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

EUGEN ANANIN

MASTER THESIS

A Quality Metric of QDA-Derived Theories Using Object-Oriented Modeling

Submitted on February 2nd 2015

Supervisor: Prof. Dr. Dirk Riehle, M.B.A.
Andreas Kaufmann, M.Sc.
Professur für Open-Source-Software
Department Informatik, Technische Fakultät
Friedrich-Alexander University 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, February 2nd 2015

License

_____

Erlangen, February 2nd 2015

Abstract

Qualitative data analysis is widely accepted as valid approach for inductively developing theories. The in-depth analysis of individual experience often results in novel findings, potentially explaining less common phenomena. However, to achieve valuable results, the discovery must be compliant to various implications and prescribed processes. Grounded theory is a qualitative methodology constituted by very specific procedures, which in turn are supposed to foster scientific rigor. However, there is no definite framework or evaluation strategy, defining which criteria constitute good theory. By building upon principles of qualitative analysis and object-oriented programming, this research suggest an approach to quality assessment for emergent theories. Results demonstrate that a semi-formal memo annotation enables evaluation of code-systems, while providing traceability and follow-up data processing.

Zusammenfassung

Qualitative Datenanalyse ist ein anerkannter Ansatz für die induktive Entwicklung von Theorien. Die tiefgründige Untersuchung individueller Erfahrungswerte führt häufig zu neuartigen Erkenntnissen und kann potentiell weniger bekannte Phänomene erklären. Um allerdings brauchbare Ergebnisse zu liefern, müssen verschiedene Implikationen und Prozesse bedacht werden. Grounded Theory als qualitative Methodik, ist durch sehr spezifische Verfahren gekennzeichnet, welche die wissenschaftliche Sorgfalt gewährleisten sollen. Allerdings gibt es in diesem Kontext kein Rahmenwerk oder definitiv festgelegte Strategien zur Bewertung von Theorien. Aufbauend auf Prinzipien aus der qualitativen Forschung, sowie der objektorientierten Programmierung entwickelt diese Forschungsarbeit ein Konzept für die Qualitätsbewertung von neuartigen Theorien. Ergebnisse dieses Projektes zeigen, wie durch semi-formales Annotieren Code-Systeme bewertet werden können, während gleichzeitig Rückverfolgbarkeit und Weiterverarbeitung gewährleistet werden kann.

Contents

List of Figures

List of Tables

List of Abbreviations

CAQDAS	–	Computer assisted qualitative analysis software (CAQDAS)
GT	–	Grounded Theory
OSS	–	Open Source Software
QDA	–	Qualitative Data Analysis

1 Introduction

Qualitative data analysis (QDA) is a common research approach for inductive theory development with ability to incorporate unique insights (Hoda, Noble, & Marshall, 2010). Its growing popularity resulted in more and more fields of application (Yin, 2011). However, this variety of research areas and the different goals of analyses ended in methodological pluralism, which in turn is complicating quality assessment (Easterby-Smith, Golden-Biddle, & Locke, 2007). Further it can be stated, that the challenge is related to the lack of straightforward procedures (Bryman & Burgess, 1994).

Grounded theory (GT) provides a set of processes aiming at establishing scientific rigor and valid results (Glaser & Strauss, 1967). However, the findings of such analysis, just like any emerging qualitative theory, cannot be evaluated in a standardized way (Lincoln & Guba, 1986). In this context the utilization of computer assisted qualitative data analysis software (CAQDAS) bears potential for comparability and follow-up data processing of theories (Rodon & Pastor, 2007).

This thesis developed and applied software metrics derived from the GT coding paradigm (Corbin & Strauss, 1990) and object-oriented principles, to a prior developed code system in order to evaluate its quality. Thereby a semi-formal method for annotating memos is proposed. As a result, quality evaluation becomes possible and eventually bridges the gap from qualitative to quantitative analysis (Baxter & Eyles, 2015).

Applying the action-oriented coding paradigm, supports the creation of well-structured theory (Kelle, 2005) and likewise contributes to subsequent quantification (Salinger, Plonka, & Prechelt, 2008). The results of this evaluation provide evidence, that GT can produce findings, which then can be translated into a domain model. Consequently, the suggested process for quality assessment can be considered as attempt to establish a generic method of evaluation. In addition it is aiming for a canonical data format suited for theory export, extension and reuse (Mühlmeyer-Mentzel, 2011).

1.1 Syntax Errors vs Quality Measures

Initially the research objective was to develop a quality metric by deriving concept types, based on conceptual relationships evident in a code-system and consequently counting syntax errors of such model. However, the analyzed data did not provide appropriate information to systematically elaborate concept-types for quality assessment. In turn, evidence from literature supported the augmentation of this data, in particular analyzing and adding concept-attributes based on the coding paradigm (Strauss & Corbin, 1994). This allowed for elaboration of a formal annotation method and subsequent computer assisted processing. Eventually, the abstraction and conceptualization of the code system could be evaluated and instead of errors quality aspects were counted, resulting in multiple software metrics.

2 Research Chapter

2.1 Introduction

Qualitative Research is a common approach in social sciences and increasingly popular in any kind of research constituted by analyzing human interaction (Bryman & Burgess, 1994). Such research design permits the analyst to get close to the data and to become familiar with the involved participants and their experiences (Mintzberg, 1979). Consequently, qualitative methods have been accepted in organizational research (Avison, Lau, Myers, & Nielsen, 1999; Buchanan & Bryman, 2007) or fields of high importance but scarce existing knowledge like for example information systems research (Walsham, 1995). (Walter Daniel Fernández, 2003; Lehmann, 2001)

Often in such context interview analysis is conducted for generating novel but valid theories based on empirical evidence (Eisenhardt, 1989). These cases can be understood as instances of richly described phenomena, highly related to the context in which they occur (Robert K Yin, 2014). With focus on elaborating constructs, measures and testable theoretical propositions the opportunity is created to bridge from rich qualitative evidence to mainstream deductive research (Eisenhardt & Graebner, 2014). However, in concurrent literature such processes are vividly discussed in terms of rigor and high quality results. (Benbasat & Zmud, 1999; Walter D Fernández, Lehmann, & Underwood, 2002; Gray, 2001)

Defining concrete methods and analytic strategies supports transparency of analysis and traceability of results (Thomas, 2006). In this context GT is seen as highly systematic approach constituted by rigorous processes of data abstraction and conceptualization (Glaser & Strauss, 1967). The ability to incorporate unique insights makes it increasingly popular in the evaluation of human aspects (Carver, 2004; Hoda et al., 2010; Orlikowski, 1993).

CAQDAS can support rigorous research and the handling of empirical evidence, however lacks the possibility of analyzing resulting theory are interchangeable data formats (Kepper, 1996; Puebla & Davidson, 2012; Reiter, Stewart, & Bruce, 2011). Canonical formats can help to overcome this challenge and bear great potential for developing frameworks, thesauruses and dictionaries (Fiat & Sanders, 2009; Glaser & Strauss, 1998; Liu, 2009). Yet, there is no common method or quality criteria to be utilized in this context. (Matavire & Brown, 2008)

2.2 Related Literature

2.2.1 Qualitative Research

Qualitative research is a broad term for various approaches, characterized by detection of novel findings in the context of human interaction (Bryman & Burgess, 1994). Such research is constituted by three elements. First comes the data collection for specific phenomena or topics. It is followed by coding, which is an analytic or interpretive process, where the data is conceptualized, named and mapped to its source (Strauss, 1995). The researcher performs a critical analysis of the provided data, while trying to recognize and avoid his own preferences and tendencies. It is particularly important to facilitate abstract thinking so that valid and reliable findings can achieved. Finally a report is composed and the research can be considered complete. (Yin, 2011)

It is widely accepted, that this kind of research is focused on creating rich descriptions und understandings of social interactions. Thereby its advantages are isolation of causal conditions, operationalizing theoretical relations, potentials for quantifying phenomena, aiding research designs for generalizing findings and finally developing general laws and theories. However, this method is related to various problems, too. The selection of adequate data sources is a critical point to analysis and the relevance of the results is often complicated by limited existing knowledge. (Flick, 2009)

Another challenge of such analysis can be seen in the management of huge amounts of empirical data, which are often coded in texts and possibly have multiple meanings on individual and social levels. Consequently, the importance for data reduction, data display and verification, can be derived (Miles & Huberman, 1994). QDA aims at fracturing and managing the data gathered into themes or essences. The elaborated results can potentially be fed into descriptions, models or theories. (Walker & Myrick, 2006)

2.2.2 Grounded Theory

The development of GT as research methodology was introduced by Glaser and Strauss (1967), resulting from their experiences in the domain of qualitative research with the focus for increasing rigor of analysis processes and validity of the findings. Several elements can be considered as core. First, the inquiry is shaped by the aim to discover social and social-psychological processes (Strübing, 2008). Further, the phases of data-collection and data-analysis happen simultaneously. The inductive analytical process prompts theory discovery and development, rather than verification of existing knowledge. Also theoretical sampling, which is purposeful selection of additional evidence, refines elaborates and exhausts conceptual categories. Finally it can be said, that systematic application of GT-analytic methods will lead to more abstract levels of information (Charmaz, 1997).

The iterative process of evaluating empirical data in order to develop concepts is called coding by Glaser and Strauss. In the context of QDA the goal is to create access to findings, based on interpretation of the data. The method of Constant Comparison between the data collection and its analysis is the driving idea. Glaser & Strauss (1967) argue, that constantly comparing the findings will lead to the generation of theoretical properties for a category. Thereby category means a theoretical construct with structural characteristics emerging from analytical comparison. This process is constituted by three phases of coding, namely open coding, axial coding and selective coding (Strauss, Corbin, et al., 1996), which are accompanied by theoretical sampling of data, systematic dimensionalizing of concepts and theoretical saturation of the elaborated concepts. (Strübing, 2008)

2.2.2.1 Open Coding

Open Coding is the procedure for developing categories by examining the data source for salient categories. Analytically extracting phenomena and their properties helps breaking-up the data and supports categorizing, which means grouping concepts that seem to be related. Beginning with microscopic analysis, theoretical information from literature or the informant's terms (in-vivo) aids development of concepts (Rodon & Pastor, 2007). These are understood as abstract representations of events, objects or actions, which the evaluator identifies as significant to the data (Glaser & Strauss, 1998). Consequently the researcher names such concepts and applies

the code to the corresponding part of the data source, which is called the labelling phenomena (Glaser, 1992).

When assigning names or properties, a mere description should be avoided and instead a more abstract conceptualization should be preferred. The grounded theory approach makes use of constant comparison, resulting in a close connection between categories and the data (Corbin & Strauss, 2008). Further theoretical sampling is performed, which according to Glaser and Strauss means gathering data with the goal of generating a theory. While gathering this data the researcher simultaneously codes and analyses the data and decides which data is to be collected next and where it can be found. The process is controlled by the material or formal theory, emerging during research. (Glaser & Strauss, 2005)

It is important to know the general properties of a category in order to examine its occurrence in the data. In dimensionalizing specifics of an occurrence are described as a sum of characteristic attributes, which are developed during systematic and constant comparison. In detail that means analysing if the occurrence is specific for a certain perspective or a rather general one, therefore may be suited for grouping into a concept. Using similar or equally important characteristics or dimensions in order to consolidate different concepts into a category, it is essential for the process of elaborating types. These attributes also prompt collection of additional data or the enrichment of the existing data in regard of theoretical sampling. This is fundamental for the connection of data-collection, data-analysis and theory elaboration. Further it will lead to theoretical density and sufficient differentiated concepts eventually. (Strübing, 2008)

2.2.2.2 Axial Coding and Coding Paradigm

The process of interconnecting categories, hence elaborating a phenomena-based relationship-model, is called axial coding. Corbin & Strauss (1990) argue, that axial coding is focused on possible relations between one category and different other concepts and categories, while the researcher has to decide upon criteria of relevance. He has to choose the phenomena, which, corresponding to the current state of analysis, will probably contribute to the clarification of the research question. Consequently, a number of vague hypotheses is constructed and afterward those are declared as core categories, which are responsible for the most useful results.

Categories are dimensionalized and also have properties, which are presented on a continuum. That means one can have multiple perspectives of the category. The dimensional analysis is an attempt to make the different perspectives explicit and systematic. It aims at creating analytical diversity, while decreasing the complexity by assigning findings to theoretical expressions. In regard of elaborating perspectives the researcher should consider the specific contexts, conditions, actions, processes and their consequences. (Corbin & Strauss, 2008)

Enhancing his work on Grounded Theory, Strauss introduced the Coding Paradigm. This concept is a suggestion for axial coding and aims at increasing the systematization of that process (Strübing, 2008). The paradigm suggests that during analysis of relations in the axial coding phase the researcher should evaluate findings by considering (1) the examined central phenomena, (2) context conditions related to the phenomena, (3) intervening or structural conditions, (4) causal conditions, (5) actions and strategies in regard of the phenomena and finally (6) consequences of the actions or strategies. This way, the prior isolated phenomena can be associated in a structural context. (Corbin & Strauss, 1990)

In contrast to the selective coding it is important to mention, that the paradigm is focused on single empiric occurrences and their abstractions. Instead of answering the research question its purpose is to explain the realization and the consequences of an incident or a certain kind of incidents (Strübing, 2004). Following graphic visualizes the meta-model of the paradigm concepts.

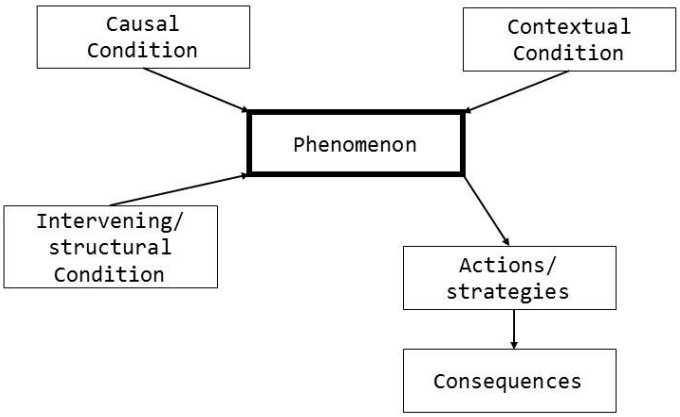


Figure 1: Coding Paradigm (Corbin & Strauss, 1990)

2.2.2.3 Selective Coding

The procedure for integrating previously developed theoretical concepts into the final theory is called selective coding. Corbin and Strauss define the process as selecting core categories, systematically relating core- to other categories, validating the relationships and filling of categories needing further refinement and elaboration. This means that a part of the data is re-coded, so that relations between data-based concepts and core categories can be examined, and eventually will lead to theoretical closure or saturation. This occurs when continued systematic data-collection supports previous findings and does not yield any new insights. Reaching this point the sampling strategy changes and the researcher tries to compare concepts that probably have differing characteristics.

