AIS SIGPRAG
Pre-ICIS Workshop 2016
"Practice-based Design
and Innovation of
Digital Artifacts"

Towards a Design Theory for Collaborative Qualitative Data Analysis

Peter Axel Nielsen

Research Centre for Socio-Interactive Design, Department of Computer Science, Aalborg University, Denmark. pan@cs.aau.dk

Abstract

This position paper addresses how to develop a design theory to support the collaborative practice of qualitative data analysis. Qualitative researchers face several challenges in making sense of their empirical data and IS-support for this practice can be found in software applications such as NVivo, Atlas.ti, and DeDoose. While these software tools have utility and are valuable, they are also limiting – and they are particularly limiting researchers in their collaborative efforts with their co-researchers. In this paper, we investigate a design theory to extend it to support collaboration. We use this as a stepping stone to discuss how to use a design theory to problematize existing applications and how to extend a design theory by abduction.

Keywords: Design theory, collaborative tools, collaborative practice, qualitative data analysis, abduction.

1 Introduction

The research position in this paper concerns how to support qualitative researchers. In particular, it seeks to develop a design theory for supporting co-researchers' collaborative performance of analysis of qualitative, empirical data. In short, the research question is:

> What are the desirable and feasible features of applications to support collaborative qualitative data analysis for researchers?

The question puts an immediate attention to which features of such applications would qualitative researchers find desirable. It leaves open what 'desirable' may be taken to be by researchers where some would see many and advanced features as desirable, and some would find a minimalist set of features desirable boiled down to what is essential for the task of analysing qualitative data. It also leaves open what 'feasible' may be taken to be as for some applications can only be acquired at a relatively high cost (purchasing or learning), while other come at a relatively low cost. As any application has a limited feature sets constrained by some overall design idea or theory. What we shall try to find out is what the design theory is for collaborative qualitative data analysis.

We use here the term 'feature' to emphasise that the focus is on *what* the application may be used for rather than how the features are provided exactly. With that fo-

cus, we also are closer to the usefulness and the utility of the applications rather than their usability. These concerns cannot easily be separated, but the emphasis remains with the former.

The term ‘qualitative analysis’ is perhaps too broad for this investigation as we are much closer to what the literature on research methods call ‘coding' of qualitative data. Though coding to many qualitative researchers is closely linked with grounded theory as a form of qualitative data analysis and they prefer ‘indexing' to emphasise a kind of analysis resulting in concepts and categories that organise and abstract the source data. Analysis of the contents of the empirical data should yield some understanding expressed by the researcher usually in concepts and relationships between concepts. For brevity, we shall refer to this process as the coding process.

Support for a complex knowledge process like coding may come in many forms. We shall here focus in particular on supporting the core characteristics of coding. These characteristics are commonly referred to as individualist activity; that is, the single researcher’s coding. It is not trivial to figure out how to support the individualist activity and try to determine which features are useful. It is then significantly more challenging to address collaboration between researchers. Researchers collaborate as co-authors in joint papers and as co-researchers in joint research projects. Collaboration in qualitative data analysis runs the risk of being reduced to comparing their individual analyses and thus eliminating the opportunities in collaborative analysis. Collaborative analysis performed by a (small) group of researchers may well create the advantage to the researchers informing, influencing, and justifying through a dialogue with each other on how they can arrive at a joint analysis. Differences in perceiving the data can then be viewed as an opportunity for learning rather than merely a source of reduced reliability. It is such a collaborative view on the coding process that we will pursue.

2 The Qualitative Coding Process

The practice we refer to in this investigation is what is commonly known as 'qualitative coding of empirical data.' There are several methodologies explaining what qualitative data analysis is and how to do it. Qualitative data analysis comes also in the form of analysis of narrative (Pentland 1999; Floersch et al. 2010), semantic analysis (Rivard & Lapointe 2005), document analysis (Bowen 2009), thematic analysis (Braun & Clarke 2006), grounded theory (Urquhart et al. 2010; Strauss & Corbin 1998), and contents analysis (Hsieh & Shannon 2005). The stringency of the analysis and the coding of the analysis varies. For example, three approaches to contents analysis have been identified: in conventional contents analysis the codes emerge from reading the text; a directed analysis is based on a theory as initial codes; a summative analysis quantifies and compares contents (Hsieh & Shannon 2005).

