Requirements Eng
DOI 10.1007/s00766-017-0284-8

ORIGINAL ARTICLE

The QDAcity-RE method for structural domain modeling using qualitative data analysis

Andreas Kaufmann[1] · **Dirk Riehle**[1]

Received: 21 February 2017 / Accepted: 16 October 2017

Abstract The creation of domain models from qualitative input relies heavily on experience. An uncodified ad-hoc modeling process is still common and leads to poor documentation of the analysis. In this article we present a new method for domain analysis based on qualitative data analysis. The method helps identify inconsistencies, ensures a high degree of completeness, and inherently provides traceability from analysis results back to stakeholder input. These traces do not have to be documented after the fact. We evaluate our approach using four exploratory studies.

Keywords Domain modeling · Domain model · Requirements engineering · Requirements elicitation · Qualitative data analysis

1 Introduction

The success of a software development project is highly dependent on the quality of the results of its requirements engineering process [19, 54, 56].

The quality of a requirements specification mainly depends on the experience of the analyst and his or her understanding of the problem domain. To establish a good understanding of the problem domain, the analyst may create a domain model as part of his or her analysis [31, 34, 43].

✉ Andreas Kaufmann
andreas.kaufmann@fau.de

Dirk Riehle
dirk@riehle.org

[1] Department of Computer Science, Friedrich-Alexander-University Erlangen Nürnberg, Martensstr. 3, 91052 Erlangen, Germany

Domain models must correctly represent the reality of the domain and be easy to understand from the stakeholder's perspective [6, 17]. The degree to which this goal can be achieved depends heavily on the modeling experience of the analyst [29, 44].

Qualitative research faces similar challenges. The area under study is often highly complex and the gathered data is frequently unstructured, inconsistent, and incomplete [2, 15, 47]. In scientific research, these challenges are addressed by using codified methods for *qualitative data analysis (QDA)*. QDA methods focus on extracting the relevant information from qualitative data, interpreting the data, and abstracting from it. QDA is employed in theory building research to study a wide range of phenomena through the gathering and interpretation of qualitative data. The process leading to the resulting theory ensures thorough documentation of the analysis process. The documentation is facilitated through memos written by the researcher.

We equate the process of theory building to the domain analysis process. Nuseibeh and Easterbrook state that "we can compare the problem of validating requirements with the problem of validating scientific knowledge" [40]. We take this comparison one step further by proposing that not only the validation of knowledge, but also its elicitation can follow similar procedures.

The domains under study both in theory building and domain analysis are highly complex and the gathered data might therefore be unstructured, inconsistent, and incomplete. Therefore, both challenges require a systematic method for extracting relevant information from qualitative data. This method should guide the analyst in structuring the data and resolving inconsistencies to arrive at a coherent theory or domain model, respectively. The context in which requirements engineering (RE) takes place is usually a socio-technical work system, in which humans

Published online: 02 November 2017

interact with one another and with systems. Understanding how these social units interact and deal with a phenomenon can be viewed as a theory building exercise.

By drawing on theory building in qualitative research as a form of knowledge generation and applying these concepts to domain analysis we address the following common challenges:

- Organizational politics and other social aspects that are uncovered during the analysis can be documented in the same artifact as the analysis result. In fact, the methods we adapt from have been used to study social constructs for decades. The appropriateness of these methods for investigating social aspects has been well established.
- Stakeholders frequently have different goals, leading to conflicting requirements. Managing these conflicts is a challenge and only one solution can be documented in the final artifacts. Within a new intermediate artifact, however, all resolved and unresolved conflicts remain documented.
- While methods and notations for a multitude of models have matured, maintaining inter-model consistency between these models is still a challenge [13]. Our method interconnects several final model types through a new intermediate artifact.

Further challenges such as reusability of requirements artifacts and management of traces are directly impacted by improved documentation as an integral part of the analysis method.

We have developed a method for domain analysis, called QDAcity-RE, which was inspired by QDA methods that have been a mainstay of social science research for more than 50 years [55]. They are also becoming increasingly common in other fields of research. In this article we describe the QDAcity-RE method and evaluate it using four exploratory studies. Through these studies we provide an evaluation of its utility and the qualities of its output.

In the QDAcity-RE method, requirements engineers sample stakeholders, interview them, correlate other materials, and perform QDA of the materials to derive a so-called *code system*. The code system is then extended to derive the relevant requirements engineering results. This paper focuses specifically on the derivation of conceptual domain models using the QDAcity-RE method.

The contributions of this article are twofold:

1. A novel method for domain modeling, which improves over the state of the art by exhibiting the following qualities:
   (a) Inherent pre-requirements-specification (pre-RS) traceability between domain model elements and stakeholder statements.
   (b) Efficient identification of inconsistences in stakeholder statements.
   (c) A high degree of completeness of the resulting domain model.
2. An evaluation of this method using four exploratory studies.

The remainder of this article is structured as follows: First we relate our work to the existing body of research in Sect. 2. Then we present our analysis method in Sect. 3 and the expected benefits in Sect. 4. Four exploratory studies are subsequently presented in Sect. 5, the results of which and their limitations are discussed in Sects. 6 and 7. Finally, a conclusion is drawn in Sect. 8.

2 Related work

We are not the first to use QDA in requirements engineering. Related fields like knowledge engineering and process modeling have also employed QDA [9, 10, 41, 59]. Most of this work does not just use QDA, rather it utilizes QDA in the larger framework of Grounded Theory (GT) [23]. GT is an approach to theory building that provides a methodological framework in addition to the supporting analysis practices. Our approach differs from mentioned work in that we define and evaluate our own method, QDAcity-RE. QDAcity-RE was inspired by GT, but drops many of its epistemological assumptions and preconditions. We do not believe that GT itself can be applied to RE but rather that we need to define and validate our method independently of the social sciences and qualitative research.

Carvalho et al. tested the application of GT to descriptive process modeling by producing two process models: one by an experienced software engineer and one by an experienced qualitative data analyst (researcher) without software engineering experience [9]. They found that using the GT method cannot compensate for the software engineer's expertise and experience. However, its application can improve the modeling process, because it forces the analyst to explore the complexity of the data and to systematically abstract from it. Similar findings are described by Pidgeon et al., who applied GT to knowledge elicitation [41].

Pidgeon et al. add that GT secures the traceability of a derived model back to original data sources through the documentation of the analysis process in codes and memos. However, they point out that the produced model is still an interpretation which needs to be validated.

Both authors criticize the complex and labor intensive analysis process of GT. Their findings can be transferred to the process of domain analysis, which also includes eliciting knowledge from domain experts and analyzing it to derive an abstract model [8]. Our studies concur with [9, 41] concerning the effort required for using methods like GT with the purpose of domain modeling. Many of the problems we encountered in this regard, however, can be solved through better tool support tailored toward domain modeling.

Würfel, Lutz and Diehl propose a holistic approach for data elicitation, data analysis and the determination of requirements, similar to the approach that is proposed by us [59]. They define two process phases: In the first phase, GT practices are employed for data elicitation and analysis. In the second phase, domain descriptions are turned into use cases.

Hughes and Wood-Harper express the need for addressing the organizational context during requirements determination. They demonstrate the use of GT to develop an abstract account of the organization with two case studies [30]. They adapt GT by using predefined categories to address time constraints. The requirements determined in the case studies cover mostly organizational aspects. Examples of such aspects are high-level goals, constraints, and aspects of change. The studies do not, however, show how to extract specific requirements on a lower abstraction level nor how to extract structural elements of an organization. In addition, they do not describe the data analysis process of their case studies in any detail.

Chakraborty and Dehlinger explain how the coding procedure of GT can be applied to determine enterprise system requirements and to derive UML diagrams, thus bridging the gap between qualitative data and final system descriptions [10]. They demonstrate their approach by deriving a UML class diagram from a textual high-level description of a university support system. However, the diagram they developed is not consistent. Features and information about the implementation are represented as classes and the relationships between classes are not specified. An important adaptation in their procedure is the addition of conjectural categories to their model, which were not derived from the data, but were based on the experience of the analysts. They discovered that, apart from the advantage of traceability, the iterative process of GT allows the analyst to discover and close information gaps earlier in the process.

Chakraborty, Rosenkranz, and Dehlinger propose a procedure called *Grounded and Linguistic-Based Requirements Analysis* for eliciting non-functional requirements (NFR) [11]. They argue that the application of GT-based practices in the analysis process improves the requirements specification by facilitating the sense-making of multiple viewpoints into a cohesive description. However, Chakraborty et al. also point out that the differences between RE and theory development make adaptations of GT necessary. Also, because system analysts are not familiar with GT, Chakraborty et al. propose to support the analyst in developing theoretical sensitivity and identifying the important concepts by giving him or her guidance about the theoretical principles to apply.