Due to the nature of Grounded Theory with its iterative and cyclic elaboration process, the selection of incidents and data cannot be planned in advance or made dependent on generic rules. Instead the selection is based on the analytic questions, derived from elaborated theoretical concepts at the current state of the research (Strübing, 2008). Therefore instead of generating hypotheses from samples, rather questions and perspectives for subsequent data gathering and analysis are deducted. The sources which will be added and used for further study, is selected in such way, that it supports the finding of new properties and dimensions of the current concepts or maybe even help to develop new categories. The changes arising from this process are not understood as corrections of wrong codings, but can be seen as adjustment of the analytic perspective for increased consistency. A consistent analysis-perspective and successive development of the research question will often result in one or few core concepts that answer the examination question.

2.2.2.4 Memos

Writing code-memos potentially leads to the best relational model provided for integrating substantial codes into theoretical concepts (Holton, 2005; Domínguez-Cherit et al., 2009). Field notes are the basis for memos, while memos are the basis for theory development (Montgomery & Bailey, 2007). Such information can be seen as conceptual meaning combined with ideas for the theory recorded at the moment of occurrence (Glaser & Strauss, 1998). Advanced field notes contribute to staying focused, result in higher conceptualization and help to avoid drowning in details. Due to creativity and coding freedom no standardized memo format was defined (Martin & Gynnild, 2011). However, the grounding of findings can be improved, when detailed descriptions of categories are linked to the evidence in text and improve traceability in consequence (Thomas, 2006).

Memos can be considered as recordings of analysis, thoughts, interpretations, questions and directions for further data collection (Glaser & Strauss, 1998). To establish the advantages of memos, it is suggested that analysts develop their own style of memoing (Corbin & Strauss, 1990). Supporting the abstraction process, these annotations are considered relevant for driving creativity, thus discovery and definition of concepts (Rodon & Pastor, 2007).

2.2.3 Qualitative Theory - Validity and Relevance

Evaluating quality of theory as result of QDA is difficult in many ways. The various approaches considered qualitative analysis do not only differ in processes, but also are characterized by different goals. Adding to that the findings are highly related to context and derived from limited amounts of sources. This makes it particularly difficult to apply the measures of validity and relevance, typically used for quantitative analysis (Eastwood & Sheldon, 1996). Even though certain scholars state validity and relevance to be universal criteria (Atkinson & Hammersley, 1994), differences in philosophical and theoretical orientations prohibit application of standardized measures (Patton, 1990).

Scholars across the field have tried to define what good qualitative research is but could not establish consensus upon such criteria (Sandelowski & Barroso, 2008; Morse et al., 2002). Indeed it has been argued, that the vast amount of publications defining quality has in fact obfuscated this topic (Field & Morse, 1985). The problems relate to a wide range of aspects, beginning with philosophical stance and role of evaluators, spanning over data collection, sampling and methods of inquiry and reaching up to applicability of results (Meyrick, 2006). The methodological pluralism complicates quality assessment (Easterby-Smith et al., 2007) and the different goals obstruct comparability. (Yin, 2014; Thomas, 2006).

Common criteria assessing trustworthiness are credibility, transferability, dependability and confirmability (Guba, Lincoln, & others, 1994). Credibility of research can be established, following methodological procedures and adhering to the evidence in the data (Yin, 2011; Eisenhart, 2006). Multiple analysts, statistical testing and confirmatory studies support transferability of results (Belk, 2007; Fournier, 1998). Closely related to reliability, dependability is considered as stability of findings (Bitsch, 2005; Rolfe, 2006). Finally, confirmability is achieved by accessible presentation of findings (Lincoln & Guba, 1986; Wholey, Hatry, & Newcomer, 2010). Ellis, Strauss, & Corbin (1992) state that quality can be assessed by considering three additional aspects. First the theory itself should be evaluated in terms of fitting the substantive area. Further, it should be understandable and relevant to

participants, while provide enough abstraction for generalization. Finally, quality is characterized by how much control can be achieved, applying the theory to reality. However, this might be limited to the social context and conditions (Böhm, 1994).

Summing up, in order to evaluate analyses claiming to produce good theory, four problem domains need to be considered. These are suited data sources, the credibility and value of the theory, the correct application of methods and finally, the empirical grounding of the results (Corbin & Strauss, 1990).

2.3 Research Question

Based on the lack of standardized measures for evaluating qualitative research, the need for a formal methodology can be derived. Focusing on GT and suggested processes, which produce findings potentially suited for testing (Glaser, 1993), this research project transformed a given code system into an object-oriented domain model. Therefore it was necessary to develop, apply, and prove criteria which indicate quality. This lead to the question if GT can be efficiently used for domain modelling and if so, which metrics can assess the quality of such analysis.

2.4 Research Approach

The goal was to develop measures for evaluating the quality of a code system derived from qualitative analysis. The provided data was elaborated using MaxQDA (MaxQDA, 2015). It is one of many computer tools supporting qualitative analysis (Mey, Mruck, & Glaser, 2011). Despite its benefits to traceability and rigour of analysis processes, such software is limited in regard of evaluation, further characterized by specific data formats, prohibiting data interchange and follow-up processing (Franzosi, Doyle, McClelland, Putnam Rankin, & Vicari, 2013). The following graphic provides the meta-model of codings and their attributes.

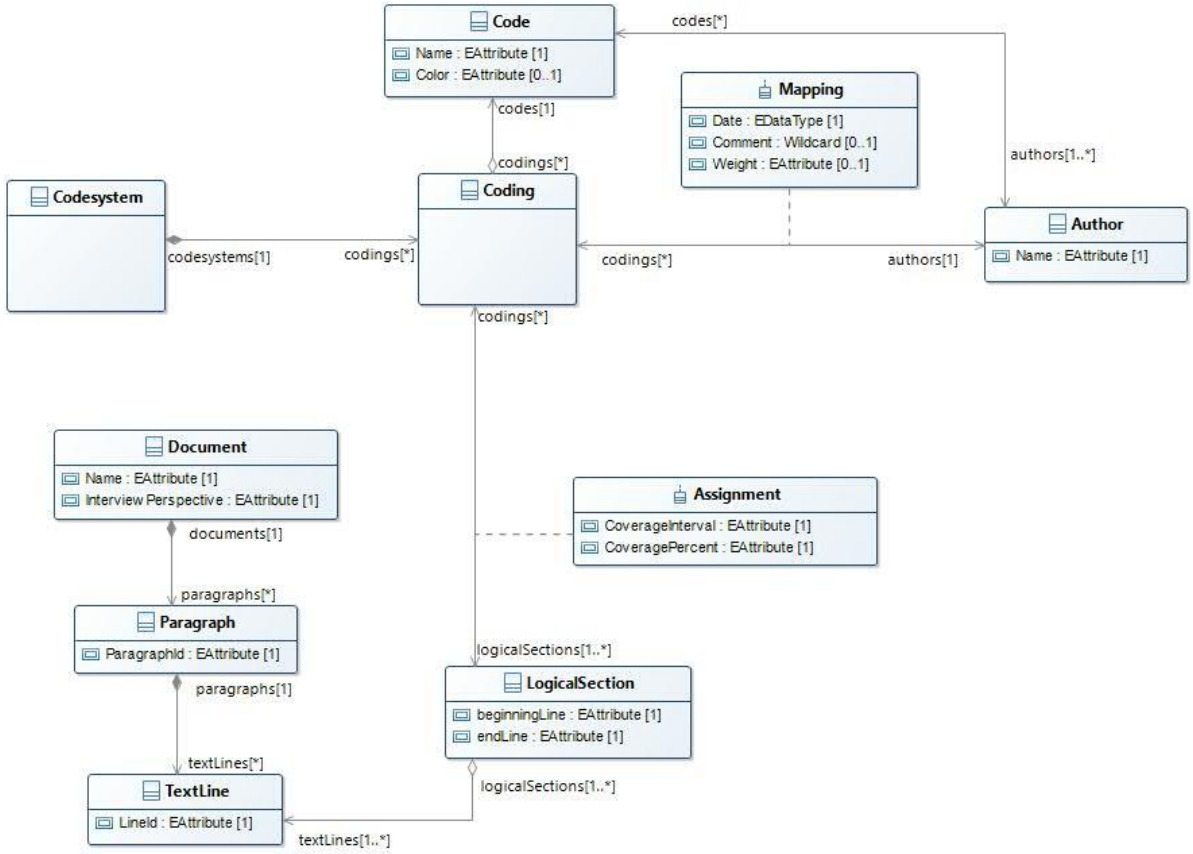


Figure 2: MaxQDA Coding Meta-Model

MaxQDA allowed for XML export and a JAVA algorithm was developed for transforming the code system into a formal model, in turn creating possibility for application of quality measures. In this context measurement can be understood as mapping from empirical evidence to a formal model, whereby a single measure is a number assigned to an entity by this mapping function in order to describe an attribute (Fenton & Pfleeger, 1998). The IEEE Standard 1061 states that an attribute is a measurable property of an entity and a quality factor is a type of management-oriented attribute of software contributing to its quality. Therefore a metric is a measurement function whose inputs are software data and its output is a numerical value that can be interpreted as degree, to which analyzed software possesses a given attribute affecting its quality (IEEE Computer Society, 2009).

In general a metric can be calculated by counting, matching, comparing and timing, respectively (Kaner, Member, & Bond, 2004). However, no quantitative data or weights could be extracted. Some color coding was applied to the code system, but it was not exhaustive enough for evaluation. In addition only five of 277 codes were annotated with additional information and only name and position in the code system could be used for analysis. Because such software metrics partially build upon object oriented concepts and information necessary for transforming the code system was not explicit, additional meta-information was necessary. Consequently the development of a memo format seemed feasible, to be exported along with the code system.

To develop a set of metrics, quality factors had to be defined (Kaner et al., 2004). In this context the GT coding paradigm provided aspects of high quality theory, which could be formulated into attributes. To prove the value of this annotation format, the evaluation was performed and its outcome was used to restructure the code system.

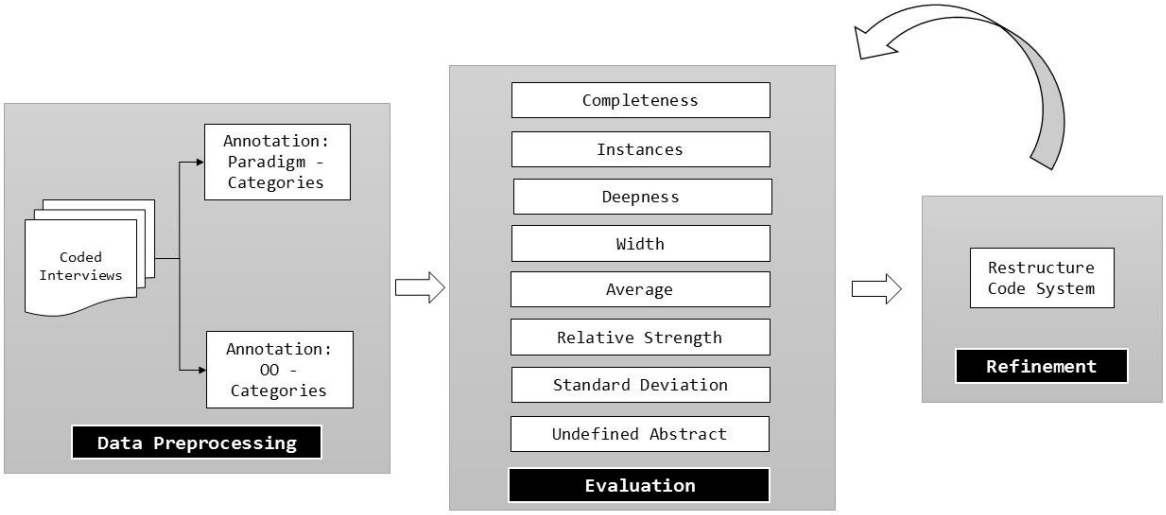


Figure 3: Research Process

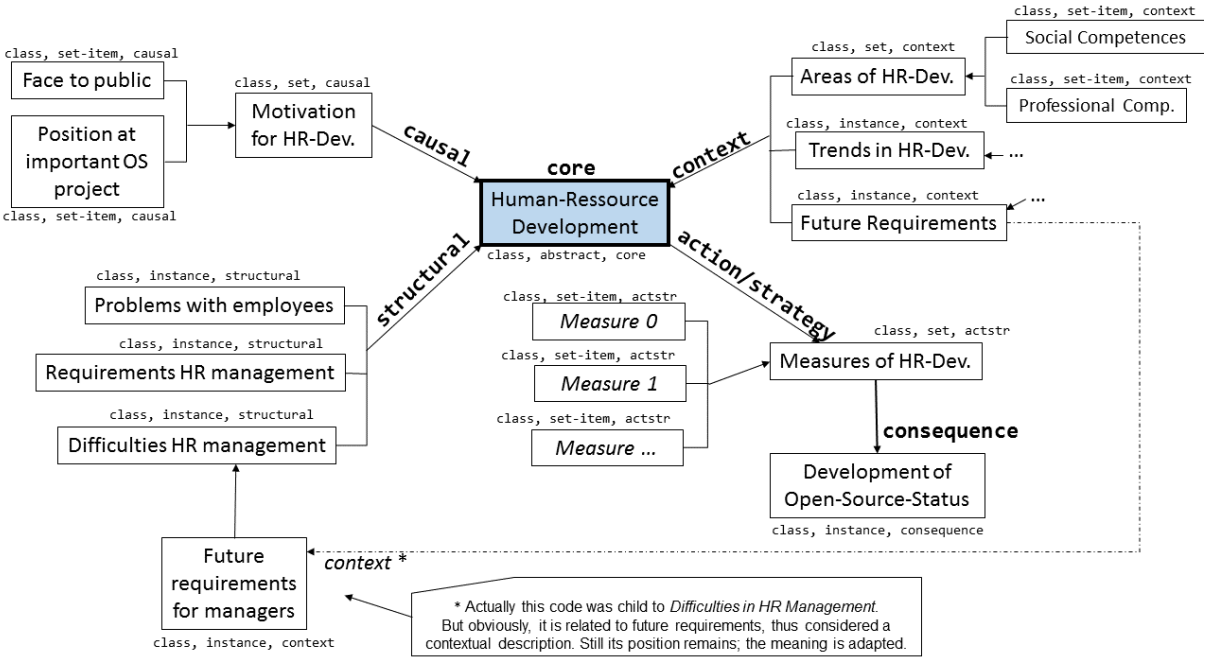
2.4.1 Annotation Format

According to (Corbin & Strauss, 2008) the coding paradigm is an analytical tool focused on supporting the emerging theory by integrating structure – that is the conditional context in which a phenomenon occurs. Describing the relations among concepts, its dimensions were applicable to externalize information. The suggested conceptual categories were added to individual codes as paradigm variable. In regard of object oriented implementation the codes were annotated with the basic concepts of *classes* and *attributes* for entity types. Further, to support domain modelling additional information was added to the codes in form of a *model* attribute. The model variable in combination with *domain=class*, was used to describes the class-entity as *abstract* or as *instance*. If codes were annotated with *domain=attribute*, the *model* attribute could be used to specify single properties or multiple characteristics of an object. Likewise the combination *domain=class* and *model=set/model=setitem* was used to describe containers.

