i.e. with the use of questionnaires. In addition, the very direct interaction of the interview partners result in a very high proportion of the quality of gathered information to the interview duration, which we appreciated, since the availability of the domain experts of social science was very limited. Furthermore the open character of semi-structured interviews often brought up topics of high importance which were not incorporated up to then.

3.1.1 Selection of Interviewee

Generally, the choice of suitable interviewees can be considered the most fundamental factor for a successful data elicitation by the use of interviews.

Since our approach proposes an iteratively executed analysis of the target domain, we decided several times which persons to interview next. During practical appliance of our approach to the analysis of Social Science research projects and particularly QDA, the options of interviewees were rather limited, however; finding people who were both sufficiently qualified and experienced on the one hand and willing to offer us their time on the other hand was relatively difficult.

The GTM principle of Theoretical Sampling could not fully be implemented in this research project. The main reason for this was the very limited time on hand within this thesis to execute our approach. It has to be distinguished between the two main focuses of our data elicitation: Considering the analysis of the social sciences domain, we state our results to feature a significant grade of saturation, while we assess our elicitation of requirements for QDACity and the quality of the corresponding specification as not fully conceived.

3.1.2 Preparation of Interview

To ensure an efficient interview execution and high-quality results, an interview guideline was prepared in advance for each interview, which was also mailed to the interviewees from the social scientist group in advance. They had all asked for the questions, since they wanted to make sure that they could provide information to all raised subject areas.

The interview questions originated in the data collected up to that point of time, as described in chapter *2.4.1.1.2 Preparation of Interview*.

The initial interviews were special cases. We consider two interviews to be initial ones, as the interviews from the two groups of stakeholders treated topics that hardly had intersection sets.

The initial interview with a social science researcher, which was also the first one from the series in total, mainly served to get an overview about the domain of social science in general. As we were no experts in that domain, it was crucial to get an overview how social science researchers work, what their daily routine is about and how they approach research projects. Previous literature research gave us a basic idea, but the personal talking to our first interviewee conveyed a profound understanding of the domain to us and helped a lot with deciding which topics to focus on and where a basic understanding would be sufficient.

The interview with the QDACity lead developer was characterized slightly different. The priority of this interview was clearly to support the development of a requirements specification. As the structure of such a specification is clearly advised to be closely oriented to the IEEE template or one of the other accredited SRS templates, the topic blocks to be covered can be seen as more or less given (please find details in chapter *2.4.2 Extraction of the Resulting Artifacts*). Hence, the task of this interview was more the definition of project specifics and the first iteration of the elicitation of data: For instance, it was found that QDACity is planned to work as a cloud application, which makes the topic of hardware constraints significantly less important than this would be the case i.e. at developing software for an embedded system. Therefore, this topic was of low importance

during the elicitation of data and definition of requirements and resulted in a subordinate representation in the specification.

Unfortunately, it was not possible to append additional interviews within the scope of this thesis. Therefore, the requirement analysis of QDAcity could not be completed.

In conclusion, we state that the initial interviews do also contribute precise, detailed and valuable information to multiple topics and their aspects, but their clear priority is to structure the research domain, delineate its scope and prioritize data elicitation tasks.

3.1.3 Interview

The interviews were carried out as telephone interviews in two cases and personal interviews in three cases. Three of the interviews were executed in English language and two in German, since the interviewees felt more comfortable and easier to speak freely about the topics.

In all cases, the interviews were audio-recorded. The record served as initial material for the following analysis process.

All interviews were executed in a semi-structured way: The intention was to let the interviewees speak freely, which is also transferred from the traditional GT techniques. This shall ease the discovery of topics that the analyst might not yet be aware of. However, when statements came up that the analyst did not understand or included unclear details, or when the current interviewee contradicted statements from earlier interviews, the analyst inquired these points and asked for more details.

The prepared questions served as an outline. They gave the interviewer an overview which topics he wanted to announce within the specific interview and therefore helped not to forget to address important issues. Yet, mostly not all questions were actually used in the interview. Sometimes it turned out that the interviewee could not give continuative information to an aspect and it was deferred. In other cases, certain aspects of interest were inherently treated by the interviewee when talking about related topics.