All these different analyses apply different concepts and utilise different underlying thinking of what to look for in the data and how to perform the analysis. The purpose here is not to explain the various types of qualitative data analysis, but rather to focus on the coding process, but not in such detail that it can substitute the literature on qualitative research, analysis of qualitative data, and the coding process. The following explains the practice of coding at a higher level of abstraction, i.e., as a process. The intention is that this exposition of the coding process is relatively free from specific assumptions made in specific analysis methodologies.

The core of the coding process is to read the data, select which data will be relevant quotes, and decide which codes should be linked to which quotes. The analysis at a more abstract level also involves the reading and comprehending at the level of the codes to see patterns in the data and the codes. This analysis will be expressed by linking codes with other codes. This converging network of codes is the result of the coding process.

This is illustrated in Figure 1 where the 'data source' contains the qualitative data to be coded. Traditionally a data source is a text, but it can be any piece of qualitative data, e.g., image, photo, video, audio. A 'quote' is then a cohesive fragment of the data source that has been selected to be linked to by a code; this is also to some known as a text unit and can in text sources be anything from a word, a sentence, to a whole paragraph. A 'code' has a that the researchers find meaningful. A 'link' is bi-directional linking both from the quote to the code and vice versa. There can also be links between codes to cater for the expression of patterns among the codes.

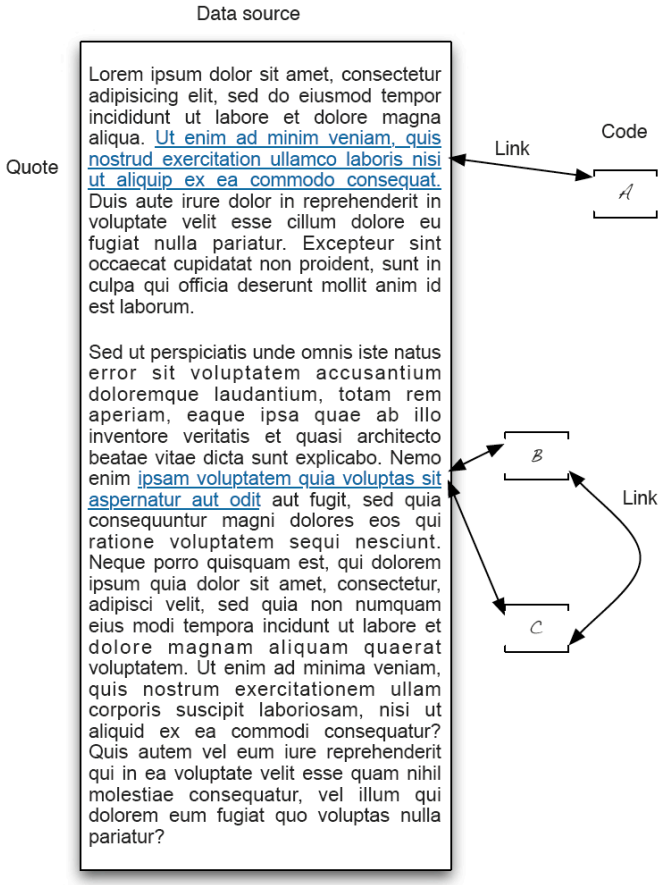


Figure 1: The basic concepts used in the coding process

Miles & Huberman explains that there are two inherent challenges in qualitative data analysis: data overload and data retrieval (Miles & Huberman 1994). There are much too many data to process in a simple manner, and the researcher may lose track of it (overload), and there is such a mass of data that the researcher will have difficulty in finding the most relevant data for the study at hand (retrieval). To support the

researcher in addressing these challenges we need powerful features in the applications.

It is central to the coding process that it is an iterative process. Bryman says: "Do it again" and "Review your codes" (Bryman 2013). To some there is a gradual process where the coding progresses from open coding, axial coding, and selective coding (Strauss & Corbin 1998; Strauss & Corbin 1994) or as phases like initial coding and focused coding (Urquhart et al. 2010; Lazar et al. 2010); but we shall not make assumptions about steps in the coding process. We will, however, suggest that all intermediate quotes, codes and links can be edited at any time.

Figure 2 illustrates that several researchers are collaborating and they can view the data sources, the quotes, the links and the codes made by others and by themselves. They can interact with all the basic elements. The co-researchers may in principle collaborate through the coding, and they may also additionally collaborate by articulating their coordination of their coding.