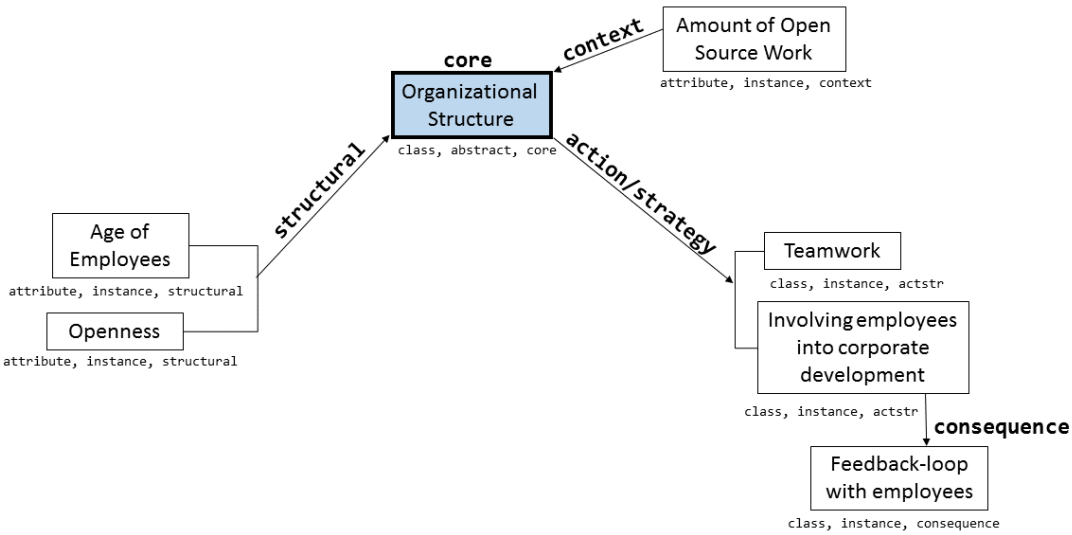
- `paradigm = { causal; core; context; structural; action/strategy; consequence}`
- `domain = { class; attribute }`
- `model = { abstract; instance; set; set-item}`

Figure 4: Annotation Format

The complete code system tree is provided in chapter 4.1 and the additional memo information can be found in chapter 4.2. Yet, to increase understanding of the annotations used, two examples are provided in the following pictures.



Conceptualization of evidence results in abstraction and ideally incidents of a phenomenon are related in the context of a core category (Glaser, 1993). However, the code system contained no declared core concept or phenomenon. Supported by the fact that eight top-level codes accounted for 94.17% of total mappings, these concepts were considered as *core* phenomenon, related to the paradigmatic instances of the subsequent codes. Afterwards the provided empirical evidence was analyzed to define the *paradigm* attribute of the child codes. In case an if-then-relationship was encountered the value *causal* was used. On the other hand influencing conditions were annotated with *structural*. The variable *context* described information characterized by time, location or distinct occurrence. Generally the paradigm attribute is handed on from parent to child. However, it can change when items are obviously related to another concepts as depicted above.



In regard of the object-oriented implications a *core* concept is supposed to be abstract and sufficiently conceptualized. Therefore the core codes were declared as *domain=class* and *model=abstract*. The annotation *domain=attribute* was used, when properties of the parent code were described. The attribute *model* was defined as *instance* when on the same level no similar information could be found. On the other hand *set* was used to mark items that can be seen as attribute lists or containers for objects, respectively. Accordingly the children of such codes were defined as *set-item*.

2.4.2 Quality Metrics

A theory can be considered of high quality when it is complete in terms of conceptualization, and elaborated concepts are highly related (Evans & John, 2013). Consequently, the code system was evaluated using the paradigm model similar to balanced scorecard, a performance measure in management science. By evaluating multiple values it is able to assess a well-rounded set of attributes (Kaplan & Norton, 1992). In this context paradigm instances were counted, for the entire model and for each *core* code individually, by traversing through its children.

The conceptual elaboration was considered complete, if all paradigm categories were encountered. Consequently, the metric *completeness* indicated that perspectives from all possible dimensions were taken into account during analysis. Further, the absolute amounts of paradigm *instances* served as additional criteria for model evaluation. These numbers revealed the sophistication of elaboration and further, how equally dimensions were considered when the concept was developed. Accordingly, for each core phenomenon the *deepness* was evaluated. The quality was considered higher in terms of abstraction, if a category had multiple levels defined in its subsequent hierarchy. In regard of understandability the *width* metric indicated how much information was provided by the children describing the phenomena.

Besides the ratios of declared paradigm instances were measured. The *average* of declared instances was calculated, for the entire model and for separated phenomena, respectively. To evaluate tendencies towards frequent declaration of single dimensions, the *relative strength* was calculated. This can be interpreted as influence of individual categories onto the core concept. In regard of a balanced model, equal distribution of categories was considered favorable and the *standard deviation* among paradigm instances was evaluated. At this point the object-oriented annotation came into play. To avoid adversely affecting quality, *set-items* of according types were grouped when comparing ratios among the entire code system. This was important when multiple similar types were encountered, because several properties or classes are actually beneficial to understanding. Finally, codes defined *abstract* demand for non-abstract instances in regard of correct domain model implementation. Consequently the amount of *abstract codes without defined instances* was counted. The following table provides overview over the quality metrics.

Metric	Value	Meaning
completeness	Percent (0% - 100%)	Considered dimensional perspectives
instances	Integer (0 – *)	Amount of defined paradigm instances
deepness	Integer (0 – *)	Levels of hierarchy for core categories
width	Integer (0 – *)	Range of describing information
average	Double (0 – *)	Average of instances declared
relative strength	Double (0 – 1)	Influence of individual dimension
standard deviation	Double (0 – *)	Distribution among paradigm instances
abstract	Integer (0 – *)	Amount of abstract objects lacking instances

Table 1: Defined Quality Metrics

2.4.3 Used Data Sources

The data used for this research project was a code system inductively developed by other analysts by coding three practitioner interviews. Such method is particularly useful for generating novel findings of high importance in specific contexts (Eisenhardt, 1989). The motivation for conducting the analysis was accessing experience of different stakeholders (Donzelli & Bresciani, 2004), more precisely a developer, project manager and human resource manager in the context of Open Source Software (OSS). The interviews were conducted by Prof. Riehle, a practitioner of OSS and Prof. Kimmelmann from the field of Human Resource Management. The exploratory interviews were transcribed and subsequently coded.

The interview-based case analysis resulted in 278 categories mapped to 446 text segments. Two codes marked introductory sections and were ignored for quality evaluation in this project. The preceding analysis was performed with MaxQDA and the data provided in corresponding format. Accordingly, the code system was a tree-model with 18 concepts located on the top-level. Further the interview-transcripts were conducted in German language as were the developed concepts. Moreover, the code system was provided together with the three interviews.

2.5 Research Results

2.5.1 Metrics for Original Code System

The annotation of the code system resulted in 18 core codes and 258 related concepts. The prior qualitative study was rated with 100% completeness. Paradigm instances and their influence after normalization are listed in the table below.

		Caus	Struct	Cons	Act/Str	Context	Total	Complete-ness	Average
Code System (before)	in-stances	38	75	23	75	47	258	100%	51.60
	rel. str.	0.29	1	0	1	0.46	standard devia-tion = 2.87		
Code System (excl. set-item) (before)	inst.	15	39	15	30	14	113	100%	22.60
	rel. str.	0.04	1	0.04	0.2	0.00	standard devia-tion = 0.46		

Table 2: Metric Calculation - Original Code System (before)

Comparing the instances revealed that structural conditions and concepts related to action/strategy were particularly high, while less causal conditions or consequences were found. However, the average of instances declared was quite good, with 51.6 declarations per category, taking into account that 258 instances were declared for five categories. Applying set-item reduction was able to increase the influence of the dimensions consequence and context, while decreasing the oversized strength of action/strategy. Further, the distribution prior accounting for 2.87 standard deviation was improved to 0.46. Yet, the importance of causal conditions was reduced and contextual aspects lost influence.

With regard to individual concepts and the defined memo format, codes declared as paradigm=core were annotated as *domain=class* and *model=abstract*, respectively. Seven of these had no concrete instances defined, what is considered inadequate for domain modelling. Adding to that a concept without provided description is more difficult to understand and has unsatisfying relation to the model. According to GT contextual relation and abstract conceptualization is key to high quality models (Glaser, 2002). As results, the codes lacking instances were definite candidates for refinement.

- OSS
- Überprüfung Verhalten in Mailinglisten (english: Checking behaviour in mailing lists)
- Nach Vorstellung der Kollegen/des Teams (english: According to colleagues/team)
- Passung ins Team (english: Fitting the team)
- Branchenkenntnisse (english: Industry Knowledge)
- Produktinnovation (english: Product Innovation)
- Produkte (english: Products)

Table 3: Abstract Codes without Defined Instances

Another six codes had instances defined, yet the low amount of dimensional categories resulted in poor quality metrics. Therefore, these codes were also candidates for refinement. Table below provides overview over these codes with their corresponding calculated metrics.

		Causal	Struct	Con-seq.	Act/Strat.	Context	Total	Complete	Average
Motivation zu Open Source (before)	in-stances	0	0	0	3	0	3	20%	0.60
	relative strength	0.00	0	0	1	0.00	standard deviation = 0.63	deep=2	width=2
Bedeutung Open-Source für das Unternehmen (before)	in-stances	0	0	1	1	0	2	40%	0.40
	relative strength	0.00	0	1	1	0.00	standard deviation = 0.39	deep=1	width=2
Kompetenzentwicklung durch Open Source Tätigkeit (before)	in-stances	0	0	1	0	0	1	20%	0.20
	relative strength	0.00	0	1	0	0.00	standard deviation = 0.31	deep=1	width=1
Projektzuweisung von Mitarbeitern (before)	in-stances	1	0	0	1	0	2	40%	0.40
	relative strength	1.00	0	0	1	0.00	standard deviation = 0.38	deep=1	width=2
Mitarbeitermerkmale (before)	in-stances	2	6	1	3	1	13	100%	2.60
	relative strength	0.2	1	0	0.4	0.00	standard deviation = 0.88	deep=3	width=9
Organisationsstruktur (before)	in-stances	0	2	1	2	1	6	80%	1.20
	relative strength	0.00	1	0	1	0.00	standard deviation = 0.37	deep=2	width=5

Table 4: Codes with Low Metric Values

The remaining top-level concepts were well elaborated and the paradigm analysis did not provide sufficient reason for remodeling codes. In most cases, all possible dimensions were considered and populated by enough instances so that the core concept could be understood. Motivation had no activities or strategies defined, but the concept itself can be interpreted as element to such. Further, the *Development Process* had no causal conditions defined. Based on multiple iterations through the interviews, the reason was found to be the concept analysis being more focused on correlated aspects and characteristics of the process than why such occurs, or what consequences might appear. Following graphics provide the counted paradigm instances and the derived strength of the individual dimensions, for the rather strong concepts.

		Causal	Struct.	Con-seq.	Act/ Strat.	Context	Total	Complete	Average
Personalentwicklung (before) = (after)	instances	3	6	2	21	17	49	100%	9.80
	relative strength	0.05	0.21	0	1	0.79	standard deviation = 3.79	deep=10	width=19
Einstellungsprozess (before) = (after)	instances	27	16	4	18	14	79	100%	15.80
	relative strength	1.00	0.52	0	0.61	0.43	standard deviation = 1.10	deep=19	width=13
Entwicklerkarriere (before) = (after)	instances	2	15	10	13	10	50	100%	10.00
	relative strength	0.00	1	0.62	0.85	0.62	standard deviation = 0.63	deep=12	width=12
Motivation (before) = (after)	instances	3	2	1	0	1	7	80%	1.40
	relative strength	1.00	0.5	0	0	0.00	standard deviation = 0.48	deep=1	width=7

Table 5: Codes with Good Metric Values

Though, one exception must be noted. The code *Entwicklungsprozess* (english: Development process), despite being well elaborated, has been subject to change. Due to another code being attached to it, the metrics before and after refinement differ. For the purpose of integrity, the values are provided below.

		Causal	Struct.	Con-seq.	Act/ Strat.	Con-text	Total	Complete	Average
Entwicklungsprozess (before)	instances	0	28	2	13	3	46	80%	9.20
	relative strength	0.00	1	0	0.42	0.04	standard deviation = 3.19	deep=14	width=14

Table 6: Code Entwicklungsprozess/Development Process

In conclusion, the quality evaluation provided good perspective on the concepts and their conceptualization. Missing dimensions or unsatisfying metric-values indicated reason for change.

2.5.2 Refinement of Code System

Isolated codes without instances were relocated for more meaningful relationships and in order to enhance the explanatory strength of the model. Likewise, when relocating codes, the particular week concepts were taken into consideration. While the detailed explanations are provided in chapter 3.4 brief summaries are described in the following.

Bedeutung Open-Source für das Unternehmen (english: Importance of OSS for the company), *Motivation for OSS* and *Kompetenzentwicklung durch Open Source Tätigkeit* (english: *Skill development by OSS*) were evidently related to the concept *OSS* and added to the concept as instances.

According to the team was found an instance of *Fitting the team*. Inspecting the interviews revealed a similar code Passung ins Team nach Vorstellung des Managers (english: *Fitting the team according to the manager*) in the model, child to a good described concept called *Einstellungskriterien* (english: *Criteria for Hiring*). The poorly described codes were added to the latter one.

Checking behaviour in mailing lists was added to *Projektzuweisung von Mitarbeitern* (english: *Project-Assessment of employees*).

Industry Knowledge was found to be an attribute to software developers and added to *Mitarbeitermerkmale* (english: *Employee Characteristics*).