3.1.4 Transcription of the Audio Record

In a next step, the audio records were transformed into transcriptions.

This was done manually by ourselves in all cases. We used Winamp as one of the common media players in combination with the add-on Pacemaker, which

enabled the deceleration of the audio record. For the text processing, Microsoft Word was used.

In comparison to conventional GTM instructions, we slightly simplified our transcription process in the following way: Speech parts of both interviewer and interviewee were transcribed word for word, however we resigned to include details as the accentuation of statements or the length of breaks, because we consider them as not important for the purpose of our research. Furthermore, we left out speech parts that only stated an expression of comprehension, since they do not include any information, but would interrupt the statements' unity and make the text more difficult to read unnecessarily. In addition, in some cases we decided to directly skip short passages in statements where the speaker misspoke and corrected himself, and some expressions of colloquial speech or dialect were transformed into equivalent standard language expressions without changing the meaning of the specific statement.

English interviews were transformed into English transcripts and German interviews were transformed into German transcripts.

3.2 Limitations of the Approach in Practical Execution

The proposed approach uses MaxQDA for the data elicitation and analysis, but also for the embedding of information structures for conceptual modeling and the deduction of a requirements specification. MaxQDA represents a mature tool for the former activities, yet executing the ladder ones came with several shortcomings and challenges, since MaxQDA is not planned to be used for these purposes.

3.2.1 Conceptual Modeling

This section describes the challenges we faced during the implementation of domain models and a glossary. Both activities are focused on embedding information units within the memos, which are later exported, readout and assembled to the target artifacts.

During the step of transforming the elicited data into the memo structure, we faced the following challenges:

- The core activity of DM development is to implement interconnections between concepts, like i.e. associations. Therefore, it is crucial to reference each concept with a unique identifier. In MaxQDA, the only unique

identifier is a code ID, which is assigned to each code when it is generated; the name of the code can be changed (which is reasonable for QDA). Yet, this ID is not accessible directly in MaxQDA, but has to be picked over the detour of a XML export as described in section *2.4.2.2 The Mapping of Memos into Artifacts and the MaxQDA File System*. In addition, the XML-file where the code-IDs can be looked up needs to be re-exported each time you want to comprise codes that were newly inserted since the last export.

This makes the definition of interconnections very complicated, laborious and error-prone.

- Memos can not be opened as part of the GUI (like i.e. the summary of codings related to a certain code), but only in front of it in an extra window. This often handicaps practical work, since this extra window obscures the information you want to model within the memo.
- It is not possible to define more than one memo per code. This, however, would be useful, since you could use conventional memos in parallel to conceptual memos. Furthermore you could also define multiple memos for each aspect, i.e. one for the glossary entry and one for DM modeling, which would offer more clarity.
- Some concepts have to be represented at more than one place in the code system, since they play an important role in multiple views or in several parts of a domain. Such situations require the possibility to link the instances of such concepts, which would provide uniqueness. Since there is no such possibility up to now, we had to keep some concepts more than once within the code system. However, when executing conceptual modeling, all these instances have to be taken into account. We think that the inevitable redundance induces a high likeliness of errors.
- We decided to implement only the outgoing edges of a concept within the corresponding memo to avoid redundancy. We think that this provides significant advantages in comparison to implementing each edge within both affected concepts, yet it is difficult to keep an overview.
- In general, keeping an overview is really difficult: You have to open each memo to see what information is already embedded in it. When executing changes, this has to be done highly-concentrated and with caution, since there is no possibility to undo changes. In addition, such changes have to be done manually in a rather complicated way: You need to look up code-IDs in an external file and maybe do an export first. We experienced this process as very laborious and error-prone.

3.2.2 Development of Requirements Specification

In respect of the development of a requirements specification, our approach is based on the basic idea of writing the specification directly in MaxQDA. MaxQDA includes a feature to modify the content of imported documents (if they were imported from .doc- or.rtf-files). In addition, new documents can be created and text can be entered there. Hence, new documents can be generated within a MaxQDA project.