Chakraborty et al. used predefined categories of NFR. These categories were related using Mylopoulos, Chung, and Nixon's NFR framework [38]. Thomas, Bandara, Price, and Nuseibeh also use an analytical framework, including predefined thematic codes and extraction rules, to use QDA for the determination of privacy requirements for mobile applications [53]. They state that QDA improves requirements elicitation by accounting for contextual factors and securing traceability.

An adaptation of GT many of the presented articles propose is the use of predefined categories. This alleviates the high amount of effort required for a systematic analysis using GT. Traditional GT would not allow for such a-priori constructs or would at least defer their use to the end of a study to make sure that theory development is not biased by preconceived notions of the researcher. In our research we found that besides the obvious impact on resources, the usefulness of predefined categories is highly dependent not only on the domain, but even more so on the desired artifact the analyst wishes to have created at the conclusion of the analysis. For instance when the derivation of natural language requirements from data was desired, we found predefined categories to be immensely helpful, while for the conceptual model we found it more helpful to start without preconceptions in many cases. All of this, however, is still dependent on the domain.

The use of GT to model requirements is also investigated in Halaweh's studies [26, 27]. Halaweh states that categories and their relationships derived from Corbin and Strauss' coding paradigm [15] can be compared to classes and their relationships in class diagrams. Thus, the informal model resulting from GT can be translated into a semi-formal model such as a UML class diagram. Theoretical sampling can help to identify users to interview and theoretical saturation can be used as an indicator to stop requirements elicitation. Halaweh argues that by applying GT and thereby letting requirements emerge from the data, requirements are user-driven, supporting user-centered design and satisfying user needs effectively.

Halaweh points out that the analyst needs to apply theoretical sensitivity in order to produce relevant results. Another claim of his studies is that GT is particularly suited to identify non-technical aspects regarding change due to the system's development and implementation, such as the user's resistance to change. This might help to initiate proactive measures for implementation and training to overcome organizational problems. Halaweh conducted a case study, in

which he analyzed interviews from which he then retrieved a class diagram. However, although he asserts equivalence of GT concepts and object-oriented analysis and design (OOAD) elements, he does not explain these equivalencies and does not present guidelines for coding and transferring an informal model to a class diagram.

The need for more precise expression of the information encoded within a code system has also been discussed in qualitative methods research.

Glaser proposes a more flexible method for relating different concepts of a theory by using theoretical codes, which he divides into coding families [22]. Charmaz criticizes that Glaser does not provide a comprehensive model and that some of the theoretical codes overlap and seem random [12]. Their use is therefore difficult for a requirements engineer who is a novice at GT [10, 12]. However, we investigated Glaser's coding families and found two theoretical codes which are relevant for deriving a structural description of a domain. Since the focus of domain analysis is the investigation of the structure of a domain, additional structural types of relationships apart from "is a" and "is property of" are needed [18].

3 The QDAcity-RE method

QDAcity-RE is a method for domain analysis. The analysis process codified by this method has the goal of creating a code system through iterative refinement from which the domain model is derived.

The code system is a unified model that bridges the gap between stakeholder materials in natural language and more formal models like requirement engineering artifacts. The code system is described in Sect. 3.3, and subsequently the process is detailed in Sect. 3.4.

3.1 Method overview

The domain analysis is performed in an iterative fashion. The main artifact of the analysis, the code system, is incrementally refined until so-called saturation is reached. Saturation, that is sufficient completeness, is reached when the code system does not change significantly with the addition of another iteration of stakeholder sampling, data gathering and analysis.

Each of the iterations consists of the following three steps:

1. Stakeholder sampling
2. Data gathering
3. Data analysis

The sampling of new data is driven by gaps and inconsistencies explicitly documented in the current state of the code system representing the results from all previous iterations. While the means of data gathering, such as interviews, workshops and legacy documentation, are not exclusive to our method, there are specific characteristics of data and techniques of data gathering that QDAcity-RE suits more than others. These characteristics are discussed in Sect. 3.4.2.

The analysis is then performed by qualitative coding of the data in three coding steps, called open, axial and selective coding.

The goal of our analysis process is to make the previously implicit and largely undocumented interpretations and decisions made during domain analysis explicit. The manifestation of this explicit documentation is the main artifact of our method: the code system.

Figure 1 provides an overview of how to perform domain analysis using our method.

3.2 Method context

Our method is primarily aimed at environments where products have a long life-cycle and thorough documentation is needed or even mandated. If it is foreseeable that the system will be replaced by something completely different in the near future, the benefits of our method may not outweigh the increased effort it requires. However if the product will evolve, then the documentation and the traces back to stakeholder statements are a major benefit. The documentation and traceability make it easy to verify if specific concepts are still relevant and why they were modeled in the first place.

Furthermore, in a context where the main sources of information are expert interviews or other highly unstructured information sources, our method is significantly more helpful when compared with situations in which the information sources are already highly structured, for example, situations with existing specifications. In principle, all types of data gathering are supported, and the sources can be combined and correlated. To combine different types of data using data triangulation is recommended.

Our method focuses the data gathering and analysis process. It is assumed that the project vision and project scope have already been defined, although the scope may be refined through the iterative analysis process, if necessary.

3.3 The code system

The code system is a hierarchical structure of codes. It represents a model that captures concepts, categories, their properties and interactions. The code system is produced as a result of the coding process of QDA, which is detailed in Sect. 3.4.3.

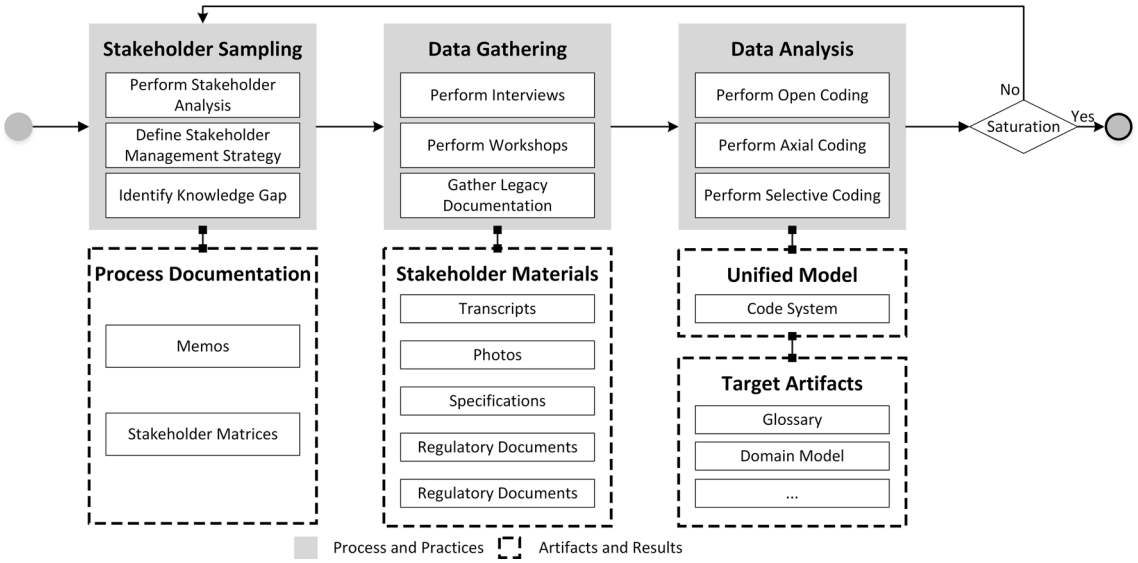

Fig. 1 The QDAcity-RE process for structural domain modeling

Each code in the code system describes a concept grounded in the gathered data. When codes are first created during the analysis process, they are loosely coupled. With an increasing number of iterations the code system is formed as hierarchical structure of these codes as nodes. Each code has a label, a definition, and instructions on its intended use as well as a description differentiating it from other codes and possible misinterpretations. In addition to this information, which is typically documented in a code book [35], each code contains meta information on which kind of entity it is (activity, actor etc.).

The code system bridges the gap between natural language text containing stakeholder information and requirements engineering artifacts like the analysis domain model. Each code is linked to one or more text segments in the gathered data. These instances of a code are called a coding, and the length of the coded text segment is called the unit of coding (sentence, paragraph, multi-paragraph, etc.).