Products was added as consequence to *Product Innovation*, while that was attached to *Organistationsstruktur* (english: *Organizational structure*).

To prove this analysis potentially provides increased quality to the model, another paradigmatic evaluation was performed, after rearranging the code system. Repeating the paradigm analysis then resulted in 268 instances, since ten concepts prior defined as core, were now used as categories. In general this led to three more causal and contextual conditions, while increasing the count of structural influences and consequences by two instances. Accordingly the relative influence of causal and context dimension was improved. The average declaration improved from 51.6 to 53.6, but on the other hand standard deviation increased from 2.87 to 3.73. Relative strength for causal increased to 0.79 and context influence improved to 0.9. Reducing the model by set-items, revealed that influence of causal and consequential criteria both increased to 0.11 and the action/strategy dimension improved from 0.2 to 0.7037, respectively. However, the increase in standard deviation, from 0.46 to 5.3 was also significant.

		Causal	Struct.	Con-seq.	Act/ Strat.	Context	Total	Complete	Average
Code System (before)	instances	38	75	23	75	47	258	100%	51.60
	relative strength	0.29	1	0	1	0.46	standard deviation = 2.87		
Code System (after)	instances	41	77	25	78	47	268	100%	53.60
	relative strength	0.79	1	0	1	0.90	standard deviation = 3.73		
Code System (excl. set-item) (before)	instances	15	39	15	30	14	113	100%	22.60
	relative strength	0.04	1	0.04	0.2	0.00	standard deviation = 0.46		
Code System (excl. set-item) (after)	instances	17	41	17	33	14	122	100%	24.40
	relative strength	0.11	1	0.11	0.7037	0.00	standard deviation = 5.30		

Table 7: Metric Calculation - Original Code System (after)

Be refining the model, several improvements could be achieved. Compared to the changes in the entire code system, the increased quality of the individual concepts is more evident. *OSS* became an understandable concept with 80% completeness and five levels of abstraction. *Employee Characteristics* was added a causal concept, increasing the influence of this dimension. *Organizational structure* gained improvement in regard of structural (before: 1, after 0.5) and consequential (before: 0, after 0.5) influence. Finally *Development Process* was augmented by one causal criteria, however, the strength three categories increased. These were consequence (before: 0, after: 0.04), action/strategy (before: 0.42, after: 0.56) and finally context (before: 0.04, after: 0.07).

		Causal	Struct.	Conseq.	Act/ Strat.	Context	Total	Complete	Average
OSS (before)	instances	0	0	0	0	0	0	0%	0.00
	relative strength	0.00	0	0	0	0.00	standard deviation = 0.0	deep=0	width=0
OSS (after)	instances	1	1	3	4	0	9	80%	1.80
	relative strength	0.00	0.00	0.67	1.00	0.00	standard deviation = 1.05	deep=5	width=3
Mitarbeiter-merkmale (before)	instances	2	6	1	3	1	13	100%	2.60
	relative strength	0.2	1	0	0.4	0.00	standard deviation = 0.88	deep=3	width=9
Mitarbeiter-merkmale (after)	instances	2	7	1	3	1	14	100%	2.00
	relative strength	0.17	1.00	0.00	0.33	0.00	standard deviation = 0.97	deep=3	width=10
Organisations-struktur (before)	instances	0	2	1	2	1	6	80%	1.20
	relative strength	0.00	1	0	1	0.00	standard deviation = 0.37	deep=2	width=5
Organisations-struktur (after)	instances	0	2	2	3	1	8	80%	1.60
	relative strength	0.00	0.50	0.50	1.00	0.00	standard deviation = 0.53	deep=3	width=6
Entwicklungs-prozess (before)	instances	0	28	2	13	3	46	80%	9.20
	relative strength	0.00	1	0	0.42	0.04	standard devia-tion. = 3.19	deep=14	width=14
Entwicklungs-prozess (after)	instances	1	28	2	16	3	50	100%	10.00
	relative strength	0.00	1.00	0.04	0.56	0.07	standard deviation = 3.61	deep=15	width=14

Table 8: Metric Calculation Conceptual Improvements

The conceptual quality of the code system could be increased by augmenting concepts with further categories or relating codes in a more meaningful way.

2.6 Results Discussion

Multiple quality metrics had been successfully derived and applied for evaluating a code model resulting from inductive analysis. Annotating additional attributes to codes allowed for externalizing meta-information, which in turn could be utilized for object-oriented domain modelling. As shown above, the conceptual strength of the theory could be measured and suggestions for refinement derived.

However, the actual implementation of the JAVA code for metric calculation, revealed that not all attributes were particularly useful. While first, aspects of and differences in *domain=class* and *domain=attributes* were evaluated, it turned out that such definitions are highly subjective to the researchers liking. Also the high abstraction of concepts led to vast amounts of possibilities for object-orientation implementation, prohibiting a meaningful measurement.

Further, in earlier iterations of program-code development, for each *class* the *attribute* instances were counted in order to assess explanatory strength of each object. However, most codes were found to be classes, thus leaving only unsufficient amounts of attributes left to be assigned. Yet, considerations about such potentials for providing understanding led to the metrics *width* and *deep*. These were found to be useful for the same reason indeed.

In regard of evaluating ratios, the *relative strength* and the related metric *standard deviation* at first seemed to be without practical use. But when code system refinement was performed, the more populated a concept became, the more meaning could be derived. The provided examples partially depict this effect, but still after just one step of refinement the practical value is considered low. Nevertheless, multiple iterations of refinement combined with more instances will certainly result in relevant metric values.

Consequently, the *paradigm* attribute was the most significant metric. Its dimensions were beneficial to access the information in the code system. Another important aspect was the potential for counting individual concept instances. The correlated *completeness* measure highlighted missing dimensions of core codes and, combined with the absolute number of *instances*, code system refinement became particularly easy and efficient.

Summing up, the memo-annotation and application of the suggested metrics could improve the quality of the model. The refinement increased the conceptualization and supported understanding of several concepts. Moreover the quality metrics provided guidance for developing a balanced theory.

2.7 Contributions to Qualitative Research

The result of this thesis contributes to the field of qualitative research on multiple ways. Proposed criteria for assessing good theory are data sources, credible and valuable theory, correct application of methods and finally the grounding in the data (Lincoln & Guba, 1986).

In general using CAQDAS already positively influences the handling of complex empirical data (Miles & Huberman, 1994). Supplementary, applying a formal or semi-formal memo

annotation increases comparability among different data sources and helps the researcher to decide upon its adequacy.

Considering the theory itself, results presented in a canonical format allow to bridge the gap from qualitative to quantitative studies (Eisenhardt & Graebner, 2014). Applying metrics for conceptual evaluation will indicate sophisticated abstraction and consequently make the theory measurable (Glaser, 1993).

The proposed method of quality assessment makes the analysis process more transparent. Further, developing good styles for annotating memos is considered key to high quality GT (Elliott & Course, 2005). In addition literature states that memoing should be performed from the very beginning of the analysis (Dick, 2005). The results of this research project provided evidence, that memoing can support the development of categories and improvement of theories.

Finally, the grounding of findings in empirical data characterizes good theory (Goldkuhl & Cronholm, 2010). The proposed method positively contributes to that. At any time during the analysis process, full traceability from data to elaborated concepts can be provided. On top of this, refinements or changes to the model can be performed, without losing the link to empirical evidence.

Summing up, the suggested evaluation method supports the four common criteria for assessing quality of QDA. It improves the process of data selection, increases comparability among results, makes the analysis more transparent and rigorous and finally provides full traceability at any point during analysis.

3 Elaboration of Research Chapter

3.1 GT – Dispute about the Coding Paradigm

The popularity of GT has resulted in various approaches and different kinds of processes. However, for this research two of many methods are most significant, both proposed by the original founders. Despite the fact that Glaser and Strauss introduced grounded theory together in 1967, their approaches dispersed over time. The main reason for dispute was the suggestion of the coding paradigm by Corbin and Strauss (1990).

In response Glaser (1992) harshly criticized the paradigm, to be an distortion of the original GT goal, resulting in forcing of categories, rather than allowing for emergence, what was confirmed by several authors (Kendall, 1999; Urquhart, 2000; Walker & Myrick, 2006). Ironically, Glaser himself suggested coding families which actually include the dimensions of the coding paradigm (Glaser, 2008). Defending their recommendation, Corbin and Strauss argue, that the vague framework should rather be considered as guidance for incorporating a holistic view onto the examined phenomenon what on the other hand is equally supported by several scholars (Allen, 2011; Bitsch, 2005; Evans & John, 2013; Strübing, 2008).

When comparing the Glaser and Strauss approach, still both methods are characterized by the same characteristics. These are parallel processes of systematic data gathering, its reflection and the theory emerging from data evidence. In the end both ways are compatible to each other and focus on the same aspects of the GT, thus integrating benefits of quantitative methods with qualitative interpretations. (Mey et al., 2011)

3.2 Building Theories from Cases

Analysing evidence from instances of a phenomenon with focus on creating theoretical constructs or mid-range theories is called building theories from cases. Compared to mainstream qualitative research, which is highly descriptive and emphasizes the social construction of reality, this approach differs in terms of activities, goals and epistemology. It is characterized by a rather positivist stance and further can be considered more objective. Instead of isolating the phenomenon from its occurrence, case studies focus on the rich, real-world context where the incident can be observed. (Eisenhardt & Graebner, 2014)

The central notion is to use case evidence to inductively develop a theory, being emergent due to its grounding in the data. Elaboration processes are characterized by pattern recognition among constructs evident within data. Key to this method is the replication logic and theoretical sampling of evidence. While single cases are independent and distinct experiments used for inductive theory development, multiple sources are discrete experiments, in turn serving as replications, contrasts and extensions to the emerging theory. (Yin, 2014)

Another aspect of this approach is the use of terminology describing the individual process and its implications. However, various terms and labels can create confusion and consequently demand for precise language and description, making the inductive process transparent and understandable. Another challenge is that findings are constituted by rich qualitative details and cannot be tightly summarized. Since there are no accepted standard templates for writing or

presenting the theory, the analyst has to develop skills of presenting his findings in according ways. Further, interpreter bias or retrospective sense making pose a risk to the validity of results.

To support the quality of such analysis and its results, it is important to ensure that the emerging theory fully exploits all available evidence, while the process should be characterized by sophisticated research design. Rich and understandable presentations of evidence, thoughtful justification of theory building, theoretical sampling of cases and choosing sources, which limit informant bias ultimately constitute a valuable analysis. This analytical approach is characterized by replication logic and supports the evaluation of resulting theories by bridging the gap between qualitative and quantitative research. The use of interview data in combination with theoretical sampling provides great potential to detailed findings of human interactions in specific contexts. In conclusion, in the context of this research project the provided code system was considered a valuable data source.

3.3 General Inductive Approach

Subsequently, findings can be justified and defended by the underlying research goal. By developing a framework based on the underlying structures or processes evident in the data, reliability and validity can be established. While being consistent with the implications of qualitative research, this approach provides a more detailed set of processes for analyzing and reporting qualitative data. Key to this method is the establishment of clear links between the evaluation of research objectives and summaries of such raw data, ensuring transparency of the results.

Knowledge in regard of efficient and defendable procedures for analyzing qualitative data is less common, thus motivates this extension to qualitative research (Thomas, 2006). With regard to clarifying the implications of data reduction this method describes detailed processes of creating meaning in complex data. Key is the development of summary themes or categories from raw data.

Several analytical strategies guide the process. First, data analysis is guided by evaluation objects providing a focus or domain of relevance, instead of a-priori expectations about specific results. Consequently the inductive component is characterized by multiple readings and interpretations allowing the findings to emerge directly from the raw data. Further, the primary mode of analysis is coding, where the evaluator constructs key concepts and elaborates categories from empirical evidence, which are combined into a theory or framework. Since findings are the result of multiple interpretations, inevitably they are shaped by assumptions and experiences of the analyst and his decisions about what is important for the theory (Thomas, 2006).

Elaborated categories have certain features. They contain labels, a term used to refer to the category and possibly reflecting specific properties of such. The description of a category can be attached including key charts, scope and limitation. Another aspect is the associated data or mapped text section, explanatory illustrating meaning, relations or perspectives for the category. Thereby concepts might be linked by hierarchical tree diagrams, or be interrelated based on commonalities in meaning as well as assumed causal relationships. In the end category system are implemented into a theory or model.

The general inductive approach is quite similar to grounded theory, however it does not separate the processes of open and axial coding. Further, while grounded theory aims at discovering theories eventually presented as description, including themes or categories, this methodology is concerned with the analysis of core meanings in text, relevant to evaluation or the research objective. Therefore the result are categories presented with descriptions for the most important themes. In conclusion the general inductive approach builds upon implications of qualitative analysis and provides additional processes for the analysis. By defining concrete methods and analytic strategies for developing categories meaning can be derived from complex data and process transparency can result in good traceability of the findings. Finally trustworthiness of such results can be assessed using techniques related to qualitative research. (Lincoln & Guba, 1990)

3.4 Decisions for Code System Refinement

In this chapter detailed considerations and supporting evidence derived from the interviews are described, which lead to the refinements conducted in the code system.

The code *OSS* was mapped to two interview sections related to OSS development. Relocating this code into other categories seemed quite difficult, because it had no attributes or meanings declared, which in turn could increase the conceptualization. However, on the top level three other core concepts were found, obviously related to the OSS domain. These were *Bedeutung Open-Source für das Unternehmen* (english: *Importance of OS for the company*), *Motivation zu Open Source* (english: *Motivation for OSS*) and *Kompetenzentwicklung durch Open Source Tätigkeit* (english: *Skill-Development by OSS activity*). These codes had paradigm concepts defined, however the dimensions were poorly populated and prohibited deep understanding, thus were considered as codes of lower quality.