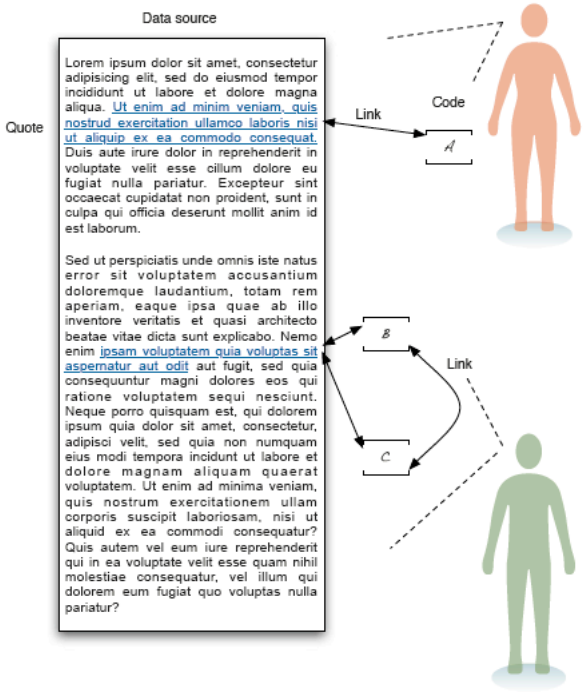


Figure 2: The basic idea of collaborative coding

It is worth noticing that (Miles & Huberman 1994) as well as many others writing about qualitative data analysis describe it as an individualist activity. The exception is the concern for intercoder agreement and intercoder reliability (Cresswell 2009; Bryman 2013) where different coders' analysis can be compared and contrasted. The view we take here on collaboration is, as we shall see, much more than mere intercoder agreement. Several coders can collaborate by viewing each other's coding and discussing their analyses. Paper-based techniques exist for collaborating on qualitative analysis (Harboe et al. 2012; Holtzblatt et al. 2005; Sharp et al. 2011), but there seems to be limited support for this in existing applications.

We take the coding process to contain the characteristics outlined in Table 1. The characteristics are generic and may well take many specific practices depending on the researchers involved and the data sources they are coding. The simplest of the coding process is the coding of data where quotes and codes are linked, and the purpose is to analyse a large data set to arrive at an overview with a set of codes that are meaningful to the co-researchers.

At a higher level of analysis, we need to be able to code the codes, i.e., linking codes with codes, because the complexity of the coding may easily become very high. Levels of abstraction are addressed by linking codes with codes. The functionality we are looking for to support this second order coding concerns displaying codes, links between codes, creating and editing codes and links between codes, and searching for codes and links between codes.

It is fundamental in the analysis that we must be able to move back and forth between the data, the quotes, the codes, and the links. We must be able to appreciate the whole and the parts as well as the specific and the general to arrive at a converging analysis result. To this end, we need to be able to iterate.

Table 1: The contents of the coding process

Feature	**Goal**	**Example Functions**
Coding data	Linking codes with a quote In a large data set	Displaying data, quotes, codes, links Creating and editing codes, links Searching
Coding codes	Linking codes with codes In a complex set of codes	Analysing codes and links Searching, creating, editing, displaying codes and links
Iterating	Revising codes, quotes, and links	Displaying and editing quotes, codes, and links
Collaboration	Authority Coordination	Displaying co-researchers coding Interleaved coding Simultaneous coding by explicit or implicit updating

Collaboration covers all the above aspects of coding, but a few aspects supplements that coding process. This includes a function where the goal is to control authority, i.e., who has access to what; and explicit coordination where co-researchers must explicate the status of what they are doing or explicitly articulate what their task is and how it depends on co-researchers. It also encompasses displaying the status of coding performed by co-researchers and visualising all codes, links and quotes.