Since the code system provides a holistic view on the phenomenon, it acts as a common denominator between multiple models describing different views on the domain (i.e. structural, behavioral, data, communication etc.). This strengthens inter-model consistency. In our exploratory studies, for instance, we derived both a UML class diagram as well as a BPMN diagram from one code system, and we derived a feature model, class diagram, and domain specific language (DSL) from another code system with all model elements traceable to the code system, linking to elements from various model types.

An example of a code system excerpt from one of our studies with its traces to associated stakeholder statements and parts of the model is presented in Fig. 2. The granularity of the traces back to the stakeholder interviews can be varied through the unit of coding. A typical unit of coding ranges from a part of one sentence to multiple paragraphs.

3.4 The QDAcity-RE process

The analyst starts the domain analysis process by constructing a broad initial interview guideline.

This initial guideline evolves with each iteration of data gathering and analysis to close knowledge gaps identified through previous iterations and to resolve inconsistencies. Increasing specificity of the guideline does not necessarily lead to a structured interview. Still, structured interviews or even questionnaires can cover information that require a larger empirical sample. For this, they can provide valuable supplementary material to strengthen the validity of the findings through method or data triangulation. Triangulation is a term describing a set of practices used to vary different aspects of the analysis to gain insights on the phenomenon from different perspectives. If different perspectives on the phenomenon lead to the same conclusion, the analysis result is believed to be more credible. The increased credibility is assumed because the analysis is grounded in different types of data (*data triangulation*), or because the data was analysed through different types of activities (*method triangulation*) or by different investigators (*investigator triangulation*). A fourth form of triangulation is considered, when the results were verified by people with an external perspective on the research project using the same data (*theory triangulation*).

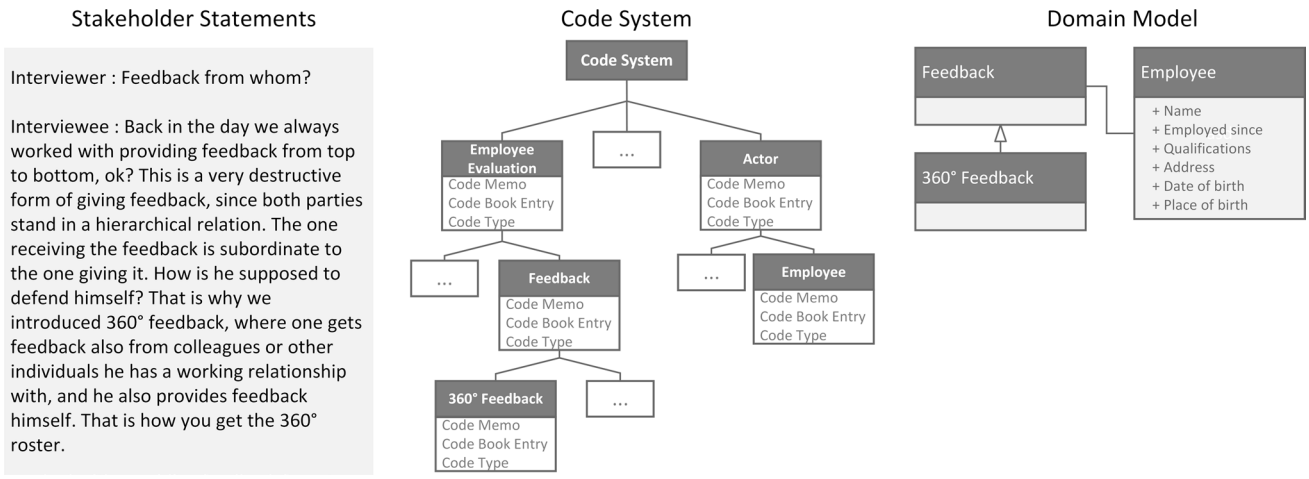

Fig. 2 Codesystem example

While triangulation is not a required part of the QDAcity-RE method, it is a practice that lends itself well to domain modeling as well and can easily be integrated.

3.4.1 Stakeholder sampling

With the interview guideline ready, the analyst then performs stakeholder sampling to find those individuals who are best able to discuss the topics of the interview guideline. The sampling strategy is not fixed for the entirety of the analysis, but only for the next iteration. It has to be performed with consideration for what the gaps, inconsistencies, or novel insights are that are present in the current state of the code system.

The sampling strategy is called *theoretical sampling*, because it draws on the current state of the theory on how the phenomenon could be modeled as supported by the evidence already collected and analyzed at the time of sampling. This information is codified in the code system. Theoretical sampling promotes a more flexible and agile way of sampling, as opposed to defining the data gathering process a-priori based on assumptions about the domain and the different stakeholders. The data gathering is not performed in a predefined order.

In qualitative research, theoretical sampling is often considered the ideal way to shield the outcome from being influenced by preconceptions [50] and to highlight information gaps [12].

3.4.2 Data gathering

During the data gathering phase of each iteration, the analyst extracts and documents unstructured information from stakeholders' both explicit and implicit knowledge. The goal is to document as much information as possible in a way that allows for a structured analysis leading to consolidation with other materials and abstraction into uniquely identifiable pieces of information that describe specific parts of the domain.

QDAcity-RE can be paired with a wide range of methods for collecting unstructured data from stakeholders, such as workshops, interviews, observations, surveys or creativity techniques such as brainstorming. Our method treats all data the same and unifies the information content of different media types in a single artifact, the codesystem, which encompasses the consolidated information from different input artifacts.

All of these data gathering methods can be used in our method, however, interviews take the most prominent role. The coding of transcribed interviews using QDA can provide the highest additional value. We have found that our method provides the highest value in a context, where the majority of the information is available only in unstructured form. Interviews document the stakeholders' thoughts in a way that supports an unbiased analysis through a third-party analyst. Through the analysis the information within the interviews gets structured and becomes easier to navigate. In contrast, workshops, for example, often require more structured moderation in order to coordinate a larger group of stakeholders. This added structure, however, lessens the value of QDA, because the analyst is most likely to just follow this structure in his analysis, preventing a possibly more natural structure to emerge from the data.

Interviews and workshops are transcribed, and can be coded together with legacy documentation and regulatory texts.

Throughout the whole process the analyst has to be open to new ideas emerging by letting the gathering process be steered in large parts by the stakeholders. This concept is called *theoretical sensitivity* and it helps to identify what is

significant to the interviewee without being biased by preconceived notions.

Within the scope of this work we use interview transcripts, marketing material, natural language documentation, regulatory documents and photographed hand-drawn illustrations as input data for our exploratory projects.

3.4.3 Data analysis

The analysis of the gathered data is driven by the *coding process*. During coding the gathered data is annotated to highlight the most insightful parts of the text, resulting in a *code system*. The code system is a hierarchically structured set of codes, representing common concepts, that connects unstructured data to structured RE artifacts, such as the domain model and the glossary. The code system thus ensures inter-model consistency.

The coding process consists of the following three activities, which are performed in sequence.

1. Open coding
2. Axial coding
3. Selective coding

These three activities are performed during each iteration of data gathering and analysis until a stopping criterion, the so-called *theoretical saturation*, is reached. Reaching the stopping criterion indicates sufficient completeness of the analysis results.

It is important to limit the amount of data that is added in each iteration. Especially the first coding step, open coding, becomes more difficult to perform as the amount of new data in each iteration increases. In a research context, typical increments add new data collected from 1 to 5 interviews for each iteration. For our exploratory projects we added one interview per iteration because of the relatively small scope of each project.

During the first stage of coding, open coding, the analyst creates an unsorted list of labels and assigns each label to one or more text segments. These labels are called *codes* and the portion of data that has been coded is called a *coding*. The granularity of the coded segment is called the *unit of coding*, and may vary from single words, sentences to multiple paragraphs or pages.

Codes are referred to as in-vivo codes [12, 15], if their name is directly mentioned in the unstructured data, but codes can also be abstractions from the original material. Because a domain model should represent the domain terminology [34], in-vivo codes should be the most common codes. Synonyms should be documented with the code and ultimately in a glossary. It is common, that in the beginning hundreds of open codes are created on a multitude of abstraction levels. We advise to first generate specific concepts for smaller units of coding and then to combine them during the abstraction process. Specifically when coding within the context of domain analysis, it is important to make all aspects of a phenomenon explicit in separate codes. A code "employee attends development measure" is not easily mapped into a domain model, because it includes several aspects: the actor "employee", the activity "attending development measure", and the event "development measure". In addition, the analyst should be careful to describe activities with verbs and not with nouns in order to distinguish them from events.