Motivation for OSS was characterized by three codes describing strategic aspects of OSS. *Importance of OSS for the company* had two dimensions defined, each describing one consequence and one action/strategy. Further, *Skill-Development by OSS* had only one consequential aspect defined - obviously because it was a direct consequence of OSS. Accordingly, those codes were relocated and attached to *OSS*. As a result these prior isolated concepts were then related to each other. So, the annotation *domain=class, model=abstract, paradigm=core* had to be changed accordingly. *Motivation for OSS* was found a causal condition because its existence is supposed to result in the respective phenomena and hence it was defined as *domain=class, model=instance, paradigm=causal*. On the other hand *Skill-Development by OSS activity* had a child *Interkulturelle Kompetenz* (english: *Intercultural competence*) and was conceptualized as consequence of *OSS*, therefore declared *domain=class, model=instance, paradigm=consequence*. Finally, *Importance of OSS for the company* described structural circumstances of the core concept, eventually resulting in the annotation *domain=class, model=instance, paradigm=structural.*

The code *Fitting the team* had no further specified characteristics. Further, *Nach Vorstellung der Kollegen/des Teams* (english: *According to colleagues/the team*) was a characteristic of such condition, also having no paradigmatic dimensions defined. The analysis of the mapped interview sections provided strong evidence that team-fit was positively correlated to the hiring

of developers and both codes heavily related. Contributing to that, there was a similar code called *Fitting the team according to the manager*. Based on these findings, matching a group from team- or management-perspective can be seen as attribute necessary for employment. Also both provide more details for the rather abstract concept of suiting a team of developers. As result, *Fitting the team* was attached to *Criteria for hiring* with the annotation *domain=attribute, model=set, paradigm=causal*. The other two codes were repositioned as children to *Fitting the team* both defined as *domain=attribute, model=setitem, paradigm=causal.*

The top-level code *Überprüfung von Verhalten in Mailinglisten* (english: *Checking behavior in mailing lists*) was neither related to any concept, nor was it further described by attributes or theoretical implications. The provided transcript revealed that this was an activity for assessing employees to appropriate projects, therefore being a strategic consideration. The code was annotated with *domain=class, model=instance, paradigm=actstr* and repositioned. It was defined as child and attached to *Projektzuweisung von Mitarbeitern* (english: *Project-Assessment of employees*), which indeed was poorly described, providing understanding from two dimensions only. Further, *Project-Assessment* was attached to *Developnent Process domain=class, model=instance, paradigm=actstr,*

The code *Industry Knowledge* was another isolated concept, but evidence strongly suggested this conception to be an attribute for describing single developers. Since it was neither a premising condition for hiring nor did it result in specifically mentioned employee properties, it was considered a structural criteria influencing various aspects of developers and therefore defined as *domain=attribute, model=instance, paradigm=structural* while being added as child to *Employee Characteristics*.

Analyzing the transcript section mapped to the code *Product Innovation*, various aspects surrounding product development were found. However, despite mentioning activities or processes, the mapped section provided more focus on organizational aspects including management, teams and departments. Similar, *Products* could be seen as consequence of product innovation, supported by a text segment containing multiple aspects of correlating organization and product development. Consequently, *Product Innovation* was defined as *domain=class, model=instance, paradigm=actstr* and related to *Organisational Structure*, while *Product* was redefined *domain=class, model=instance, paradigm=consequence* as child to *Product Innovation*.

4 Appendix

4.1 Code System Tree-Model

__ 279 OSS
__ 236 Überprüfung Verhalten in Mailinglisten
__ 235 Nach Vorstellung der Kollegen/des Teams
__ 234 Passung ins Team
__ 228 Bedeutung Open-Source für das Unternehmen
____ 229 Einfluss auf Produkte nehmen
____ 224 Open-Source-Engagement führt zu (gesteigertem) Kundenvertrauen
__ 172 Branchenkenntnisse
__ 106 Kompetenzentwicklung durch Open Source Tätigkeit
____ 107 Interkulturelle Kompetenz
__ 18 Mitarbeitermerkmale
____ 218 Unterschiedliche Charaktere
_____ 220 Umgang mit Publizititität
_____ 36 Angst vor Publizität
_____ 219 Extrovertierte Fachexperten
_____ 24 Nach kultureller Diversität
____ 217 Intrinsische Motivation für OS-Arbeit
____ 187 Unmotivierte Entwickler leisten keine gute Arbeit
____ 182 Angst vor Inkompetenz bei Minimierung der Entwicklertätigkeit
____ 166 Geringe Fluktuation
____ 165 Verhalten in Loyalitätskonflikten
____ 105 Langsames Warmwerden mit Menschen
____ 102 Mitarbeiterloyalität zum Unternehmen
____ 101 Flexibilitätswunsch
__ 62 Personalentwicklung
____ 169 Anforderungen an die Personalverwaltung
____ 113 Bereiche der Personalentwicklung
_____ 248 Soziale Kompetenzen
_____ 114 Technische Kompetenzen
____ 112 Schwierigkeiten der Personalentwicklung
_____ 170 Zukünftige Anforderungen an Manager
____ 109 Maßnahmen der Personalentwicklung
_____ 263 Probleme explizit machen als Projektleader
_____ 227 Austausch mit anderen Kollegen
_____ 226 Sprachkurse
_____ 160 Maßnahmen gegen Burnout
_____ 156 Gespräche bei gemeldeten Problemen
_____ 155 Möglichkeit zum Ausprobieren eigener Projekte
_____ 145 Beobachtung der Arbeitsleistung
_____ 99 Anreize zur Mitarbeitermotivation
_____ 116 Mitarbeitergespräch
_____ 115 Selbststudium
_____ 111 Individuelle Maßnahmen
_____ 110 Interkulturelle Trainings
_____ 108 Kein OS-spezifisches Programm
_____ 63 Mentoring
_____ 64 Pair Programming
_____ 65 Regelmässiges Feedback
_____ 66 Schulung
_____ 74 Organisation von SUSE Konferenz
_____ 73 Entsenden auf Konferenzen

____ 84 Trend in der Personalentwicklung
_____ 249 Kommunikationsbarrieren abbauen durch persönliche Treffen
_____ 85 China holt auf
____ 77 Zukünftige Anforderungen
_____ 256 Fortführung technischer Kompetenz
_____ 245 Gesteigerte Sozialkompetenzen
_____ 257 Einfühlungsvermögen
_____ 253 Kommunikationskompetenzen
_____ 255 Englische Sprachkompetenzen
_____ 254 Feedback konstruktiv formulieren
_____ 246 Erhöhter Wirkungskreis
_____ 247 Zielgruppenorientierte Kommunikationskompetenz
_____ 171 Web Development
_____ 88 Hart-im-Nehmen-Sein
_____ 87 Bereitschaft zu Sichtbarkeit
_____ 86 Open-Source-Erfahrung
_____ 90 Demonstrierte technische Kompetenz
_____ 89 Als Contributor
____ 75 Motivation zur Personalentwicklung
_____ 78 Gesicht nach Draussen
_____ 76 Positionierung in wichtigem Open-Source-Projekt
____ 72 Entwicklung von Open-Source-Status
____ 67 Probleme mit Mitarbeitern
__ 25 Motivation zu Open Source
____ 223 politische Motivationen
_____ 221 Sendungsbewusstsein
_____ 222 Idee der demokratischen Software
__ 23 Einstellungsprozess
____ 242 Einstellungsgründe
_____ 241 Unternehmensmarketing durch Einstellung von Personen
_____ 240 Strategische Einflussnahme durch Einstellung
____ 33 Bewerber-Assessment
_____ 258 Persönliches Treffen zur Feststellung der Kompatibilität
_____ 173 Stellenschaffung für Rockstars
_____ 146 Probleme des Assessments
_____ 61 Entscheidungsfindung im Assessment
_____ 147 Vorbesprechungen zwischen Personen die einstellen
_____ 143 Referenzen
_____ 142 Fachartikel
_____ 141 Öffentliches Portfolio begutachten
_____ 135 Rollenspiele
_____ 134 Fachliche Arbeitsprobe
_____ 92 Teambasierte Entscheidungsfindung
_____ 60 Aufwand für Assessment
_____ 59 Kommunikationsfähgikeit
_____ 31 Dokumentierte Open-Source-Erfahrung
____ 30 Einstellungskriterien
_____ 237 Persönliche Kontakte im Vorfeld (Vitamin B)
_____ 238 Einfluss in der Community
_____ 243 Commit-Rechte
_____ 140 Passung ins Team nach Vorstellung des Managers
_____ 138 Interkulturelle Kompetenzen
_____ 131 Personale Kompetenzen
_____ 136 Bereitschaft in virtuellen Teams zu arbeiten
_____ 133 Menschliche Kompatibilität
_____ 132 Anpassungsfähigkeit

_____ 45 Technische Kompetenzen
_____ 52 Umsetzung von Feedback
_____ 48 Architekturkompetenz
_____ 46 Programmierfähigkeit
_____ 44 Soziale Kompetenzen
_____ 53 Kommunikationsfähigkeit
_____ 233 Einhaltung sozialer Kommunikationsregeln (Kein Arschloch)
_____ 208 Umgang mit unterschiedlichen Kommunikationsstilen
_____ 209 Dolmetscher-Rolle
_____ 54 Schriftliche Kommunikationsfähigkeit
_____ 56 Bugtracker
_____ 55 E-Mailverkehr
_____ 51 Kritikfähigkeit
_____ 50 Hilfsbereitschaft
_____ 40 Teamfähigkeit
_____ 49 Umgang mit Problemen
_____ 47 Bereitschaft sich auf Vorgaben einzulassen
_____ 43 Vorhandene Projekte
_____ 41 Englische Sprachfähigkeiten
_____ 39 wie sie an Aufgaben rangehen
_____ 38 Offenheit für Neues
_____ 37 Lernfähigkeit
_____ 32 Open-Source-Erfahrung
_____ 35 Durch passive Teilnahme am Open Source
_____ 34 Durch aktive Teilnahme an Open Source
_____ 58 Involvierung in firmenfremde Projekte
_____ 57 Involvierung in Firmeneigene Projekte
____ 29 Entwicklerrekrutierung
_____ 176 Hohe Vorqualifikation im OS
_____ 122 Probleme der Rekrutierung
_____ 277 Unterschiedliche Probleme international
_____ 276 Begrenztes Budget
_____ 130 Schnelligkeit notwendig
_____ 82 Mangel an qualifizierten Bewerbern
_____ 273 Gründe für Mangel an Bewerbern
_____ 275 Persönliche Motivation notwendig
_____ 274 Hoher Leistungsdruck durch Vergleichbarkeit
_____ 83 Frauenmangel
_____ 91 Eingeschränkte Bewertungfähigkeiten
_____ 121 Rekrutierungsprozess
_____ 239 Über Konferenzen
_____ 175 über Ausbildungsplätze
_____ 129 Über die Uni
_____ 128 Über Headhunter
_____ 127 Über eigene Website
_____ 126 Über Jobsuchmaschinen
_____ 125 Über Social Networks
_____ 124 Über OS-Konferenzen
_____ 123 Werkstudenten
_____ 96 Quereinsteiger
_____ 120 Über OS-Community
__ 15 Entwickler-Karriere
____ 188 Open-Source-Karriere
_____ 201 Reputationsaufbau
_____ 266 Projekteinstieg
_____ 192 Open-Source-Karriere-Status

_____ 189 Committer-Status
_____ 190 Maintainer
_____ 193 Foundation-Mitglied
_____ 194 Projektmanagement-Komitee-Mitglied
____ 144 Unterstützung Open Source Karriere
_____ 150 Finanzielle Unterstützung
_____ 151 Zeitliche Unterstützung
_____ 152 Interessen-Aufgaben-Matching bei Zuteilung auf OS-Projekte
____ 70 Bedeutung von Open-Source-Status
_____ 244 Auswirkungen auf Gehalt
_____ 162 Erhöhte Unabhängigkeit der Selbstbestätigung vom Arbeitgeber
_____ 161 OS-Schlüsselposition führt zu höherem Gehalt
_____ 71 Bedeutung von Open-Source-Rockstars
___ 178 Unternehmensinterne Karrierepfade
____ 271 Nominierungsbasierte Positionsvergabe
____ 268 Gleichberechtigung von Fach- und Managementkarriere
_____ 270 Wertschätzung der Facharbeit
____ 93 Motivation sich weiterzuentwickeln
____ 16 Verharren in der Fachkarriere
____ 20 Interner Stellenwechsel
____ 80 Neue Rollen durch Open Source
_____ 79 Gesicht nach Draussen
____ 103 Flexible Wege in der Karriere
____ 119 Hocharbeiten im eigenen Level
____ 174 Ausbildung
____ 184 Management-Karriere
_____ 267 Wechsel in Fachkarriere
_____ 185 Fachliche Kompetenzaufrechterhaltung
_____ 183 Ergebnisbetrachtung durch Debugging
_____ 17 Wechsel in Management-Karriere
____ 69 Fachkarriere
_____ 269 Ausdifferenzierte Stufen in Fachkarriere
_____ 81 Zuarbeiter zu Gesicht-nach-Draussen
____ 177 Beratung-Produktentwicklung-Projektmanagement
___ 118 Einflussfaktoren
____ 272 Doktorgrad
____ 265 Eigeninteresse folgen
____ 251 Soziale Kompetenzen
_____ 252 Überzeugungskompetenz gegenüber Maintainer
____ 200 Konferenzvorträge
____ 179 Reine Open-Source-Erfahrung
____ 153 Neugierde
____ 154 unternehmerisches Denken
____ 149 Technische Kompetenzen
____ 148 Sichtbarkeit nach außen
____ 117 Anforderungskataloge
__ 12 Motivation
____ 278 Internationale Unterschiede
____ 203 Spass an Internationalität
____ 26 Chance zu Open-Source-Arbeit
____ 202 Spass an Open-Source-Arbeit
____ 100 Flexible Arbeit
____ 14 Konstante Teams
____ 13 Interesse an der Arbeit
__ 9 Produktinnovation
__ 1 Einstieg

__ 10 Projektzuweisung von Mitarbeitern
____ 19 Nach benötigten Kompetenzen
____ 11 Mehrfachzuweisung auf Projekte
__ 4 Entwicklungsprozess
____ 232 Portfolio-Planung des OS-Engagements
____ 230 Erfolgskriterien OS-Engagement
_____ 231 Timing des Engagements
____ 197 Open-Source-Projektorganisation
_____ 199 Teamorientierte Projektorganisation
_____ 198 Hierarchische Projektorganisation
____ 22 Arbeitsmerkmale
_____ 225 Hohe Mitspracherechte der Kunden/Aktive Mitsprache der Kunden
_____ 207 Hoher Kommunikationsbedarf
_____ 181 Sozial-Projektkoordination
_____ 180 Kombination Management und Produktentwicklung
_____ 159 Selbst gewählte hohe Arbeitsbelastung
_____ 158 Mitarbeiter repräsentieren die Firma
_____ 104 Familiengefühl
_____ 97 Flexibilität
_____ 42 Verteilte Teams
_____ 137 Wandel in der internen Arbeitsorganisation
_____ 27 Selbstorganisation
_____ 28 Hackweek
_____ 8 Teamarbeit
_____ 6 Home Office
_____ 5 Internationalität
_____ 210 Probleme Feedback zu geben/anzunehmen
_____ 139 Probleme Vertrauen aufzubauen
_____ 21 Open-Source-Arbeit
_____ 260 Sexismus
_____ 259 Unbeabsichtigte Diskriminierung durch Kommunikationsstile
_____ 250 Verbessertes Projektmanagement durch persönliche Treffen
_____ 211 Open-Source-Demographics
_____ 214 Aktueller Stand
_____ 213 Hoher männlicher Anteil
_____ 212 Hoher westlicher Anteil
_____ 215 Wandel
_____ 264 Gesteigertes Problembewusstsein für Diskriminierung
_____ 261 Verringerte technische Zugangsschranken
_____ 216 Steigender Anteil Frauen
_____ 206 Steuermechanismen
_____ 262 Soziales Führen von Projektmitgliedern
_____ 205 Projektaufspaltung
_____ 204 Einflussgewinnung
_____ 196 Community-Management
_____ 195 Patch-Einreichung
_____ 186 Arbeitsauswahl nach eigener Motivation/Lust
_____ 157 Probleme in der OS-Arbeit
_____ 163 Konfliktumgang im OS-Projekt
_____ 164 Unterstützung in Problemsituationen
__ 3 Produkte
__ 2 Organisationsstruktur
____ 167 Einbindung der Mitarbeiter in Organisationsentwicklung
_____ 168 Feedbackschleife mit den Mitarbeitern
____ 98 Belegschaftsalter
____ 95 Anteil Open-Source-Arbeit