These are features of collaboration and dependency between co-researchers that is fundamentally different from the view researchers as individualists that must be separated and independent. We will not pursue co-researchers' independent coding further. We will instead focus on how to support co-researchers' dependency on each

other. This is particularly the case with the two remaining functions. Interleaved coding refers to researchers taking turns in coding. Each consecutive coder can view the results of the previous researcher and add new codes, links, quotes, and memos as well as edit existing ones. This resembles what co-authors do when one co-author sends a document to a second co-author and that co-author works on it for a while after which it gets send back to be worked on by the first co-author. They are working on the same document, but never at the same time. Simultaneous coding or concurrent coding is co-researchers working on the same analysis at the same time without the need to send the analysis back and forth between co-researchers. This resembles what co-authors do with Google Docs or what authors of Wikipedia do when editing articles. The commonality between these two familiar examples is that both applications are offered as a service on the web, and there is access to a shared resource. The difference is that in Google Docs there is (almost) instantaneous updating of own edits and displaying others edits, i.e., without an explicit action by the user to save the editing; and in Wikipedia, the editing of an article is explicitly saved and synchronised by the users after editing.

Table 1 can be taken as a simple design theory. It is far from as elaborate as advocated by influential design theorists (Gregor & Jones 2007; Walls et al. 1992).

3 Feature Comparison of Existing Applications

Table 1 and the features of qualitative coding can be used to illustrate feature comparison using a design theory. Table 2 shows an overall comparison highlighting which functions a specific application possesses to support the design feature. The three applications evaluated and compared are NVivo (www.qsrinternational.com), Atlas.ti (www.atlasti.com), and DeDoose (www.dedoose.com).

Table 2: Comparison of major differences

Feature	NVivo	Atlas.ti	DeDoose
Coding data	Creating, displaying, editing	Creating, displaying, editing	Creating, displaying, editing
Coding codes	Hierarchical linking structure Analytics: predefined and query-based	Network linking structure Analytics: predefined and query-based	Hierarchical linking structure Analytics: predefined
Iterating	Displaying and editing	Displaying and editing	Displaying and editing
Collaboration	Interleaving through file sharing	Interleaving through file sharing	Cloud-based explicit synchronisation

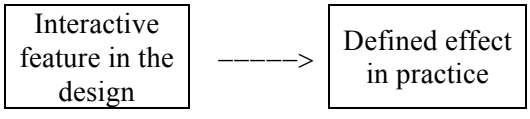
The differences and the similarities between the three design instances are many if we look at the form and function. Table 2, however, can be used to an overall evaluation of how the features are instantiated. The coding of data is very similar across the applications and as the provided functionality basically works like an editor of

codes and links there are genuine support for iteration as well. The differences with 'coding codes' come about because NVivo and DeDoose support hierarchical linking structure while Atlas.ti links in a more general form of networks (and by implication Atlas.ti therefore also supports a hierarchical structure). There are several predefined analytics that can be utilised in all three applications, but both NVivo and Atlas.ti also support query-based analytics requiring the user to form often elaborate logical propositions. The main differences are, however, between NVivo and Atlas.ti on the one hand, that supports collaboration by file sharing and burdening the users with file access, sharing, control, and version conflicts. DeDoose is cloud-based, yet with explicit, non-automatic functionality to save updates to the cloud.

4 Abducting a Better Design Theory

Table 2 and the feature comparison is not without merits, but we need to elaborate the design theory to include why we need these features. A small digression is necessary. A design theory may be taken to consist of (Gregor & Jones 2007): (1) purpose and scope, (2) constructs, (3) principles of form and function, (4) artefact mutability, (5) testable propositions, (6) justificatory knowledge, (7) principles of implementation, (8) expository instantiation. These elements are necessary for a design theory, yet the position taken in this paper is that we need to be able to discuss design theories, elements and their development without these design theories being complete and comprehensive.

In their article Gregor & Jones analyses what they suggest is a design theory of risk management (Iversen et al. 2004) that was developed through action research. Two issues are central. In (Gregor & Jones 2007) it is claimed that the design theory of the risk management approach has a testable proposition (5) that it "is adaptable to other organisational settings, although it is seen as a general approach, rather than a procedure to be followed blindly" (p. 324). But that read more like a principle of implementation (7) and ignores the effect that the approach reportedly has, according to (Iversen et al. 2004). It is further claimed that the justificatory knowledge (6) is that the "approach is derived from other risk management approaches (and other theories)" (p. 324). But that ignores the four iterations of systematic evaluation that justifies the approach, according to (Iversen et al. 2004). This illustrates that it is easy to state design propositions without explicating the effects and that it is easy to state justification without empirical backing. The position taken here is that design propositions must include effects on practice and that they should be the foundation of empirical justification. The suggestion is therefore that a design proposition should have the form:



By 'interactive feature' is meant features as illustrated in Tables 1 and 2 and that these are engaged interactively by users. The arrow should be understood as 'supports' and the 'defined effect in practice' is an effect measure, assessed, and understood in practice, i.e., in the users' situation. The 'supports' part of the propositions is

also not a causal relationship and should more precisely be labelled 'may support' or 'with the intention to support.'