The extracted codes are then structured hierarchically by grouping them into categories during axial coding to form a map of concepts supported by the analyzed documents. Categories represent the aspects central to the domain and are described further with regard to their properties and context through constant comparison and questioning. The data fragments indicating the properties should also be coded. Both structural and dynamic aspects can be developed into categories. However, if the purpose of the analysis is clear, such as the extraction of a conceptual domain model, the analyst may focus on aspects which are central for the analysis and investigate these first. During the axial coding step, the code system metamodel is used to define the types of relationships that may be modeled within the code system.

The last step in a coding iteration, selective coding, helps model only aspects within the definition of the scope of the project. The selection and focus on a few high-level phenomena reflects what is central to the domain, what belongs to it to support the central concepts, and what is not part of the code system and consequently not significant for the domain.

During this coding step, core categories are chosen, which holistically describe the studied phenomenon. All other codes have to be subsumed by a core category. The code system should describe core categories in all of the following five dimensions to be considered complete:

1. Actions & strategies
2. Consequences
3. Causal condition
4. Contextual condition
5. Structural condition

This is borrowed from social science research, where these dimensions form the corner stone of the coding paradigm [16]. The coding paradigm aims at increasing the systematization of that process [51].

Codes that do not fit in any core category are considered not relevant. While in research this criterion of relevance is highly influenced by the research question as well as the domain, for domain modeling it is purely dependent on the domain since the question constituting the reason for the

analysis is always "What are the concepts in this domain and what are the relations between them?".

In theory building research, all categories of the code system will ultimately be subsumed under a single core category. However, selecting a single core category does not make sense for the purpose of domain analysis, because a domain model should give a complete representation of the domain [6]. The phenomena, i.e. entities, which are central to the domain have already been identified as being important by developing them into categories.

3.4.4 Iterative refinement

After the initial collection and analysis of data, the analyst enters an iterative process repeating the concurrent collection and analysis of data, where the sampling of new data should be sensitive to how the code system evolves.

After each iteration of data collection and analysis, the analyst should reevaluate what data should be collected next based on the current state of the emerging domain theory. The current state of the code system and lacking description of the core categories thereby drives the sampling process for the next iteration of data gathering and analysis.

New iterations are performed until saturation reaches a defined level. Full saturation is reached, when through the additional gathering of materials selected through theoretical sampling, no new codes in the code system emerge, and the definitions of existing codes remain stable.

Although the concept of theoretical saturation is frequently suggested, the metrics for measuring saturation are rarely documented. An overview of the problem of data saturation is presented by Francis et al. [20]. They propose to start with an initial analysis sample and a stopping criterion, which is the number of consecutive interviews that have to be analyzed following the initial analysis sample, without new themes emerging from the data. Both measures have to be defined a-priori, depending on the complexity of the studied phenomenon. They conclude, that a 10 + 3 (initial + stopping) rule for their saturation criterion may be regarded as a reasonable value, if there is no specific indications on the required sample size within the problem domain.

Whatever saturation metric may be considered adequate in a specific case, it is imperative in any case that the measure be clearly documented and consistently measured. Although the analysis process can be used without measuring saturation, we advise to define an explicit criterion that is actually measurable and track this metric throughout the lifespan of the project.

Using the concept of theoretical saturation yields a metric for the quality criterion of completeness. Further, the hierarchical structure of the code system assists the analyst in identifying conflicts and contradictions. This hierarchical structure is a direct result of the coding process and since concepts describing the same semantic entity will be located in close proximity within the tree structure, inconsistencies will be easier to identify.

During each new iteration, the analyst is required to look for evidence or contrary indications of newly emerging codes in already coded documents. He or she is also required to look looking for evidence of established codes in the new data and to identify new concepts that previously were not prevalent. This behavior is called constant comparison.

The code system is further refined iteratively while finding support for a theory, establishing new codes or combining and eliminating codes which are not sufficiently supported by the data.

Throughout the whole analysis process, the documentation of the data gathering and analysis process plays a vital role. One common practice for facilitating such documentation in qualitative research is memo writing. Memos can be attached either to specific codes or as project memos on the analysis process to the whole code system or specific documents. Memos are thus important for describing the meaning of different concepts within the domain and explaining the decisions within the analyst's mental process. In current tool support for QDA, memos are also the only way of describing more expressive relationships between concepts. This is one aspect we want to improve about current qualitative research processes: We want to make information which researchers usually only write up informally, explicit and machine processable by using more convenient and reliable means than natural language processing (NLP).

4 Expected benefits

When applied to the creation of domain models during requirements engineering, the expected benefits of our approach are the following:

1. It closes the gap between the informal stakeholder material and formal domain models by adding pre-Requirements-Specification (pre-RS) traceability.

   This traceability is embedded in the RE process and documented in a new unified model, the code system. The inherent traceability eliminates the need to create and maintain traces after the fact.
2. It improves the process for deriving domain models from stakeholder materials by

   (a) providing a defined process, where previously business analysts mostly had to rely on intuition and experience.

(b) allowing the definition of a measurable stopping criterion to determine when the requirements elicitation process exhausted the relevant cases.

3. It improves domain model quality by

(a) ensuring completeness of domain models, where previously key input might have been missed.
(b) ensuring consistency, by following principles of the constant comparison method.

Current tools support the documentation of traceability manually, linking requirements back to specific artifacts. However, creating and maintaining these matrices is a laborious documentation task that does not provide additional benefits for the actual analysis of the source documents [5, 14, 28]. If the documentation, however, is created as part of the text analysis, it can serve both purposes: to better understand the target domain, and to create better documentation of the analysis process. The improved documentation makes each element traceable.

We use qualitative research methods to solve part of the "grand challenge of traceability" [24], ubiquitous traceability, in a pre-RS context.

The fine grain traceability provided by using QDA methods further improves the ability to perform change impact analysis. If any passage within the source documents changes, the corresponding parts of the model can be identified and adapted, and vice versa.

Through the traces, decisions made during the analysis process become explicit. For instance, when resolving conflicting descriptions, alternative interpretations or conflicting viewpoints are documented beyond what is visible in the final model.

Our approach also empowers less experienced analysts by offering a codified method, that the analyst may follow to achieve higher quality models of the domain.

Further, the in-depth analysis through qualitative coding, especially when a high degree of in-vivo codes are used, contributes to a better understanding and definition of terminology that is close to the language use of the stakeholders.

These benefits can, however, not be achieved at zero cost. Our experience, which is in line with previous related studies, is that using QDA methods for domain analysis increases the effort required for the analysis significantly. A cost-benefit estimation has to be made on a case-by-case basis. Further research into a measurable effort impact of QDAcity-RE is pending. The required effort is directly dependent on the eventual tool support.

Table 1 Coded data

Study	Data	Coded segments
Medical image diagnostics	8 in-depth interviews	1563
Railway systems	4 in-depth interviews Project documentation Norms and standards	754
Human resource development	6 in-depth interviews 6 workshop transcripts	1237
Qualitative research	6 in-depth interviews	778

5 Exploratory studies

After having defined our method, we now present four studies in which we explore its application. These studies are detailed in their respective Sects. 5.1, 5.2, 5.3 and 5.4.

The first two projects in which we developed the method as outlined in Sect. 3 were performed in the domains of medical imaging diagnostics and railway systems. Both required the creation of a conceptual domain model which we developed using QDA techniques. These projects were interwoven with the initial creation of the method and were used to explore how the transfer from research method to domain analysis can be implemented.

Following these two projects we applied our learnings in two additional studies. In these two studies we evaluated the QDAcity-RE method within the domains of human resource (HR) development and qualitative research methods.

Table 1 provides an overview of the qualitative data coded in the scope of the four exploratory studies.

Table 2 further details some of the key differences among the four cases.