Yet, this feature aims at QDA tasks like correcting clerical mistakes in interview transcripts, or transcribing interview records directly within MaxQDA. Therefore, the text processing functionality is relatively limited:

- assortment of different fonts
- change of font size
- support of bold, italics and underlining
- the color of text can be changed
- text aligning to left, middle and right

In particular, MaxQDA does not support:

- the insertion of figures. In addition, figures within documents to be imported are deleted during the import. Moreover, figure subscriptions are kept within the document, which causes inconsistencies.
- the ability to generate a table of content automatically out of flagged headlines
- the inserting of tables
- spell checking
- an undo-function
- background-coloring of text passages

We decided to prepare a specification document in advance, import it and complement the content within this document. One of the reasons for this decision was that the idea to write a specification within MaxQDA was developed during the research process, when we already had started to write down our requirements in a specification. However, there are some additional benefits and we would keep this order in follow-up projects:

- we could import our table of contents and could keep our formatting of headlines. Adding or modifying headlines had to be implemented manually, from that point on, and the modification had to be done at three points

within MaxQDA: the written-out chapter in the specification, the table of contents in the specification, and the corresponding code within the code system (see also chapter *2.4.2.3.1 Introduction of the Implementation*).

- the tables from the imported document were kept, and you could also change the content of the table items. However, it is not possible to add table entries or to generate new tables. We decided to create tables in advance where we wanted to use them and prepare a broad number of table entries, which could be deleted later if we would not need them.

After the development of the specification document is finished, it can by exported into an external file via the MaxQDA export functionality. We propose two activities to follow up:

- We suggest to do a spell check with the help of a conventional text processing tool like MS Word, where the exported file can be opened.
- We had labeled incomplete parts or any sections which we wanted to mark with a "to do" with the construct "TODOBLUE[__text__]" within the specification in MaxQDA, with "__text__" representing a comment or description of the "to do". In this step, we replaced this structure with a graphical accentuation, which contained the description of the "to do". Figure *3.1* gives an example for this process.

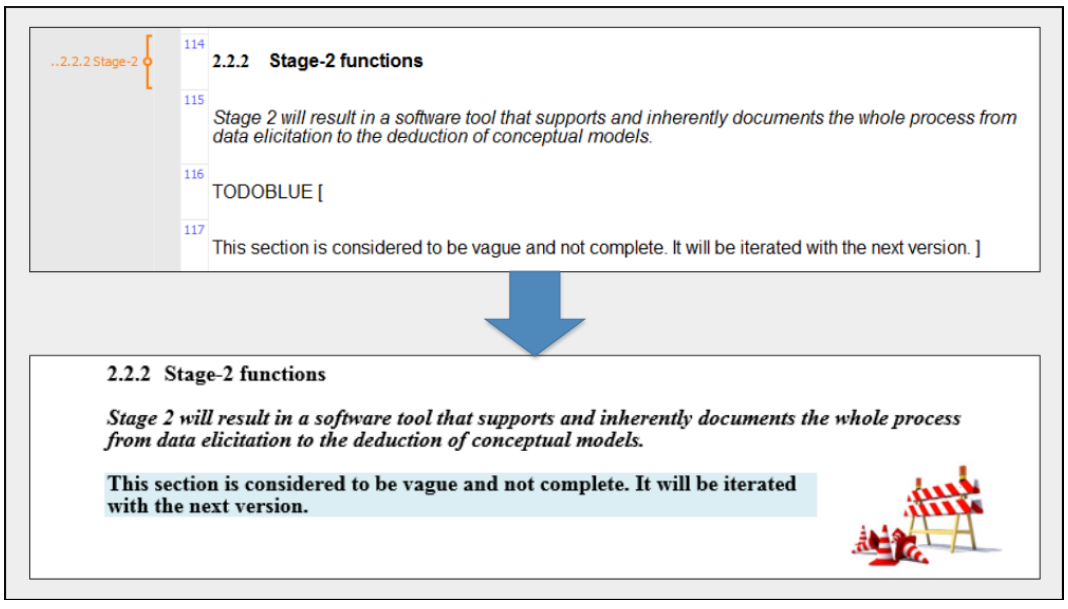


Figure 3.1: Example of the Transformation of a "TODOBLUE"-section to the corresponding accentuated section of the end result