For each of the features in Tables 1 and 2, we may translate it into design proposition that relates design features directly to practice as follows. The first three features are more or less straight forward. These features may also be categorised as routine design (Gregor & Hevner 2013) where the application domain (the methods and practice of qualitative coding) are quite mature and the maturity of current instantiations (cf. Table 2) is also relatively high – yet reproducing a manual process.

1	**Coding data:** Moving between parts (quotes and codes) and wholes (text)	------>	first-order understanding of the text
2	**Coding codes:** Modelling structure (linking) between parts (codes and links)	------>	second-order understanding of the text
3	**Iteration:** Visualisation and editing	------>	more clean coding structure hence an improved first-order and second-order understanding of the text

The effects on the right-hand side have been explicated and elaborated compared to Table 1 and 2. The effects can be assessed through an experiment (lab practice) or action research (experienced practice). We can also use the comparative differences in Table 2 in the abduction of more precise propositions. E.g., the coding-codes proposition (2) could be refined when we look at the differences in the table, and we could now ask if hierarchical or network linking structure leads to a better understanding of the text, or if predefined analytics leads to a better understanding of the text than query-based analytics. No research seems to have evaluated these design propositions (1, 2, 3).

The fourth feature, collaboration, is a bit trickier. This may in simple terms be because the design proposition is closer to an invention as it has been defined by (Gregor & Hevner 2013). The methods and practice of *collaborative* qualitative coding are in their infancy, they are immature, and can perhaps best be understood by learning from web 2.0, tagging, and collective writing and editing. The current instantiations, cf. Table 2, are also lower on maturity in their collaborative parts and their collaborative support than in the support for individualist practices.

Again, abduction based on the cases in Table 2 can help in sharpening design propositions. Both NVivo and Atlas.ti both support collaboration by interleaving. That means the coding structure is saved in a file by one researcher and a co-researcher can then open it, work on it, and save it again. In a shared or cloud storage, this can support a traditional type of collaborative coding where co-researchers can compare codings and measure the intercoder reliability. This means that a proposition (4a) to explain this should focus on the *joining* of independent researchers to increase the consistency of the joint coding in a positivist sense.

4a	**Collaboration:** Interleaved coding between co-researchers	——————>	Co-researchers comparing coding structures, measuring reliability to improve reliability

DeDoose is clearly different from this yet be explained by interleaving, but it offers more. DeDoose offers a cloud-based solution where co-researchers can work simultaneously, however, without the possibility to see co-researchers coding immediately as that requires the explicit synchronisation by the users, e.g., the pushing of the sync-button. Disregarding this limitation will lead to a more ambitious design proposition (4b). As suggested in section 2 this may support more collaborative analysis where co-researchers discuss the coding and influence each other in making meaning of the data, e.g., (Harboe et al. 2012; Holtzblatt et al. 2005; Sharp et al. 2011). The traditional view of the coding process would perhaps warn against this social sense-making. A more collaborative sense-making opens new possibilities for the coding process, it may well have other advantages – as well as disadvantages.

4b	**Collaboration:** Simultaneous coding between co-researchers	——————>	Co-researchers discussing and influencing each other to improve social sense-making

We are well aware that we have only suggested the design propositions, and they are not evaluated yet. We suggest however that the explanation of the practice of qualitative coding is in agreement with existing methodology of qualitative data analysis. We have based the design theory both on this and on the abduction from features of existing applications.

We suggest that existing applications are likely to satisfy Propositions 1 – 4a as we came to these through abduction; but we did not observe the importance of 4a until talking to qualitative researchers. In an informal focus group, they all claimed to miss a better support for collaboration. The focus group found the support for 4a laborious and scored it quite low. The focus group had initially little idea of Proposition 4b as they did not initially see the analogy to Google Docs and few had in fact used Google Docs. It became our prerogative to suggest Proposition 4b.