With each of the four cases, which were executed in the order presented here, we aimed to refine our method and thus focused on a specific aspect. The focus of each study is as follows:

1. Medical image diagnostics
   - Establish feasibility for semi-formal modeling
   - Create a DSL using QDA
2. Railway systems
   - Improve conceptual modeling using QDA
3. HR development
   - Compare workshops and interviews as input
   - Document full traceability
4. Qualitative research
   - Derive a conceptual model, a behavioural model and a natural language specification from the same codesystem

Table 2 Overview of exploratory studies

Study	Industry project	Member checks	Data triangulation	Outline conversations	Fully traceable	Evaluation
Medical image diagnostics	×	Informal	×	–	–	–
Railway systems	×	Follow-up interviews	×	×	–	–
Human resource Development	partly	Follow-up interviews	–	×	×	Expert survey Ontology
Qualitative research	–	Informal	–	×	×	Expert survey

Table 3 Evaluation model

Quality/goal	Evaluation criteria
1. Leads to better documentation	Model elements can be traced to stakeholder statements
	All model elements are grounded in stakeholder statements
2. Improves completeness	Domain expert does not consider important elements missing
	Model covers all aspects in existing knowledge representations
	The analysis process reached saturation
3. Improves consistency	Domain expert can not identify any inconsistencies
	Inconsistencies were uncovered during coding
4. Can handle different types of input materials	Different input materials used (Interviews, Workshops etc.)
5. Supports the creation of different target artifacts	Different output artifacts created with inter-model traceability

The evaluation model we used to determine the success of these studies is presented in Table 3. The criteria of this model are examined for each case in Sect. 7.1.

We used MAXQDA[1] as the analysis tool for our exploratory studies.

5.1 Domain modeling for medical imaging diagnostics

We first applied our method to the design of a DSL for medical imaging diagnostics. This study was conducted in collaboration with Siemens Healthcare [36].

To create a DSL, the analyst not only requires technical know how, but needs to have a deep understanding of the domain. A deep understanding is required so the DSL will be accessible intuitively for the domain experts, who represent the target audience of most DSLs. We therefore conducted a domain analysis using our QDA based method. As part of this domain analysis a feature model and a conceptual domain model were created.

Through our analysis we identified typical workflows for medical imaging diagnostics using computer tomography (CT) and magnetic resonance imaging (MRI). These workflows guide the diagnostician through the diagnosis processes. The goal of defining these workflows using a DSL was to achieve standardized medical findings.

To elicit the processes in which users typically engage to diagnose an image, eight interviews were conducted. These were subsequently transcribed and analyzed using an early outline of our qualitative analysis method. The interviewed personnel included the two project managers, two test engineers a software architect and three product coaches who were working with all stakeholders to define the product requirements.

In addition to these interviews we utilized background material on typical oncological diagnostics provided by the Siemens AG. We triangulated the results using multiple data sources. This increases trustworthiness [25].

After the qualitative analysis concluded, and a stable code system supported by over 1500 coding instances within the texts had emerged from the data the code system was then transformed into a formal feature model.

Even without tailoring the coding process toward feature models around 80% of the codes that emerged from the data could directly be translated into a node of a feature model.

While the translation into a feature model could be performed with only minor modifications, the modeling on the domain's structure through a UML class diagram required more implicit knowledge of the domain. This knowledge was not represented in the code system. We attribute this to the fact that within this study the coding was performed very similarly to how it is performed in theory building research. This meant a high degree of freedom regarding the structure and semantic of the code system. A classification of codes

[1] http://www.maxqda.com.

as classes, attributes or relationships had to be added ad-hoc during the manual transformation process. Some of this information could be extracted from code memos, but most of it was created through an additional interpretative step. A more structured approach would help this transformation, however if the coding becomes more restrictive it also limits the possibility of unexpected results emerging from the analysis. Further research into finding the appropriate balance in this respect is needed.

In a final step, the feature model was then translated into a DSL, ensuring that the terminology used in the DSL is the same that was used by the interviewed participants of the study.

From this experiment we concluded that a partial automation of transforming a code system into a domain model warrant further investigation. We also concluded, that such a transformation would have to be supported by a more formal code definition which is supported by a meta model.

5.2 Domain modeling for the openETCS toolchain

The second exploratory study was performed in the domain of railway systems in collaboration with members of the openETCS project. Our partner for this second study was Deutsche Bahn AG. The purpose of this project was to understand the needs within the openETCS project toward their tool chain. In essence, we performed requirements elicitation and analysis for tool needs of the software development process within the openETCS project.

5.2.1 Data sources

Two major data sources were used for this project: We conducted four interviews with three openETCS stakeholders and we processed the relevant official documents from the openETCS repository on GitHub and involved them into our coding process.

The individuals we interviewed are directly involved in the openETCS project as team members of the development team. All interviews were executed in a semi-structured way. The initial interview took place in September 2013. Our interview partner was the project leader of the openETCS initiative. At this point, we had received an informal introduction into the project through a telephone conference with two of the project leaders. We had also received access to the documents repository. In preparation for the first interview we analyzed the currently existing requirements document and carved out inconsistencies, imprecise wordings and mistakes. The discussion of these aspects served as an introduction to the different topics, but the interview became very open and often one aspect brought up the next one. This resulted in a long and detailed conversation which covered the whole project.

5.2.2 Data analysis

The interview guideline was then incrementally adapted to address issues that were revealed through constant comparison such as the following.

- Gaps within the current code system, where a lack of deeper information was evident.
- Discrepancies within the current code system which originated in contradictions in the statements of different interviewees.
- New aspects of the target domain that appeared during the analysis.

We used open questions to create a relaxed atmosphere and encouraged the interviewees to talk about what came to their mind. They were also free to change the topic if they wanted to. The prepared questions served only as an outline of the conversation. The intention was to let the interviewee 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 we did not understand or included unclear details, or when the current interviewee contradicted statements from earlier interviews, we specifically inquired these issues and asked for more details.

After running through the coding process, we then revised and checked the new version of our code system for quality. Hence, the code system is smoothed and corrected after each iteration. In addition, the memos containing the code definitions were updated after each iteration. Inconsistencies were documented, as were poorly understood concepts, which served as the basis for discussion within the next interview.

5.2.3 Evaluation of openETCS study

Within this study we investigated the abstraction levels which naturally occur in the code system through the coding process with regard to a mapping of codes to domain model elements. We found that our method is capable of processing pieces of information on all levels of abstraction, since it facilitates their hierarchical and logical ordering. An abstract concept will be found on a high level within the code system and its details will be subsumed in the subordinate levels. In general, the codes that were mapped into concepts of the domain model could typically be found on the middle levels. The highest code system levels provided an abstract perspective split and therefore a structural order, i.e. “tools” vs. “artifacts”. The low-level codes on the other hand mostly represented details of a concept such as the concept’s behaviour or particular attributes.

With this being the first within our four studies that documented full traceability between the domain model and

original stakeholder material by means of the code system, we were frequently challenged by lacking tool support for the coding process. We chose to document the additional meta information necessary for creating a conceptual model from the data within the code memo and maintaining these manually. On the side of the domain model the links were represented by the code ID, which was also maintained manually. We expect to address these issues with our own tooling solution in the future.

5.3 Domain modeling for human resource development

In this study we employed our method for domain modeling in the environment of human resource (HR) development. The main functions of HR development are (1) training and development, (2) organization development, and (3) career development of employees within an organization [58]. The goal of this project was to evaluate the use of QDAcity-RE for the creation of a conceptual domain model, using in-depth interviews with experts working in the field of HR development as a data source.

5.3.1 Data sources

All domain experts who participated in this study had high level management positions in HR and experience in HR development.

Their employing companies varied in size from a local company with 50 employees to an international corporation with over 100,000 employees worldwide and operated in the sectors IT and market research.

The first interview was guided by 12 open questions, which aimed at gaining an overview over the domain. For the following interviews, analysis results determined the interview questions according to the principle of theoretical sampling. We conducted semi-structured interviews, so the prepared questions were used as a guideline and we adjusted to participant's answers [15, 37]. This was important because we wanted to capture the knowledge of the domain experts and not force preconceptions on the data [15, 39]. To clarify inconsistencies, close information gaps, and extract more detailed information, we conducted follow-up interviews with two of the domain experts.

As a secondary data source, literature on HR development [1, 4, 48, 52] was used to clarify the definitions of terms. Although literature research prior to or at the beginning of the research project is avoided in GT, Corbin and Strauss believe that literature may be used to support the analysis as soon as the main categories of the theory have emerged [21].

The interviews were audio recorded, anonymized and transcribed manually. Corbin and Strauss advise to transcribe interviews fully at the beginning of the research project and in later stages only to transcribe those parts of an interview which are important for the theory [15]. To limit the risk of missing useful information, we transcribed the whole content, but left out introductory and closing conversations and defined a simplified transcription system [33]. The speech parts of interviewees were transcribed word for word, including laughter. However, we did not include details such as accentuation or the lengths of breaks, because they are not relevant for the purpose of our research [3]. For the speech parts of the interviewer, we left out parts which did not include any information such as expressions of comprehension, because this would interrupt the information given by interviewees unnecessarily.