____ 94 Offenheit
____ 7 Teamarbeit

4.2 Codes with Memos

Titel	**Memotext**
Überprüfung Verhalten in Mailinglisten	domain=class, model=abstract, paradigm=core,
Nach benötigten Kompetenzen	domain=class, model=instance, paradigm=causal,
Mehrfachzuweisung auf Projekte	domain=class, model=instance, paradigm=actstr,
Projektzuweisung von Mitarbeitern	domain=class, model=abstract, paradigm=core,
Produktinnovation	domain=class, model=abstract, paradigm=core,
Produkte	domain=class, model=abstract, paradigm=core,
Web Development	domain=class, model=setitem, paradigm=context,
Demonstrierte technische Kompetenz	domain=attribute, model=instance, paradigm=context,
Als Contributor	domain=attribute, model=instance, paradigm=context,
Open-Source-Erfahrung	domain=class, model=setitem, paradigm=context,
Hart-im-Nehmen-Sein	domain=class, model=setitem, paradigm=context,
Zielgruppenorientierte Kommunikationskompetenz	domain=attribute, model=setitem, paradigm=context,
Feedback konstruktiv formulieren	domain=attribute, model=set, paradigm=context,
Englische Sprachkompetenzen	domain=attribute, model=set, paradigm=context,
Kommunikationskompetenzen	domain=attribute, model=set, paradigm=context,
Erhöhter Wirkungskreis	domain=attribute, model=setitem, paradigm=context,
Einfühlungsvermögen	domain=attribute, model=setitem, paradigm=context,
Gesteigerte Sozialkompetenzen	domain=class, model=setitem, paradigm=context,
Fortführung technischer Kompetenz	domain=class, model=setitem, paradigm=context,
Bereitschaft zu Sichtbarkeit	domain=class, model=setitem, paradigm=context,
Zukünftige Anforderungen	domain=class, model=set, paradigm=context,
Kommunikationsbarrieren abbauen durch persönliche Treffen	domain=class, model=setitem, paradigm=actstr,

China holt auf	domain=class, model=setitem, paradigm=structural,
Trend in der Personalentwicklung	domain=class, model=set, paradigm=context,
Zukünftige Anforderungen an Manager	domain=class, model=instance, paradigm=context,
Schwierigkeiten der Personalentwicklung	domain=class, model=instance, paradigm=structural,
Probleme mit Mitarbeitern	domain=class, model=instance, paradigm=structural,
Positionierung in wichtigem Open-Source-Projekt	domain=class, model=setitem, paradigm=causal,
Gesicht nach Draussen	domain=class, model=setitem, paradigm=causal,
Motivation zur Personalentwicklung	domain=class, model=set, paradigm=causal,
Sprachkurse	domain=class, model=setitem, paradigm=actstr,
Selbststudium	domain=class, model=setitem, paradigm=actstr,
Schulung	domain=class, model=setitem, paradigm=actstr,
Regelmässiges Feedback	domain=class, model=setitem, paradigm=actstr,
Probleme explizit machen als Projektleader	domain=class, model=setitem, paradigm=actstr,
Pair Programming	domain=class, model=setitem, paradigm=actstr,
Organisation von SUSE Konferenz	domain=class, model=setitem, paradigm=actstr,
Möglichkeit zum Ausprobieren eigener Projekte	domain=class, model=setitem, paradigm=actstr,
Mitarbeitergespräch	domain=class, model=setitem, paradigm=actstr,
Mentoring	domain=class, model=setitem, paradigm=actstr,
Maßnahmen gegen Burnout	domain=class, model=setitem, paradigm=actstr,
Kein OS-spezifisches Programm	domain=class, model=setitem, paradigm=actstr,
Interkulturelle Trainings	domain=class, model=setitem, paradigm=actstr,
Individuelle Maßnahmen	domain=class, model=setitem, paradigm=actstr,
Gespräche bei gemeldeten Problemen	domain=class, model=setitem, paradigm=actstr,
Entsenden auf Konferenzen	domain=class, model=setitem, paradigm=actstr,
Beobachtung der Arbeitsleistung	domain=class, model=setitem, paradigm=actstr,
Austausch mit anderen Kollegen	domain=class, model=setitem, paradigm=actstr,
Anreize zur Mitarbeitermotivation	domain=class, model=setitem, paradigm=actstr,
Maßnahmen der Personalentwicklung	domain=class, model=set, paradigm=actstr,
Entwicklung von Open-Source-Status	domain=class, model=instance, paradigm=consequence,
Technische Kompetenzen	domain=class, model=setitem, paradigm=context,
Soziale Kompetenzen	domain=class, model=setitem, paradigm=context,

Bereiche der Personalentwicklung	domain=class, model=set, paradigm=context,
Anforderungen an die Personalverwaltung	domain=class, model=instance, paradigm=structural,
Personalentwicklung	domain=class, model=abstract, paradigm=core,
Passung ins Team	domain=class, model=abstract, paradigm=core,
OSS	domain=class, model=abstract, paradigm=core,
Teamarbeit	domain=class, model=instance, paradigm=actstr,
Offenheit	domain=attribute, model=instance, paradigm=structural,
Feedbackschleife mit den Mitarbeitern	domain=class, model=instance, paradigm=consequence,
Einbindung der Mitarbeiter in Organisationsentwicklung	domain=class, model=instance, paradigm=actstr,
Belegschaftsalter	domain=attribute, model=instance, paradigm=structural,
Anteil Open-Source-Arbeit	domain=attribute, model=instance, paradigm=context,
Organisationsstruktur	domain=class, model=abstract, paradigm=core,
Nach Vorstellung der Kollegen/des Teams	domain=class, model=abstract, paradigm=core,
Spass an Open-Source-Arbeit	domain=class, model=instance, paradigm=causal,
Spass an Internationalität	domain=class, model=instance, paradigm=causal,
Konstante Teams	domain=class, model=instance, paradigm=structural,
Internationale Unterschiede	domain=class, model=instance, paradigm=context,
Interesse an der Arbeit	domain=class, model=instance, paradigm=causal,
Flexible Arbeit	domain=class, model=instance, paradigm=structural,
Chance zu Open-Source-Arbeit	domain=class, model=instance, paradigm=consequence,
Sendungsbewusstsein	domain=class, model=setitem, paradigm=actstr,
Idee der demokratischen Software	domain=class, model=setitem, paradigm=actstr,
politische Motivationen	domain=class, model=set, paradigm=actstr,
Motivation zu Open Source	domain=class, model=abstract, paradigm=core,
Motivation	domain=class, model=abstract, paradigm=core,
Verhalten in Loyalitätskonflikten	domain=class, model=instance, paradigm=actstr,
Extrovertierte Fachexperten	domain=attribute, model=instance, paradigm=structural,
Angst vor Publizität	domain=attribute, model=instance, paradigm=causal,
Umgang mit Publizititität	domain=attribute, model=instance, paradigm=actstr,
Nach kultureller Diversität	domain=attribute, model=setitem, paradigm=context,

Unterschiedliche Charaktere	domain=attribute, model=instance, paradigm=causal,
Unmotivierte Entwickler leisten keine gute Arbeit	domain=class, model=instance, paradigm=consequence,
Mitarbeiterloyalität zum Unternehmen	domain=attribute, model=instance, paradigm=structural,
Langsames Warmwerden mit Menschen	domain=class, model=instance, paradigm=structural,
Intrinsische Motivation für OS-Arbeit	domain=attribute, model=instance, paradigm=structural,
Geringe Fluktuation	domain=class, model=instance, paradigm=structural,
Flexibilitätswunsch	domain=attribute, model=instance, paradigm=structural,
Angst vor Inkompetenz bei Minimierung der Entwicklertätigkeit	domain=attribute, model=instance, paradigm=actstr,
Mitarbeitermerkmale	domain=class, model=abstract, paradigm=core, Hier müssen wir noch in einem zweiten Durchgang slektiren, welche Merkmale sich auf Mitarbeiter allgemein, Entwickler und Open-Source-Entwickler beziehen bzw. wo Überschneidungen vorliegen
Interkulturelle Kompetenz	domain=class, model=instance, paradigm=consequence,
Kompetenzentwicklung durch Open Source Tätigkeit	domain=class, model=abstract, paradigm=core,
Portfolio-Planung des OS-Engagements	domain=class, model=instance, paradigm=actstr,
Teamorientierte Projektorganisation	domain=attribute, model=instance, paradigm=structural,
Hierarchische Projektorganisation	domain=attribute, model=instance, paradigm=structural,
Open-Source-Projektorganisation	domain=class, model=instance, paradigm=actstr,
Timing des Engagements	domain=class, model=setitem, paradigm=actstr,
Erfolgskriterien OS-Engagement	domain=class, model=set, paradigm=structural,
Wandel in der internen Arbeitsorganisation	domain=class, model=instance, paradigm=structural,
Verteilte Teams	domain=attribute, model=setitem, paradigm=structural,
Teamarbeit	domain=attribute, model=setitem, paradigm=structural,
Sozial-Projektkoordination	domain=attribute, model=setitem, paradigm=structural, Hier ist zu klären, mit welchen Attributen dieser soziale Aspekt versehen wird. Sozialkram?
Hackweek	domain=class, model=instance, paradigm=actstr,
Selbstorganisation	domain=attribute, model=setitem, paradigm=structural,
Selbst gewählte hohe Arbeitsbelastung	domain=attribute, model=setitem, paradigm=structural,

Verbessertes Projektmanagement durch persönliche Treffen	domain=class, model=instance, paradigm=actstr,
Unbeabsichtigte Diskriminierung durch Kommunikationsstile	domain=class, model=instance, paradigm=consequence,
Soziales Führen von Projektmitgliedern	domain=class, model=setitem, paradigm=actstr,
Projektaufspaltung	domain=class, model=setitem, paradigm=actstr,
Einflussgewinnung	domain=class, model=setitem, paradigm=actstr,
Steuermechanismen	domain=class, model=set, paradigm=actstr,
Sexismus	domain=attribute, model=instance, paradigm=structural,
Unterstützung in Problemsituationen	domain=class, model=instance, paradigm=consequence,
Konfliktumgang im OS-Projekt	domain=class, model=instance, paradigm=actstr,
Probleme in der OS-Arbeit	domain=class, model=instance, paradigm=structural,
Patch-Einreichung	domain=class, model=instance, paradigm=actstr,
Verringerte technische Zugangsschranken	domain=attribute, model=instance, paradigm=structural,
Steigender Anteil Frauen	domain=attribute, model=instance, paradigm=structural,
Gesteigertes Problembewusstsein für Diskriminierung	domain=attribute, model=instance, paradigm=structural,
Wandel	domain=attribute, model=setitem, paradigm=context,
Hoher westlicher Anteil	domain=attribute, model=instance, paradigm=structural,
Hoher männlicher Anteil	domain=attribute, model=instance, paradigm=structural,
Aktueller Stand	domain=attribute, model=setitem, paradigm=context,
Open-Source-Demographics	domain=attribute, model=set, paradigm=structural,
Community-Management	domain=class, model=instance, paradigm=actstr,
Arbeitsauswahl nach eigener Motivation/Lust	domain=attribute, model=setitem, paradigm=structural,
Open-Source-Arbeit	domain=class, model=instance, paradigm=context,
Mitarbeiter repräsentieren die Firma	domain=attribute, model=setitem, paradigm=structural,
Kombination Management und Produktentwicklung	domain=attribute, model=setitem, paradigm=structural,
Probleme Vertrauen aufzubauen	domain=class, model=instance, paradigm=structural,
Probleme Feedback zu geben/anzunehmen	domain=class, model=instance, paradigm=structural,
Internationalität	domain=attribute, model=setitem, paradigm=structural,
Home Office	domain=attribute, model=setitem, paradigm=structural,
Hoher Kommunikationsbedarf	domain=attribute, model=setitem, paradigm=structural,

Hohe Mitspracherechte der Kunden/Aktive Mitsprache der Kunden	domain=attribute, model=setitem, paradigm=structural,
Flexibilität	domain=attribute, model=setitem, paradigm=structural,
Familiengefühl	domain=attribute, model=setitem, paradigm=structural,
Arbeitsmerkmale	domain=attribute, model=set, paradigm=structural,
Entwicklungsprozess	domain=class, model=abstract, paradigm=core, Hypothese: Unterteilen in (a) Open-source-Prozess (b) Unternehmensinterner Entwicklungsprozess (c) Kriterien beiden gemein
Verharren in der Fachkarriere	domain=class, model=instance, paradigm=actstr,
Nominierungsbasierte Positionsvergabe	domain=class, model=setitem, paradigm=context,
Gesicht nach Draussen	domain=class, model=instance, paradigm=consequence,
Neue Rollen durch Open Source	domain=class, model=instance, paradigm=structural,
Motivation sich weiterzuentwickeln	domain=class, model=setitem, paradigm=context,
Wechsel in Management-Karriere	domain=class, model=instance, paradigm=actstr,
Wechsel in Fachkarriere	domain=class, model=instance, paradigm=actstr,
Fachliche Kompetenzaufrechterhaltung	domain=class, model=instance, paradigm=actstr,
Ergebnisbetrachtung durch Debugging	domain=class, model=instance, paradigm=actstr,
Management-Karriere	domain=class, model=setitem, paradigm=context,
Interner Stellenwechsel	domain=class, model=setitem, paradigm=context,
Hocharbeiten im eigenen Level	domain=class, model=setitem, paradigm=context,
Wertschätzung der Facharbeit	domain=attribute, model=instance, paradigm=causal,
Gleichberechtigung von Fach- und Managementkarriere	domain=attribute, model=instance, paradigm=structural,
Flexible Wege in der Karriere	domain=class, model=instance, paradigm=structural,
Zuarbeiter zu Gesicht-nach-Draussen	domain=class, model=instance, paradigm=actstr,
Ausdifferenzierte Stufen in Fachkarriere	domain=attribute, model=instance, paradigm=structural,
Fachkarriere	domain=class, model=setitem, paradigm=context,
Beratung-Produktentwicklung-Projektmanagement	domain=class, model=setitem, paradigm=context,
Ausbildung	domain=class, model=setitem, paradigm=context,
Unternehmensinterne Karrierepfade	domain=class, model=set, paradigm=context,
Zeitliche Unterstützung	domain=class, model=setitem, paradigm=actstr,
Interessen-Aufgaben-Matching bei Zuteilung auf OS-Projekte	domain=class, model=setitem, paradigm=actstr,