We consciously resigned adding figures in hindsight, since we think that it would significantly affect the established traceability when the document is changed

on proper style with regards to content. Nevertheless, we consider the missing possibility to use figures as a major drawback, they normally act as an important part of specifications and are essential for illustrating relations and structures.

3.3 Outline of Grounded Theory

In this thesis we propose an approach that is based on concepts of the Grounded Theory Method (GTM). The following chapter outlines the basic principles of the original Grounded Theory approach. The adaption is discussed in chapter *2.4 Research Approach*.

3.3.1 Grounded Theory within the Domain of Qualitative Data Analysis

Qualitative data analysis is a discipline from social studies which deals with the investigation and analysis of so-called qualitative data. It is focused on the description, interpretation and understanding of behavior and interdependencies.

Qualitative data are non-numerically measurable data which often present themselves in continuous text or in audiovisual forms. As they are not processable with common standardized techniques (like i.e. mathematical methods), the research methods are mostly flexible and open.

Qualitative research is employed at fields of study where no structures or theories are present that allow a deduction of hypotheses. It aims at finding general rules and therefore derive explanations and an understanding for the field of research.

Grounded Theory is one of the standard methods of methodically controlled qualitative research. It can be described as a research method that "uses a systematic set of procedures to develop an inductively derived grounded theory about a phenomenon" (Strauss & Corbin, 1990, p.24). Originally it was developed by (Glaser & Strauss, 1967).

Today, there are various schools of thought concerning Grounded Theory, which developed out of the diverging understanding of the originators themselves. These different approaches will not be discussed in this thesis since they are beyond its scope. This thesis will follow the methodology proposed by (Strauss & Corbin, 1990).

Grounded Theory is an iterative process with continual elicitation of new data which are coded and analyzed in parallel. This means, in every iteration data is

collected, coded and processed and this data serves also as an input for deciding how to progress further on.

This particularly distinguishes Grounded Theory from other techniques, where in a first step huge amount of data is collected and in a second step this data is analyzed.

3.3.2 Key Concepts of Grounded Theory

3.3.2.1 The Coding Process

Coding is the key process in Grounded Theory (Bryant & Charmaz, 2007, p.265). The task of data coding is to split up the data into units of meaning, ordering them into categories and sub-categories and finally derive a theory about the research topic by building a well-structured, hierarchical arrangement of categories.

The coding process consists of three coding steps: Open Coding, Axial Coding and Selective Coding.

3.3.2.1.1 Open Coding Open Coding is the initial coding step during the data analysis in Grounded Theory and is "the process of breaking down, examining, comparing, conceptualizing and categorizing data" (Strauss & Corbin, 1990, p.61). The new data, like i.e. an interview transcript, is analyzed and split up into units of meaning, thus statements, ideas, or phrases. Each unit is labeled with a so-called *code*, which represents the content of the excerpt or an annotation.

There are four components that build a theory when using the Grounded Theory approach:

- **codes** represent the content of the associated phrase. They should be selected logically and represent the data.
- **concepts** are collections of codes with a similar content. This allows the data to be grouped.
- **categories** denote phenomena. They are collections of similar codes. They serve as a first level of grouping the codes and deriving a theory.
- **properties** are attributes of a category.

3.3.2.1.2 Axial Coding The term axial coding labels the process of ordering the data which were fractionated during the open coding step. By comparing the

similarities and differences between the concepts from the open coding step, they are ordered into categories, which are more abstract.