5.3.2 Data analysis

When we applied the method described in Sect. 3 to our example, concepts emerged from the data during open coding as explained above. The coding process started after the first interview had been conducted and transcribed. In order to represent the domain terminology, primarily in vivo codes were used [3, 34]. Units of coding varied in size from one phrase to a whole paragraph. Coding a whole paragraph was sometimes necessary to preserve information about the relationships between concepts. The units of coding belonging to one concept were compared to investigate their differences and similarities and to guide the questions for the following interviews.

Usually, actors are not coded explicitly in GT research projects, because they are intertwined with other concepts. For example, a study investigating how patients deal with pain includes concepts such as "experiencing pain" or "pain", but no concept "patient" [12, 15]. However, actors, including external systems and organizational units, need to be represented in a conceptual domain model [34, 45, 57]. For the domain of HR development, for example, "employee" is a central concept. The same is the case for objects and places, which are normally not investigated explicitly during GT research. Therefore, actors, places and objects, which includes tangible and intangible objects and the concept type "idea" of GT, need to be coded as well.

Because conceptual domain models represent the entities of a domain, these are the phenomena we want to study and were therefore developed into categories. Concepts which seemed to belong to the same aspect were grouped into categories. For example "giving feedback", "feedback survey", "360-degree feedback" and "evaluating feedback" were grouped under "feedback".

We also coded background information, such as the position of the interviewee in the organizational structure and the current systems in use, as well as information about the purpose of HR development. Although these codes should be clearly distinguished, such information should be captured and kept in mind during the analysis, as it might be the

Table 4 Evaluation of domain model by domain experts

Question	Disagree	Rather disagree	Undecided	Rather agree	Agree
It was easy for me to understand what the model was trying to model			1	1	1
The model represents the domain correctly			3		
The model is a realistic representation of the domain			1	2	
All the elements in the model are relevant for the representation of the domain			2	1	
The model gives a complete representation of the domain				3	
The model contains contradicting elements		2	1		

reason for differences between incidents and contain important information for later design decisions.

5.3.3 Evaluation of HR study

The domain model created through our method was evaluated with regard to the following quality aspects proposed by Bolloju and Leung [6]:

- Syntactic quality: The domain model adheres to the modeling language.
- Semantic quality: The domain model represents the reality correctly and completely.
- Pragmatic quality: The domain model is easy to understand from the stakeholders' perspective.

We used basic notation elements of UML class diagrams in accordance with the UML. Adherence to the syntax was ensured by using tool support for domain modeling. To assess the perceived semantic and pragmatic quality, we conducted a qualitative survey of the participating domain experts. The evaluation of semantic quality was completed by comparing our domain model with an existing ontology of the domain to assess the congruence of identified concepts with established research.

For our written survey (adapted from [42]), we received answers from three of the four participating domain experts as shown in Table 4.

The domain model was evaluated to give a rather complete, realistic and correct representation of the domain. The only concept which was identified as missing was "criteria of potential". Within the interviews we conducted, the topic of potential was only mentioned once as being currently in discussion for implementation, thus did not show to be significantly relevant according to the data. However, as saturation could not be reached, this concept might appear during further analysis. The only inconsistency which was reported, was that performance assessment did not necessarily evaluate target agreements. Domain experts' descriptions of the relationship between competency, performance, employee assessment and target agreements were inconsistent and imprecise. Their statements were therefore compared and further investigated in interviews, which resulted in the distinction between competency and performance assessment and a defined relationship between performance evaluation and target agreements. However, the inconsistencies and imprecisions in the data were not completely resolved because saturation could not be reached and would need to be investigated further with additional interviews. The received feedback suggests that regular validation of analysis results should be part of the domain analysis process to improve the quality of the domain model.

The domain experts were undecided if all elements in the domain model were relevant for the representation of the domain. This was to be expected as the evaluation of relevance depends on the purpose of the domain model and the desired level of abstraction. These concerns also lead experts to be undecided whether the domain is represented correctly.

Answers regarding the perceived pragmatic quality varied. The domain model was perceived as confusing by some of the domain experts. This might be attributed to the challenge of identifying the optimal abstraction levels within the code system to be matched to the conceptual model, and presents an opportunity for improvement in regard to the design of the domain model and our analysis method. Specifically, it should be investigated if clearly defined abstraction levels in the code system can help to improve the clarity of the domain model.

To further assess the congruence of identified concepts, we compared our domain model with Schmidt and Kunzmann's competency-based ontology of HR development [49]. While the ontology only covers HR development with regard to competency management, all participating domain experts stated that performance management was also a part of HR development and our analysis showed a close interrelationship between these two sub-domains. Thus, our domain model provides a more holistic representation of the domain. In comparison, our domain model covers 70% of the concepts from the ontology, while 50% of the competency-related classes (excluding sub-classes) from our domain model are represented in the ontology. However, the identification of equivalent concepts was based on our interpretation, because Schmidt and Kunzmann do not provide definitions of their concepts. This shows the value

of creating a glossary to provide a thorough understanding of the identified concepts. Using our method, concepts and their definitions are developed simultaneously and directly linked, which ensures consistency between the domain model and the glossary.

5.4 Domain modeling for qualitative research

The goal of our fourth study was the application of our method for the domain analysis and requirements elicitation for tool support for qualitative analysis methods such as employed in our domain modeling method. This study had a focus on the integration of multiple views in code systems (conceptual view, process view, requirements specification etc.). This specific research question is beyond the scope of this article, therefore we focus on it as another application of our method with a conceptual domain model as a resulting artifact which was evaluated through expert feedback.

5.4.1 Data sources

The data source for this project was a series of expert interviews. In total, five interviews with five stakeholders were carried out. Four of them were 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 QDA and specifically on how they perform the task of coding. Three of the four researchers we talked to perform social science research and use QDA 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 fifth interviewee was a software engineer working on QDA tooling.

5.4.2 Data analysis

As with the other exploratory projects, we let the code system structure emerge out of the data. Hence, we initially kept very close to GT. All interviews were performed in a semi-structured way. The intention was to let the interviewees speak freely. This eases the discovery of topics that the analyst might not yet be aware of. However, when statements came up we did not understand or included unclear details, or when the current interviewee contradicted statements from earlier interviews, we asked for more details about these topics. During the first iterations' axial and selective coding steps, the code system developed into a direction that might be the expected outcome of a conventional analysis as well: Since the purpose of this project was the creation of a domain model and a requirements specification through QDAcity-RE, the structure split up into separate parts for the social science domain analysis and for the requirements analysis. Issues like the project steps for the former and development constraints for the latter emerged out of the data as core concepts. These concepts naturally 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?".

Speech of both interviewer and interviewee were transcribed word for word. We excluded accentuation of statements and non-verbal communication. Furthermore, we left out speech parts that only stated an expression of comprehension, which would interrupt the flow of information and make the text more difficult to read. In some cases we decided to directly skip short passages in statements where the speaker misspoke and corrected himself. Some expressions of colloquial speech or dialect were transformed into equivalent standard language expressions.

In order to assess the current state of the code system with regard to its completeness at any given point in time, we assigned a traffic light color to each code within the memo, specifying how well the concept is described in the code system:

- *Red* huge gaps of information, lots of data missing
- *Yellow* contradictions within related codes, information gaps, open tasks for this section
- *Green* section is complete, no questions, requirements are written down, tasks are done

We denoted the specifics of the gaps, problems, contradictions, thoughts or questions within the code memos. Hence, they provide both a quick overview of status and a flexible way to denote issues.

5.4.3 Evaluation of qualitative research study

We asked the domain experts to give feedback on our results by participating in a written survey.

Two of the experts provided feedback, which was positive. One of them pointed to a missing detail. Beyond, they agreed to our results (see Table 5).

During the evaluation of the previous study we assumed that some of the "undecided" answers were the result of a lacking context with regard to the purpose of the model, we added the category "can't be assessed" to allow for this distinction to be made explicit.

Table 5 Evaluation of domain model by domain experts

Question	Disagree	Rather disagree	Undecided	Rather agree	Agree	Can't be assessed
It was easy for me to understand						
what the model was trying to model				1	1	
The model represents the domain correctly				1		1
The model is a realistic representation of the domain				1		1
The model gives a complete representation of the domain				1		
The model contains contradicting elements	1		1	1		

6 Limitations

6.1 Application domain

During our exploratory studies, some important differences between traditional GT and its application to domain analysis became apparent. The most significant difference is the focus on either behavioral or structural aspects. GT focuses on interaction systems and therefore mainly uses concepts to describe dynamic aspects of a research area, i.e. actions which indicate a phenomenon [16]. A conceptual domain model however describes the important entities of a problem domain and their structural relationships [7].