Finanzielle Unterstützung	domain=class, model=setitem, paradigm=actstr,
Unterstützung Open Source Karriere	domain=class, model=set, paradigm=actstr,
Reputationsaufbau	domain=class, model=instance, paradigm=consequence,
Projekteinstieg	domain=class, model=instance, paradigm=actstr,
Projektmanagement-Komitee-Mitglied	domain=attribute, model=instance, paradigm=actstr,
Maintainer	domain=attribute, model=setitem, paradigm=consequence,
Foundation-Mitglied	domain=attribute, model=setitem, paradigm=consequence,
Committer-Status	domain=attribute, model=setitem, paradigm=consequence,
Open-Source-Karriere-Status	domain=attribute, model=set, paradigm=causal,
OS-Schlüsselposition führt zu höherem Gehalt	domain=class, model=setitem, paradigm=consequence,
Erhöhte Unabhängigkeit der Selbstbestätigung vom Arbeitgeber	domain=class, model=setitem, paradigm=consequence,
Bedeutung von Open-Source-Rockstars	domain=class, model=setitem, paradigm=consequence,
Auswirkungen auf Gehalt	domain=class, model=setitem, paradigm=consequence,
Bedeutung von Open-Source-Status	domain=class, model=set, paradigm=consequence, Status meint Position im Projekt. Wir müssen hier überlegen, ob es eine eigene Kategorie zum Thema "Unternehmensziel" geben sollte
Open-Source-Karriere	domain=class, model=instance, paradigm=context,
unternehmerisches Denken	domain=attribute, model=setitem, paradigm=structural,
Technische Kompetenzen	domain=attribute, model=setitem, paradigm=structural,
Überzeugungskompetenz gegenüber Maintainer	domain=attribute, model=instance, paradigm=structural,
Soziale Kompetenzen	domain=attribute, model=setitem, paradigm=structural,
Sichtbarkeit nach außen	domain=attribute, model=setitem, paradigm=structural,
Reine Open-Source-Erfahrung	domain=attribute, model=setitem, paradigm=structural,
Neugierde	domain=attribute, model=setitem, paradigm=structural,
Konferenzvorträge	domain=class, model=setitem, paradigm=structural,
Eigeninteresse folgen	domain=class, model=setitem, paradigm=actstr,
Doktorgrad	domain=class, model=setitem, paradigm=structural,
Anforderungskataloge	domain=class, model=setitem, paradigm=structural,
Einflussfaktoren	domain=class, model=set, paradigm=structural,
Entwickler-Karriere	domain=class, model=abstract, paradigm=core,
Über Social Networks	domain=class, model=setitem, paradigm=context,
Über OS-Konferenzen	domain=class, model=setitem, paradigm=context,
Über OS-Community	domain=class, model=setitem, paradigm=context,

Über Konferenzen	domain=class, model=setitem, paradigm=context,
Über Jobsuchmaschinen	domain=class, model=setitem, paradigm=context,
Über Headhunter	domain=class, model=setitem, paradigm=context,
Über eigene Website	domain=class, model=setitem, paradigm=context,
Über die Uni	domain=class, model=setitem, paradigm=context,
über Ausbildungsplätze	domain=class, model=setitem, paradigm=context,
Werkstudenten	domain=class, model=setitem, paradigm=context,
Quereinsteiger	domain=class, model=setitem, paradigm=context,
Rekrutierungsprozess	domain=class, model=set, paradigm=actstr,
Unterschiedliche Probleme international	domain=class, model=setitem, paradigm=context,
Schnelligkeit notwendig	domain=class, model=setitem, paradigm=causal,
Persönliche Motivation notwendig	domain=class, model=setitem, paradigm=structural,
Hoher Leistungsdruck durch Vergleichbarkeit	domain=class, model=setitem, paradigm=structural,
Gründe für Mangel an Bewerbern	domain=class, model=set, paradigm=structural,
Mangel an qualifizierten Bewerbern	domain=class, model=setitem, paradigm=structural,
Frauenmangel	domain=class, model=setitem, paradigm=structural,
Eingeschränkte Bewertungfähigkeiten	domain=class, model=setitem, paradigm=structural,
Begrenztes Budget	domain=class, model=setitem, paradigm=structural,
Probleme der Rekrutierung	domain=class, model=set, paradigm=structural,
Hohe Vorqualifikation im OS	domain=attribute, model=instance, paradigm=causal,
Entwicklerrekrutierung	domain=class, model=instance, paradigm=consequence,
wie sie an Aufgaben rangehen	domain=attribute, model=setitem, paradigm=causal,
Vorhandene Projekte	domain=attribute, model=setitem, paradigm=causal,
Umsetzung von Feedback	domain=attribute, model=instance, paradigm=causal,
Programmierfähigkeit	domain=attribute, model=instance, paradigm=causal,
Architekturkompetenz	domain=attribute, model=instance, paradigm=causal,
Technische Kompetenzen	domain=attribute, model=setitem, paradigm=causal,
Umgang mit Problemen	domain=attribute, model=instance, paradigm=causal,
Teamfähigkeit	domain=attribute, model=instance, paradigm=causal,
Kritikfähigkeit	domain=attribute, model=instance, paradigm=causal,
Dolmetscher-Rolle	domain=class, model=instance, paradigm=consequence,
Umgang mit unterschiedlichen Kommunikationsstilen	domain=class, model=instance, paradigm=actstr,
E-Mailverkehr	domain=class, model=instance, paradigm=actstr,
Bugtracker	domain=class, model=instance, paradigm=actstr,
Schriftliche Kommunikationsfähigkeit	domain=attribute, model=instance, paradigm=causal,

Einhaltung sozialer Kommunikationsregeln (Kein Arschloch)	domain=class, model=instance, paradigm=actstr,
Kommunikationsfähigkeit	domain=attribute, model=instance, paradigm=causal,
Hilfsbereitschaft	domain=attribute, model=instance, paradigm=causal,
Bereitschaft sich auf Vorgaben einzulassen	domain=attribute, model=instance, paradigm=causal,
Soziale Kompetenzen	domain=attribute, model=setitem, paradigm=causal,
Persönliche Kontakte im Vorfeld (Vitamin B)	domain=attribute, model=setitem, paradigm=causal,
Menschliche Kompatibilität	domain=attribute, model=instance, paradigm=causal,
Bereitschaft in virtuellen Teams zu arbeiten	domain=attribute, model=instance, paradigm=causal,
Anpassungsfähigkeit	domain=attribute, model=instance, paradigm=causal,
Personale Kompetenzen	domain=attribute, model=setitem, paradigm=causal,
Passung ins Team nach Vorstellung des Managers	domain=attribute, model=setitem, paradigm=causal,
Durch passive Teilnahme am Open Source	domain=class, model=instance, paradigm=actstr,
Involvierung in firmenfremde Projekte	domain=class, model=instance, paradigm=context,
Involvierung in Firmeneigene Projekte	domain=class, model=instance, paradigm=context,
Durch aktive Teilnahme an Open Source	domain=class, model=instance, paradigm=actstr,
Open-Source-Erfahrung	domain=attribute, model=setitem, paradigm=causal,
Offenheit für Neues	domain=attribute, model=setitem, paradigm=causal,
Lernfähigkeit	domain=attribute, model=setitem, paradigm=causal,
Interkulturelle Kompetenzen	domain=attribute, model=setitem, paradigm=causal,
Englische Sprachfähigkeiten	domain=attribute, model=setitem, paradigm=causal,
Commit-Rechte	domain=class, model=instance, paradigm=consequence,
Einfluss in der Community	domain=attribute, model=setitem, paradigm=causal,
Einstellungskriterien	domain=attribute, model=set, paradigm=causal,
Unternehmensmarketing durch Einstellung von Personen	domain=class, model=setitem, paradigm=actstr,
Strategische Einflussnahme durch Einstellung	domain=class, model=setitem, paradigm=actstr,
Einstellungsgründe	domain=class, model=set, paradigm=actstr,
Stellenschaffung für Rockstars	domain=class, model=instance, paradigm=actstr,
Probleme des Assessments	domain=class, model=instance, paradigm=structural,

Persönliches Treffen zur Feststellung der Kompatibilität	domain=class, model=instance, paradigm=actstr,
Kommunikationsfähgikeit	domain=class, model=instance, paradigm=structural,
Öffentliches Portfolio begutachten	domain=class, model=setitem, paradigm=actstr,
Vorbesprechungen zwischen Personen die einstellen	domain=class, model=setitem, paradigm=actstr,
Teambasierte Entscheidungsfindung	domain=class, model=setitem, paradigm=actstr,
Rollenspiele	domain=class, model=setitem, paradigm=actstr,
Referenzen	domain=attribute, model=instance, paradigm=structural,
Fachliche Arbeitsprobe	domain=attribute, model=instance, paradigm=structural,
Fachartikel	domain=attribute, model=instance, paradigm=structural,
Entscheidungsfindung im Assessment	domain=class, model=set, paradigm=actstr,
Dokumentierte Open-Source-Erfahrung	domain=class, model=instance, paradigm=structural,
Aufwand für Assessment	domain=class, model=instance, paradigm=consequence,
Bewerber-Assessment	domain=class, model=instance, paradigm=actstr,
Einstellungsprozess	domain=class, model=abstract, paradigm=core,
Branchenkenntnisse	domain=class, model=abstract, paradigm=core,
Open-Source-Engagement führt zu (gesteigertem) Kundenvertrauen	domain=class, model=instance, paradigm=consequence,
Einfluss auf Produkte nehmen	domain=class, model=instance, paradigm=actstr,
Bedeutung Open-Source für das Unternehmen	domain=class, model=abstract, paradigm=core,

Table 9: Codes with Memos

4.3 Example Output Paradigm Analysis

```
The abstract concept: [OSS] had no instances.
The abstract concept: [Überprüfung Verhalten in Mailinglisten]
had no instances.
The abstract concept: [Nach Vorstellung der Kollegen/des Teams]
had no instances.
The abstract concept: [Passung ins Team] had no instances.
The abstract concept: [Branchenkenntnisse] had no instances.
The abstract concept: [Produktinnovation] had no instances.
The abstract concept: [Produkte] had no instances.

The model had (7) undefined abstract classes.

================================================
```

The paradigmatic analysis of the concept [OSS, id:279] found [9] concepts declared:
causal[1], structural[1], consequence[3], action/strategy[4], context[0].
deep: 5 width: 3
Of [5] possible dimensions [4] were considered, resulting in [80%] dimensional completeness.
If dimensions have been examined, than on average [1.8] instances were defined for this phenomenon/core category.
To compare the number of dimensional instances, amounts have been normalized and the strength could be measured on a scale [0 - 1] .
causal [0.0], structural [0.0], consequence [0.67], action/strategy [1.0], context [0.0],
The standard deviation is: 1.0507935617461448
==========

==========
The paradigmatic analysis of the concept [Mitarbeitermerkmale, id:18] found [14] concepts declared:
causal[2], structural[7], consequence[1], action/strategy[3], context[1].
deep: 3 width: 10
Of [5] possible dimensions [5] were considered, resulting in [100%] dimensional completeness.
If dimensions have been examined, than on average [2.8] instances were defined for this phenomenon/core category.
To compare the number of dimensional instances, amounts have been normalized and the strength could be measured on a scale [0 - 1] .
causal [0.17], structural [1.0], consequence [0.0], action/strategy [0.33], context [0.0],
The standard deviation is: 0.9730443842776415
==========

==========
The paradigmatic analysis of the concept [Personalentwicklung, id:62] found [49] concepts declared:
causal[3], structural[6], consequence[2], action/strategy[21], context[17].
deep: 10 width: 19
Of [5] possible dimensions [5] were considered, resulting in [100%] dimensional completeness.
If dimensions have been examined, than on average [9.8] instances were defined for this phenomenon/core category.
To compare the number of dimensional instances, amounts have been normalized and the strength could be measured on a scale [0 - 1] .
causal [0.05], structural [0.21], consequence [0.0], action/strategy [1.0], context [0.79],
The standard deviation is: 3.792356204027733
==========

==========
The paradigmatic analysis of the concept [Einstellungsprozess, id:23] found [81] concepts declared:
causal[29], structural[16], consequence[4], action/strategy[18], context[14].
deep: 20 width: 13
Of [5] possible dimensions [5] were considered, resulting in [100%] dimensional completeness.
If dimensions have been examined, than on average [16.2] instances were defined for this phenomenon/core category.
To compare the number of dimensional instances, amounts have been normalized and the strength could be measured on a scale [0 - 1] .

causal [1.0], structural [0.48], consequence [0.0], action/strategy [0.56], context [0.4],
The standard deviation is: 1.261961930383953
==========

==========
The paradigmatic analysis of the concept [Entwickler-Karriere, id:15] found [50] concepts declared:
causal[2], structural[15], consequence[10], action/strategy[13], context[10].
deep: 12 width: 12
Of [5] possible dimensions [5] were considered, resulting in [100%] dimensional completeness.
If dimensions have been examined, than on average [10.0] instances were defined for this phenomenon/core category.
To compare the number of dimensional instances, amounts have been normalized and the strength could be measured on a scale [0 - 1] .
causal [0.0], structural [1.0], consequence [0.62], action/strategy [0.85], context [0.62],
The standard deviation is: 0.6258754311103978
==========