To help the researcher with a systematic thinking about the data, (Strauss & Corbin, 1990) provide the so-called *coding paradigm*, which focuses the relationships between the concepts to the following aspects: causal conditions, phenomenon, context, intervening conditions, action/interaction and consequences.

3.3.2.1.3 Selective Coding Selective Coding is the final of the three coding steps and applies the definition of core categories. The categories are rarefied at a high level of abstraction and those categories that appear repeatedly are candidates for core categories. In the end, the step of selective coding will lead to the setting up of a theory.

Therefore selective coding resembles axial coding, yet it is done on a more abstract level of analysis (Strauss & Corbin, 1990, p.117).

3.3.2.2 Memos

Memos are short text fields which are each directly associated with a code. The researcher typically composes them throughout the whole process of coding. They are used for recording ideas and thoughts, to note arising questions etc. and therefore support the coding process and ultimately the deriving of the desired theory.

The approach which is proposed in this thesis does also use memos in this conventional style, but in addition suggests to use this free text fields to deposit information which is required to derive domain models, requirements and a glossary. This is described in detail in chapter *2.4.2 Extraction of the Resulting Artifacts*.

3.3.2.3 Theoretical Sampling

"Theoretical Sampling denotes the process of collecting data with the goal of deriving a theory. During the process the researcher collects, codes and analyzes data in parallel and decides, which pieces of information to elicit next and where to find them. This process is controlled by the theory to be developed" (Glaser & Strauss, 1998, p.53).

Hence, this technique aims at choosing new data sources in a way that enables a broader and deeper understanding. The desired result is that only important data is processed. Another result is that the volume of data to be analyzed is reduced.

3.3.2.4 Constant Comparison

The fundamental principle of data analysis in Grounded Theory is the constant comparative method. This means that after the researcher coded a particular phenomenon, he will systematically search for more data which either contrast or confirm the coded phenomenon.

Thereby, data become an indicator for an underlying concept which is denominated through the coding. During the further process, codings will then be mapped into concepts, out of which categories and ultimately core categories will be extracted (Mey & Mruck, 2007, p.25).

3.3.2.5 Theoretical Sensitivity

This term describes the ability of the analyst to recognize what is important in the hitherto data. Therefore it characterizes a personal quality of the researcher.

"Theoretical sensitivity refers to the attribute of having insight, the ability to give meaning to data, the capacity to understand and capability to separate the pertinent from that which isn't. All this is done in conceptual rather than concrete terms. It is theoretical sensitivity that allows one to develop a theory that is grounded, conceptually dense and well-integrated - and to do this more quickly than if this sensitivity were lacking." (Harvard Business School, n.d., p.41f.)

3.3.2.6 All is Data

One of the fundamental properties of Grounded Theory is that not only interviews or documents, but anything that is data can be used by the researcher at studying a certain domain or phenomenon. Everything that helps the researcher generate concepts for the emerging theory is a valid data source, including i.e. meeting protocols, newspaper articles, films, television talk shows, law texts, group meetings etc.

3.3.2.7 Theoretical Saturation

Theoretical saturation means that "no additional data can be found which helps the researcher to develop further properties of the category (Glaser & Strauss, 1998, p.69). When all categories are saturated, the elicitation of data is finished.