Although the data sources used for domain analysis, such as domain expert knowledge, contain mostly dynamic descriptions of the domain, an analysis method must provide a way to extract structural entities [46, 57]. These structural aspects need to be described with their attributes and related to each other, the same way phenomena under study in GT are represented with categories and properties.

6.2 Sampling and saturation

In all four cases theoretical sampling could not be fully applied due to limited access to interview partners. This meant that we applied theoretical sampling mainly to the choice of questions and not the choice of interview partners. Especially when we interviewed domain experts from different companies, we had to start each interview with basic questions to understand the specific context within the company. Thus, the interviews provided rather high level information. To address a lack of detail in certain cases we conducted follow-up interviews to retrieve more detailed information. However, theoretical saturation could not be reached due to availability constraints of domain experts, and time constraints for each of the projects.

Furthermore the definition of a sensible criterion for saturation in the context of domain analysis is subject of ongoing study. Configuring an individual metric based on logged changes based on the specific domain is our current recommendation. However what dimensions of changes should be tracked, how they should be weighted and what a reasonable default preset could be is subject to further research.

6.3 Lack of tool support

Due to a lack of dedicated tool support for parts of the method, assumptions we have on the effectiveness of the method regarding the effort needed for its execution can not be validated. While the activity of coding the data is supported by existing tooling which we employed in our cases, these tools do not adequately support the documentation of machine readable meta information required for our method. Thus many of the process steps were executed using a pen and paper method of tracking all links between the documents which is error prone and also distracts from the actual task. We expect to alleviate some of these concerns with our own tool support in the future.

7 Discussion

7.1 Evaluation of exploratory projects

Table 6 provides an overview over the four studies with regard to our evaluation model presented in Sect. 5.

All four of our studies confirmed an excellent level of documentation for the analysis process. This documentation includes traces from each model element back to individual stakeholder statements and documentation. The documentation also includes the reasoning for most interpretations in short memos. A researcher not involved with the coding of the data could easily assess the relevance of different aspects to the group of interviewed experts.

The first two studies did not yield an explicit validation of the resulting artifacts against a fixed reference point or through an objective third party. However, these studies were useful in exploring the viability of the approach and identifying pitfalls that had to be addressed by our method. They also validated our hypothesis that, using QDAcity-RE, it is possible to use the code system as a unified model that connects different target artifacts through inter-model traces

Table 6 Study evaluation

	1 Documentation	2 Completeness	3 Consistency	4 Input types	5 Model types
Medical Imaging	Partial traceability Memos	–	–	Expert interviews	Feature model DSL Conceptual model
openETCS Tool-chain	Full traceability Memos	–	Problems systematically documented and resolved	Expert interviews Norms	Conceptual model Glossary
HR Development	Full traceability Memos	Expert survey Ontology	Expert survey Problems systematically documented and resolved	Expert interviews Workshops Drawings	Conceptual model
Qualitative Research	Full traceability Memos	Expert survey	Expert survey Problems systematically documented and resolved	Expert interviews	Process model SRS Conceptual model

which makes navigating between the different artifacts easier and also fosters inter-model consistency. To this end, the first study derived a DSL a feature model as well as a conceptual model from the same code system, while the second study did the same for a conceptual model and a glossary. The fourth project also contributed toward this dimension of evaluation by combining the extraction of a Software Requirements Specification (SRS) in natural language with the creation of a process model and a conceptual model from the same dataset using the same code system. In this last case the code system had to be adapted with a predefined structure on the top two levels to make part of it specific to NL requirements documents.

The artifact quality concerning completeness and consistency of the created models was evaluated through surveys with domain experts conducted within the 3rd and 4th study as well as through a comparison with an ontology that has been established independently from our research.

All of the four studies used expert interviews as a means of data collection, supplemented by a variety of other materials such as drawings, norms and regulations, formal documentation and workshop transcripts. While our studies indicate that all of these types of materials can be analysed using the same method it is our finding that our method provides more value the less structured the data is. This correlates with our finding, that a bottom up or middle-out approach suits our method better than a top down approach. This coincides with the most common use of QDA in qualitative research that, to a significant extent, can be considered inductive theory building.

7.2 General results

During our study, we found that the coding procedure supported the structuring and analysis of qualitative data for conceptual domain modeling. Important concepts became apparent already early in the coding process. This was also the case for data which emphasized process descriptions, which interviews tend to favor. The participating domain experts primarily gave an account of their domain from a process point of view. Through the development of concepts and categories, the structural aspects emerged and could be further investigated through theoretical sampling. Inconsistencies could be investigated through comparing the respective data fragments and notes could be taken in code memos about questions which need to be asked in the next interview and about the different options of interpretation. This was especially important for integrating company-specific descriptions of HR development into a consistent domain model.

The systematic coding procedure and the writing of memos make the process of domain analysis traceable, but coding and modeling decisions are still interpretive and therefore depend on the analyst's experience and expertise. What to code and how to develop concepts into categories is a difficult task for which there is not one simple solution. We found that abstracting too early in the process or focusing too much on the domain model while coding can make later changes more difficult. However, the analysis method provides more guidance to a novice analyst for extracting a domain model. Systematic coding helped us to engage with the domain to be analyzed, where previous domain knowledge was limited.

Although theoretical sensitivity also depends upon the researcher's level of experience in qualitative research and the phenomenon under study, it develops further during the research process and can be enhanced using techniques for questioning the data or systematically analyzing a word or phrase and comparing different incidents [15, 27, 32]. This suggests that while a requirements engineer still benefits from his or her experience in the domain under study, a systematic analysis procedure can support him or her

to develop theoretical sensitivity with regard to domain analysis.

The practices of constant comparison, theoretical sensitivity and questioning of the data can also help to prevent experienced analysts from prejudiced misconceptions. On the other hand we experienced the coding process as time consuming and requiring a high cognitive effort, in accordance with many of the authors of related work.

Our studies suggest that our analysis method favors a bottom up or middle-out approach and provides less additional value for a top-down analysis approach.

8 Conclusions

We present and evaluate a novel approach to domain analysis by adapting qualitative research methods from the social sciences. In our approach, the social science research process of theory building facilitates domain analysis within the requirements elicitation phase. We show how an iterative process of concurrent data collection and analysis can be applied to requirements engineering, including open, axial, and selective coding of qualitative data. Our method inherently produces traceability of requirements back to original statements by stakeholders, which does not have to be created and maintained separately after the fact. The traces are documented in an analysis artifact called the code system which evolves iteratively with the analysis process.

We showed that by applying QDA to domain analysis, structural elements and relationships needed to derive a UML class diagram can be extracted from a code system based on interviews with domain experts. Constant comparison and theoretical sampling assist in integrating differing domain descriptions into an abstract model. While the analysis process still includes interpretations and modeling decisions, our method provides more guidance than existing domain analysis approaches and a thorough documentation of these decisions. In addition, codes and memos ensure traceability between the original data and the derived model and assist in connecting several RE artifacts ensuring a high degree of inter-model consistency.

Acknowledgements We would like to thank Katharina Kunz, Florian Schmitt and Benjamin Mempel for their valuable contributions executing the exploratory studies. We would also like to thank all anonymous interview partners as well as Siemens Healthcare, Deutsche Bahn and the openETCS project for participating in our studies and providing valuable feedback to improve our method. Finally, thanks to Hannes Dohrn, Maximilian Capraro, Michael Dorner, Nikolay Harutyunyan and Daniel Knogl for workshopping this paper to improve its presentation, and to Ann Barcomb for proofreading.