==========
The paradigmatic analysis of the concept [Motivation, id:12] found [7] concepts declared:
causal[3], structural[2], consequence[1], action/strategy[0], context[1].
deep: 1 width: 7
Of [5] possible dimensions [4] were considered, resulting in [80%] dimensional completeness.
If dimensions have been examined, than on average [1.4] instances were defined for this phenomenon/core category.
To compare the number of dimensional instances, amounts have been normalized and the strength could be measured on a scale [0 - 1] .
causal [1.0], structural [0.5], consequence [0.0], action/strategy [0.0], context [0.0],
The standard deviation is: 0.4824468517397738
==========

==========
The paradigmatic analysis of the concept [Entwicklungsprozess, id:4] found [50] concepts declared:
causal[1], structural[28], consequence[2], action/strategy[16], context[3].
deep: 15 width: 14
Of [5] possible dimensions [5] were considered, resulting in [100%] dimensional completeness.
If dimensions have been examined, than on average [10.0] instances were defined for this phenomenon/core category.
To compare the number of dimensional instances, amounts have been normalized and the strength could be measured on a scale [0 - 1] .
causal [0.0], structural [1.0], consequence [0.04], action/strategy [0.56], context [0.07],
The standard deviation is: 3.6115468653063756
==========

==========
The paradigmatic analysis of the concept [Organisationsstruktur, id:2] found [8] concepts declared:
causal[0], structural[2], consequence[2], action/strategy[3], context[1].

deep: 3 width: 6
Of [5] possible dimensions [4] were considered, resulting in [80%] dimensional completeness.
If dimensions have been examined, than on average [1.6] instances were defined for this phenomenon/core category.
To compare the number of dimensional instances, amounts have been normalized and the strength could be measured on a scale [0 - 1] .
causal [0.0], structural [0.5], consequence [0.5], action/strategy [1.0], context [0.0],
The standard deviation is: 0.5311816639690516
==========

========== Entire Code System ==========

==========
The paradigmatic analysis of complete code system resulted in total declared paradigm concepts: [268]. Of which:
causal[41], structural[77], consequence[25], action/strategy[78], context[47].
Mean for all intances: 53.6
Standard deviation for all instances: 3.738538641560413
==========

========== Code System reduced set-items ==========

==========
The paradigmatic analysis of complete code system resulted in total declared paradigm concepts: [122]. Of which:
causal[17], structural[41], consequence[17], action/strategy[33], context[14].
Mean for all intances: 24.4
Standard deviation for all instances: 5.305043888353444
==========

4.4 Calculated Code System Metrics

		Causal	Struct.	Conseq.	Act/ Str.	Context	Total	Complete-ness	Aver-age
Code System (before)	instances	38	75	23	75	47	258	100%	51.6
	rel. strength	0.29	1	0	1	0.46	standard deviation = 2.87		
Code System (after)	instances	41	77	25	78	47	268	100%	53.6
	rel. strength	0.79	1.00	0.00	1.00	0.90	standard deviation = 3.73		
Code System (excl. set-item) (before)	instances	15	39	15	30	14	113	100%	22.6
	rel. strength	0.04	1	0.04	0.2	0.00	standard deviation = 0.46		
Code System (excl. set-item) (after)	instances	17	41	17	33	14	122	100%	24.4
	rel. strength	0.11	1.00	0.11	0.70	0.00	standard deviation = 5.30		
OSS (before)	instances	0	0	0	0	0	0	0%	0.0
	rel. strength	0.00	0	0	0	0.00	standard deviation = 0.0	deep=0	width= 0
OSS (after)	instances	1	1	3	4	0	9	80%	1.8
	rel. strength	0.00	0.00	0.67	1.00	0.00	standard deviation = 1.05	deep=5	width= 3
	instances	0	0	0	3	0	3	20%	0.6

Motivation zu Open Source (before)	rel. strength	0.00	0	0	1	0.00	standard deviation = 0.63	deep=2	width=2
Motivation zu Open Source (after)	instances	(relocated)							
	rel. strength								
Bedeutung Open-Source für das Unternehmen (before)	instances	0	0	1	1	0	2	40%	0.4
	rel. strength	0.00	0	1	1	0.00	standard deviation = 0.39	deep=1	width=2
Bedeutung Open-Source für das Unternehmen (after)	instances	(relocated)							
	rel. strength								
Kompetenz-entwicklung durch Open Source Tätigkeit (before)	instances	0	0	1	0	0	1	20%	0.2
	rel. strength	0.00	0	1	0	0.00	standard deviation = 0.31	deep=1	width=1
Kompetenzentwicklung durch Open Source Tätigkeit (after)	instances	(relocated)							
	rel. strength								
Passung ins Team (before)	instances	0	0	0	0	0	0	0%	0.0
	rel. strength	0.00	0	0	0	0.00	standard deviation = 0.0	deep=0	width=0
Passung ins Team (after)	instances	(relocated)							
	rel. strength								
Nach Vorstellung der Kollegen/des Teams (before)	instances	0	0	0	0	0	0	0%	0.0
	rel. strength	0.00	0	0	0	0.00	standard deviation = 0.0	deep=0	width=0
Nach Vorstellung der Kollegen/des Teams (after)	instances	(relocated)							
	rel. strength								
Projekt-zuweisung von Mitarbeitern (before)	instances	1	0	0	1	0	2	40%	0.4
	rel. strength	1.00	0	0	1	0.00	standard deviation = 0.38	deep=1	width=2
Projekt-zuweisung von Mitarbeitern (after)	instances	(relocated)							
	rel. strength								
Überprüfung Verhalten in Mailinglisten (before)	instances	0	0	0	0	0	0	0%	0.0
	rel. strength	0.00	0	0	0	0.00	standard deviation = 0.0	deep=0	width=0
Überprüfung Verhalten in Mailinglisten (after)	instances	(relocated)							
	rel. strength								
Mitarbeiter-merkmale (before)	instances	2	6	1	3	1	13	100%	2.6
	rel. strength	0.2	1	0	0.4	0.00	standard deviation = 0.88	deep=3	width=9
Mitarbeiter-merkmale (after)	instances	2	7	1	3	1	14	100.00%	2.0
	rel. strength	0.17	1.00	0.00	0.33	0.00	standard deviation = 0.97	deep=3	width=10
Branchen-kenntisse (before)	instances	0	0	0	0	0	0	0%	0.0
	rel. strength	0.00	0	0	0	0.00	standard deviation = 0.0	deep=0	width=0

Branchen-kennnisse (after)	instances	(relocated)							
	rel. strength								
Organisations-struktur (before)	instances	0	2	1	2	1	6	80%	1.2
	rel. strength	0.00	1	0	1	0.00	standard deviation = 0.37	deep=2	width=5
Organisations-struktur (after)	instances	0	2	2	3	1	8	80%	1.6
	rel. strength	0.00	0.50	0.50	1.00	0.00	standard deviation = 0.53	deep=3	width=6
Produkt-innovation (before)	instances	0	0	0	0	0	0	0%	0.0
	rel. strength	0.00	0	0	0	0.00	standard deviation = 0.0	deep=0	width=0
Produkt-innovation (after)	instances	(relocated)							
	rel. strength								
Produkte (before)	instances	0	0	0	0	0	0	0%	0.0
	rel. strength	0.00	0	0	0	0.00	standard deviation = 0.0	deep=0	width=0
Produkte (after)	instances	(relocated)							
	rel. strength								
Personal-entwicklung (before) = (after)	instances	3	6	2	21	17	49	100%	9.8
	rel. strength	0.05	0.21	0	1	0.79	standard deviation = 3.79	deep=10	width=19
Einstellungs-prozess (before) = (after)	instances	27	16	4	18	14	79	100%	15.8
	rel. strength	1.00	0.52	0	0.61	0.43	standard deviation = 1.10	deep=19	width=13
Entwickler-Karriere (before) = (after)	instances	2	15	10	13	10	50	100%	10.0
	rel. strength	0.00	1	0.62	0.85	0.62	standard deviation = 0.63	deep=12	width=12
Motivation (before)	instances	3	2	1	0	1	7	80%	1.4
	rel. strength	1.00	0.5	0	0	0.00	standard deviation = 0.48	deep=1	width=7
Motivation (after)	instances	(relocated)							
	rel. strength								
Entwicklungs-prozess (before)	instances	0	28	2	13	3	46	80%	9.2
	rel. strength	0.00	1	0	0.42	0.04	standard deviation = 3.19	deep=14	width=14
Entwicklungs-prozess (after)	instances	1	28	2	16	3	50	100%	10.0
	rel. strength	0.00	1.00	0.04	0.56	0.07	standard deviation = 3.61	deep=15	width=14

Table 10: Calculated Code System Metrics

5 References

Avison, D. E., Lau, F., Myers, M. D., & Nielsen, P. A. (1999). Action research. *Communications of the ACM*, *42*(1), 94–97.

Benbasat, I., & Zmud, R. W. (1999). Empirical research in information systems: the practice of relevance. *MIS Quarterly*, 3–16.

Bryman, A., & Burgess, R. G. (1994). *Analyzing qualitative data*. (A. Bryman & R. G. Burgess, Eds.). Abingdon, UK: Taylor & Francis. doi:10.4324/9780203413081

Buchanan, D., & Bryman, A. (2007). Contextualizing methods choice in organizational research. *Organizational Research Methods*, *10*(3), 483–501. doi:10.1177/1094428106295046

Carver, J. (2004). The impact of background and experience on software inspections. *Empirical Software Engineering*, *9*(3), 259–262.

Charmaz, K. (1997). Grounded Theory Method, 397–412.

Corbin, J. M., & Strauss, A. (1990). Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative Sociology*, *13*, 3–21. doi:10.1007/BF00988593

Crabtree, C. A., Seaman, C. B., & Norcio, A. F. (2009). Exploring language in software process elicitation: A grounded theory approach. In *Proceedings of the 2009 3rd international symposium on empirical software engineering and measurement* (pp. 324–335).

Eisenhardt, K. M. (1989). Building Theories from Case Study Research. *The Academy of Management Review*, *14*(4), 532–550. doi:10.2307/258557

Eisenhardt, K. M., & Graebner, M. E. (2014). THEORY BUILDING FROM CASES : OPPORTUNITIES AND CHALLENGES diverse. *Academy of Management Journal*, *50*(1), 25–32.

Fernández, W. D. (2003). *Metateams in Major Information Technology Projects*. School of Information Systems, Queensland University of Technology.

Fernández, W. D., Lehmann, H., & Underwood, A. (2002). RIGOUR AND RELEVANCE IN STUDIES OF IS INNOVATION: A GROUNDED THEORY METHODOLOGY APPROACH Walter. *European Conference on Information Systems*, 110–119.

Flick, U. (2009). *An Introduction to Qualitative Research*.

Glaser, B. G. (1992). *Emergence vs forcing: Basics of grounded theory analysis*. Sociology Press.

Glaser, B. G., & Strauss, A. L. (1967). The discovery of grounded theory. *International Journal of Qualitative Methods*, *5*, 1–10. doi:10.2307/588533

Glaser, B. G., & Strauss, A. L. (1998). Grounded theory. *Strategien Qualitativer Forschung. Bern*, 53–84.

Glaser, B., & Strauss, A. (2005). Grounded Theory Methods and Qualitative Family Research, *67*(November), 837–857.

Gray, P. (2001). Introduction to the special volume on relevance. *Communications of the AIS*, *6*(1), 1–12.

Hoda, R., Noble, J., & Marshall, S. (2011). Developing a grounded theory to explain the practices of self-organizing Agile teams. *Empirical Software Engineering*, *17*(6), 609–639. doi:10.1007/s10664-011-9161-0

Kock, N., Gray, P., Hoving, R., Klein, H., Myers, M. D., & Rockart, J. (2002). IS research relevance revisited: Subtle accomplishment, unfulfilled promise, or serial hypocrisy? *Communications of the Association for Information Systems*, *8*(1), 23.

Lehmann, H. (2001). *A grounded theory of international information systems*. ResearchSpace@ Auckland.

Lincoln, Y. S., & Guba, E. G. (1986). But is it rigorous? Trustworthiness and authenticity in naturalistic evaluation. *New Directions for Program Evaluation*, *1986*(30), 73–84. doi:10.1002/ev.1427

Lincoln, Y. S., & Guba, E. G. (1990). Judging the quality of case study reports. *International Journal of Qualitative Studies in Education*, *3*(1), 53–59. doi:10.1080/0951839900030105

Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis: An expanded sourcebook*. Sage.

Mintzberg, H. (1979). The structuring of organizations: A synthesis of the research. *University of Illinois at Urbana-Champaign's Academy for Entrepreneurial Leadership Historical Research Reference in Entrepreneurship*.

Orlikowski, W. J. (1993). CASE tools as organizational change: Investigating incremental and radical changes in systems development. *MIS Quarterly*, 309–340.

Robey, D., Boudreau, M.-C., & Rose, G. M. (2000). Information technology and organizational learning: a review and assessment of research. *Accounting Management and Information Technologies*, *10*(2), 125–155. doi:10.1016/S0959-8022(99)00017-X

Rodon, J., & Pastor, J. A. (2007). Applying Grounded Theory to Study the Implementation of an Inter-Organizational Information System, *5*(2), 71–82.

Strauss, A. (1995, March 1). Notes on the Nature and Development of General Theories. *Qualitative Inquiry*. doi:10.1177/107780049500100102

Strauss, A., & Corbin, J. (1994). Handbook of qualitative research. *... Qualitative Research*. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:No+Title#0

Strübing, J. (2008). *Grounded Theory* (2nd ed.). Wiesbaden: VS Verlag.

Thomas, D. R. (2006). A General Inductive Approach for Analyzing Qualitative Evaluation Data. *American Journal of Evaluation*, *27*(2), 237–246. doi:10.1177/1098214005283748

Walker, D., & Myrick, F. (2006). Grounded theory: An exploration of process and procedure. *Qualitative Health Research*, *16*(4), 547–59. doi:10.1177/1049732305285972

Walsham, G. (1995). Interpretive case studies in IS research: nature and method. *European Journal of Information Systems*, *4*(2), 74–81.

Whitworth, E., & Biddle, R. (2007). The social nature of agile teams. In *Agile conference (AGILE), 2007* (pp. 26–36).

Yin, R. K. (2011). *Qualitative research from start to finish*. Retrieved from http://books.google.com/books?hl=en&lr=&id=_XP8iOtMKaoC&oi=fnd&pg=PP1&dq=QUALITATIVE+RESEARCH+FROM+START+TO+FINISH&ots=QkqCGQ6uGK&sig=x2tVml9bfjnKkTfgmo-Ie_nrcZw

Yin, R. K. (2014). *Case Study Design and Methods* (p. 265). Sage publications. doi:10.1097/FCH.0b013e31822dda9e