With these design proposition explicating both interactive features and effects in practice, it becomes possible to evaluate the design theory in practice. This can be done in part with existing applications where it is possible to set up lab experiments as well as field experiments and evaluate the effects on practice. This can be done for propositions 1 – 3. For propositions 4a it can be evaluated through experiments with DeDoose as it contains functionality for interleaving, we just do not know yet whether it has the expected effect on practice. For Proposition 4b a completely new prototype will have to be developed and a lab experiment can be performed to evaluate how the expected effect occurs (in a lab practice). Or the prototype can be evaluated through action research to see how it improves practice as experienced by actors in the situation.

There are several limitations in what we have been trying to argue. To name the most obvious: the abstracted coding process may be flawed in the light of some of the

analysis methodologies, the analysis and comparison of the three applications may be superficial, the abduction of the propositions may be incomplete or inaccurate, the propositions may be blurred and in need of Occam's Razor, and the impact of the propositions without an evaluation in practice is less a contribution.

5 Conclusion

This is a position paper. The intention has been to give background and to create a position from which a design theory can be explored. The design theory is limited to a set of design propositions for a design to support researchers in performing qualitative data analysis. The design propositions have been abducted based on three design instantiations. There is a particular focus on how to advance the design propositions into collaborative, qualitative data analysis.

Another part of the position is to show how propositions can be formulated as 'feature-supports-effect-in-practice'.

References

Bowen, G.A., 2009. Document analysis as a qualitative research method. *Qualitative research journal*, 9(2), pp.27–40.

Braun, V. & Clarke, V., 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2), pp.77–101.

Bryman, A., 2013. *Social research methods*, Oxford: Oxford University Press.

Cresswell, W.J., 2009. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*, Thousand Oaks: Sage Publications.

Floersch, J. et al., 2010. Integrating Thematic, Grounded Theory and Narrative Analysis A Case Study of Adolescent Psychotropic Treatment. *Qualitative Social Work*, 9(3), pp.407–425.

Gregor, S. & Hevner, A.R., 2013. Positioning and Presenting Design Science Research for Maximum Impact. *MIS Quarterly*, 37(2), pp.337–355.

Gregor, S. & Jones, D., 2007. The anatomy of a design theory. *Journal of the Association for Information Systems*, 8(5), pp.312–335.

Harboe, G. et al., 2012. Computer support for collaborative data analysis. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work - CSCW '12*. p. 1179.

Holtzblatt, K., Wendell, J.B. & Wood, S., 2005. *Rapid Contextual Design*, Amsterdam: Morgan Kaufmann.

Hsieh, H.-F. & Shannon, S.E., 2005. Three Approaches to Qualitative Content Analysis. *Qualitative Health Research*, 15(9), pp.1277–1288.

Iversen, J., Mathiassen, L. & Nielsen, P.A., 2004. Managing Risk in Software Process Improvement: An Action Research Approach. *MIS Quarterly*, 28(3), pp.395–411.

Lazar, J., Feng, J.H. & Hochheiser, H., 2010. *Research methods in human-computer interaction*, John Wiley & Sons.

Miles, M.B. & Huberman, A.M., 1994. *Qualitative Data Analysis: An expanded sourcebook* 2nd ed., Thousand Oaks: Sage Publications.

Pentland, B.T., 1999. Building process theory with narrative: From description to explanation. *Academy of Management Review*, 24(4), pp.711–724.

Rivard, S. & Lapointe, L., 2005. A multilevel model of resistance to information technology implementation. *MIS Quarterly*, 29(3), p.2005.

Sharp, H., Rogers, Y. & Preece, J., 2011. *Interaction Design: Beyond Human-Computer Interaction* 3rd ed., Chichester: Wiley.

Sommerville, I., 2015. *Software Engineering* 10th ed., Harlow: Pearson.

Strauss, A. & Corbin, J., 1998. *Basics of qualitative research: Grounded theory procedures and techniques*, Thousand Oaks: Sage Publications.

Strauss, A. & Corbin, J., 1994. Grounded Theory Methodology: An overview. In N. K. Denzin & Y. S. Lincoln, eds. *Handbook of Qualitative Research*. Thousand Oaks: Sage Publications, pp. 273–285.

Urquhart, C., Lehmann, H. & Myers, M.D., 2010. Putting the "theory"back into grounded theory: guidelines for grounded theory studies in information systems. *Information Systems Journal*, 20(4), pp.357–381.

Walls, J., Widmeyer, G. & El-Sawy, O., 1992. Building an Information System Design Theory for Vigilant EIS. *Information Systems Research*, 3(1), pp.36–59.