References

1. Achouri C (2015) Human resources management: eine praxisbasierte Einführung. Springer, New York
2. Balzert H (2010) Lehrbuch der softwaretechnik: Basiskonzepte und requirements engineering. Springer, New York
3. Bazeley P (2013) Qualitative data analysis: practical strategies. Sage, Newbury Park
4. Becker M (2009) Personalentwicklung-bildung, förderung und organisationsentwicklung in theorie und praxis. 5. erw. Aufl. Stuttgart. Schäffer-Poeschel Verlag, S 546
5. Blaauboer F, Sikkel K, Aydin MN (2007) Deciding to adopt requirements traceability in practice. In: Krogstie J, Opdahl AL, Sindre G (eds) Advanced information systems engineering. Springer, pp 294–308
6. Bolloju N, Leung FS (2006) Assisting novice analysts in developing quality conceptual models with uml. Commun ACM 49(7):108–112
7. Broy M (2013) Domain modeling and domain engineering: Key tasks in requirements engineering. In: Münch J, Schmid K (eds) Perspectives on the future of software engineering. Springer, pp 15–30
8. Byrd TA, Cossick KL, Zmud RW (1992) A synthesis of research on requirements analysis and knowledge acquisition techniques. MIS Q 16:117–138
9. Carvalho L, Scott L, Jeffery R (2005) An exploratory study into the use of qualitative research methods in descriptive process modelling. Inf Softw Technol 47(2):113–127
10. Chakraborty S, Dehlinger J (2009) Applying the grounded theory method to derive enterprise system requirements. In: 10th ACIS international conference on software engineering, artificial intelligences, networking and parallel/distributed computing, 2009. SNPD'09. IEEE, pp 333–338 (2009)
11. Chakraborty S, Rosenkranz C, Dehlinger J (2015) Getting to the shalls: facilitating sensemaking in requirements engineering. ACM Trans Manag Inf Syst TMIS 5(3):14
12. Charmaz K (2014) Constructing grounded theory. Sage, Newbury Park
13. Cheng BHC, Atlee JM (2007) Research directions in requirements engineering. In: 2007 future of software engineering, FOSE '07. IEEE Computer Society, Washington, pp 285–303. doi:10.1109/FOSE.2007.17
14. Cleland-Huang J, Gotel OC, Huffman Hayes J, Mäder P, Zisman A (2014) Software traceability: trends and future directions. In: Proceedings of the on future of software engineering. ACM, pp 55–69
15. Corbin J, Strauss A (2014) Basics of qualitative research: techniques and procedures for developing grounded theory. Sage, Newbury Park
16. Corbin JM, Strauss A (1990) Grounded theory research: procedures, canons, and evaluative criteria. Qual Sociol 13(1):3–21
17. Cruzes DS, Vennesland A, Natvig MK (2013) Empirical evaluation of the quality of conceptual models based on user perceptions: a case study in the transport domain. In: Ng W, Storey VC, Trujillo JC (eds) Conceptual modeling. Springer, pp 414–428
18. Daoust N (2012) UML requirements modeling for business analysts: steps to modeling success. Technics Publications, Denville
19. Dvir D, Raz T, Shenhar AJ (2003) An empirical analysis of the relationship between project planning and project success. Int J Proj Manag 21(2):89–95
20. Francis JJ, Johnston M, Robertson C, Glidewell L, Entwistle V, Eccles MP, Grimshaw JM (2010) What is an adequate sample

size? Operationalising data saturation for theory-based interview studies. Psychol Health 25(10):1229–1245
21. Gibson B, Hartman J (2013) Rediscovering grounded theory. Sage, Newbury Park
22. Glaser BG (1978) Theoretical sensitivity: advances in the methodology of grounded theory. Sociology Press, Mill Valley
23. Glaser BG, Strauss AL (2009) The discovery of grounded theory: strategies for qualitative research. Transaction Publishers, Piscataway
24. Gotel O, Cleland-Huang J, Hayes JH, Zisman A, Egyed A, Grünbacher P, Dekhtyar A, Antoniol G, Maletic J (2012) The grand challenge of traceability (v1.0). In: Cleland-Huang J, Gotel O, Zisman A (eds) Software and systems traceability. Springer, pp 343–409
25. Guion LA (2002) Triangulation: establishing the validity of qualitative studies. Extension Institute of Food and Agricultural Sciences, Gainesville, FL. http://www.rayman-bacchus.net/uploads/documents/Triangulation.pdf. Accessed 27 Sept 2009
26. Halaweh M (2012) Application of grounded theory method in information systems research: methodological and practical issues. Rev Bus Inf Syst (Online) 16(1):27
27. Halaweh M (2012) Using grounded theory as a method for system requirements analysis. JISTEM J Inf Syst Technol Manag 9(1):23–38
28. Hofmann HF, Lehner F (2001) Requirements engineering as a success factor in software projects. IEEE Softw 18(4):58
29. Hruschka P (2014) Business analysis und requirements engineering: Produkte und Prozesse nachhaltig verbessern. Carl Hanser Verlag GmbH & Co. KG. ISBN-13: 978-3446438071
30. Hughes J, Wood-Harper T (1999) Systems development as a research act. J Inf Technol 14(1):83–94
31. Insfrán E, Pastor O, Wieringa R (2002) Requirements engineering-based conceptual modelling. Requir Eng 7(2):61–72
32. Kelle U (2010) The development of categories: different approaches in grounded theory. The Sage handbook of grounded theory. Sage, Newbury Park, pp 191–213
33. King N, Horrocks C (2010) Interviews in qualitative research. Sage, Newbury Park
34. Larman C (2005) Applying UML and patterns: an introduction to object-oriented analysis and design and iterative development. Pearson Education India, Delhi
35. MacQueen KM, McLellan E, Kay K, Milstein B (1998) Codebook development for team-based qualitative analysis. CAM J 10(2):31–36
36. Mempel B (2014) Definition einer DSL mittels QDA, Master thesis, self published
37. Myers MD, Newman M (2007) The qualitative interview in is research: examining the craft. Inf Organ 17(1):2–26
38. Mylopoulos J, Chung L, Nixon B (1992) Representing and using nonfunctional requirements: a process-oriented approach. IEEE Trans Softw Eng 18(6):483–497
39. Nohl AM (2013) Narrativ fundierte interviews. In: Bohnsack R, Flick U, Lüders C, Reichertz J (eds) Interview und dokumentarische Methode. Springer, pp 13–26 (2013)
40. Nuseibeh B, Easterbrook S (2000) Requirements engineering: a roadmap. In: Proceedings of the conference on the future of software engineering. ACM, pp 35–46
41. Pidgeon NF, Turner BA, Blockley DI (1991) The use of grounded theory for conceptual analysis in knowledge elicitation. Int J Man Mach Stud 35(2):151–173
42. Poels G, Maes A, Gailly F, Paemeleire R (2005) Measuring the perceived semantic quality of information models. Springer, New York
43. Pohl K, Rupp C (2011) Requirements engineering fundamentals: a study guide for the certified professional for requirements engineering exam-foundation level-IREB compliant. Rocky Nook Inc, San Rafael
44. Prieto-Díaz R (1990) Domain analysis: an introduction. ACM SIGSOFT Softw Eng Notes 15(2):47–54
45. Rosenberg D, Stephens M (2007) Use case driven object modeling with UML. APress, Berkeley
46. Rupp C, Queins S et al (2012) UML 2 glasklar: Praxiswissen für die UML-Modellierung. Carl Hanser Verlag GmbH Co KG, Munich
47. Rupp C et al (2014) Requirements-Engineering und-Management: Aus der Praxis von klassisch bis agil. Carl Hanser Verlag GmbH Co KG, Munich
48. Ryschka J, Solga M, Mattenklott A (2010) Praxishandbuch Personalentwicklung: Instrumente, Konzepte. Springer, Beispiele
49. Schmidt A, Kunzmann C (2006) Towards a human resource development ontology for combining competence management and technology-enhanced workplace learning. In: On the move to meaningful internet systems 2006: OTM 2006 workshops. Springer, pp 1078–1087
50. Strauss A, Corbin J et al (1990) Basics of qualitative research, vol 15. Sage, Newbury Park
51. Strübing J (2004) Was ist grounded theory? In: Bohnsack R, Flick U, Lüders C, Reichertz J (eds) Grounded theory. Springer, pp 13–35
52. Thom N, Zaugg RJ (2009) Moderne Personalentwicklung: Mitarbeiterpotenziale erkennen, entwickeln und fördern. Springer, New York
53. Thomas K, Bandara AK, Price BA, Nuseibeh B (2014) Distilling privacy requirements for mobile applications. In: Proceedings of the 36th international conference on software engineering. ACM, pp 871–882
54. Verner J, Cox K, Bleistein S, Cerpa N (2005) Requirements engineering and software project success: an industrial survey in Australia and the us. Aust J Inf Syst 13(1):225–238
55. Wacker JG (1998) A definition of theory: research guidelines for different theory-building research methods in operations management. J Oper Manag 16(4):361–385
56. Wallace L, Keil M (2004) Software project risks and their effect on outcomes. Commun ACM 47(4):68–73
57. Wazlawick RS (2014) Object-oriented analysis and design for information systems: modeling with UML, OCL, and IFML. Elsevier, Amsterdam
58. Werner J, DeSimone R (2011) Human resource development. Cengage Learning, Boston
59. Würfel D, Lutz R, Diehl S (2015) Grounded requirements engineering: an approach to use case driven requirements engineering. J Syst Softw 117:645–657