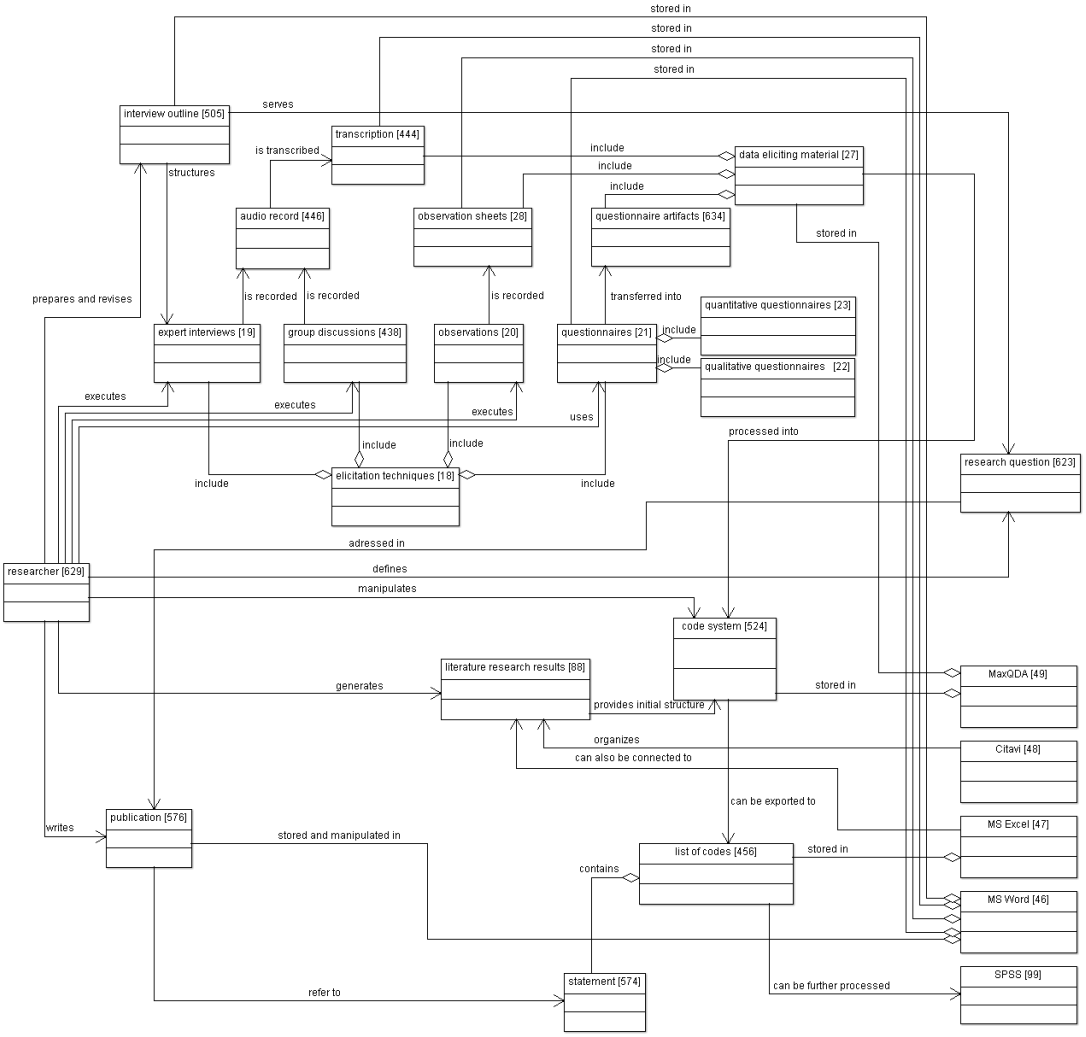
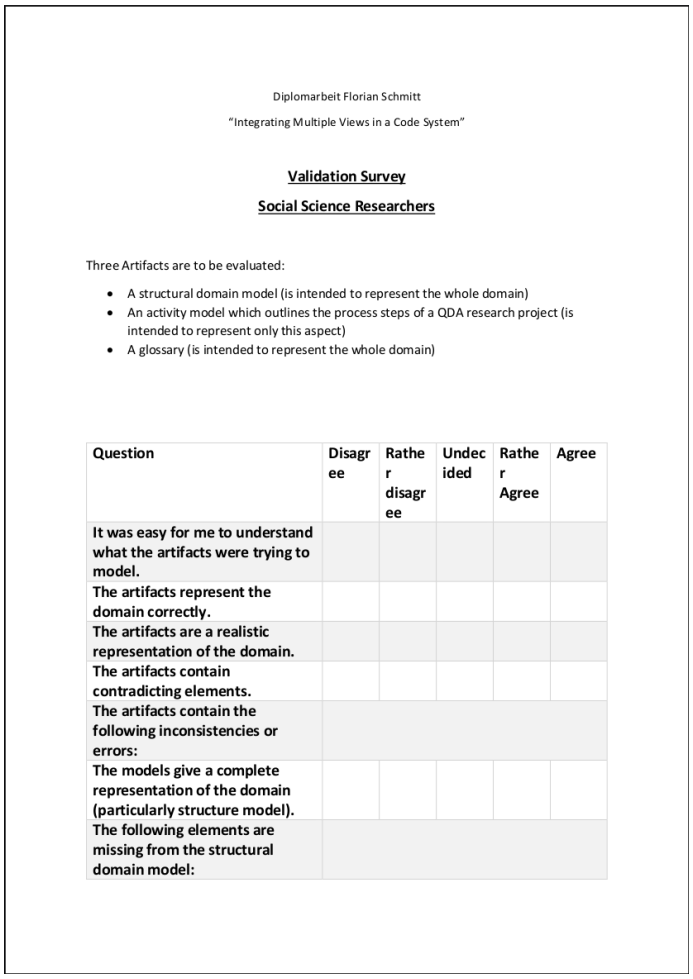


Figure 3.2: Domain Model of the Social Science Domain

Appendix A Domain Model of the Social Science Domain



Diplomarbeit Florian Schmitt

"Integrating Multiple Views in a Code System"

Validation Survey

Social Science Researchers

Three Artifacts are to be evaluated:

- A structural domain model (is intended to represent the whole domain)
- An activity model which outlines the process steps of a QDA research project (is intended to represent only this aspect)
- A glossary (is intended to represent the whole domain)

Question	Disagree	Rather disagree	Undecided	Rather Agree	Agree
It was easy for me to understand what the artifacts were trying to model.					
The artifacts represent the domain correctly.					
The artifacts are a realistic representation of the domain.					
The artifacts contain contradicting elements.					
The artifacts contain the following inconsistencies or errors:					
The models give a complete representation of the domain (particularly structure model).					
The following elements are missing from the structural domain model:					

Figure 3.3: The Validation Survey we provided to the Social Science Domain Experts

Appendix B Validation Survey

Bibliography

Bryant, A. & Charmaz, K. (2007). *The sage handbook of grounded theory*. SAGE.

Coleman, G. & O'Connor, R. (2007). Using grounded theory to understand software process improvement: A study of irish software product companies. , *49*, 654–667.

Glaser, B. G. & Strauss, A. L. (1967). *The discovery of grounded theory: Strategies for qualitative research*. Aldine Pub. Co.

Glaser, B. G. & Strauss, A. L. (1998). Grounded theory. , 53–84.

Gomaa, H. & Wijesekera, D. (2003). Consistency in multiple-view uml models: a case study. In *Workshop on consistency problems in uml-based software development ii* (pp. 1–8).

Harvard Business School. (n.d.). *Basics of qualitative research.* Retrieved from `http://isites.harvard.edu/fs/docs/icb.topic536759.files/Strauss_Corbin_Chaps_3_and_4.pdf`

Kaufmann, A. & Riehle, D. (2015). Improving traceability of requirements through qualitative data analysis. In *Software engineering & management* (pp. 165–170).

Lange, C., Chaudron, M. R., Muskens, J., Somers, L. J. & Dortmans, H. M. (2003). An empirical investigation in quantifying inconsistency and incompleteness of uml designs. In *Workshop consistency problems in uml-based software development ii* (pp. 26–34).

Mey, G. & Mruck, K. (2007). Grounded theory methodologie—bemerkungen zu einem prominenten forschungsstil. , 11–39.

Simmonds, J., Van Der Straeten, Ragnhild, Jonckers, V. & Mens, T. (2003). Maintaining consistency between uml models using description logic.

Strauss, A. & Corbin, J. M. (1990). *Basics of qualitative research: Grounded theory procedures and techniques*. Sage Publications, Inc.

Würfel, D., Lutz, R. & Diehl, S. (2015). Grounded requirements engineering: An approach to use case driven requirements engineering. doi: 10.1016/j.jss.2015.10.024