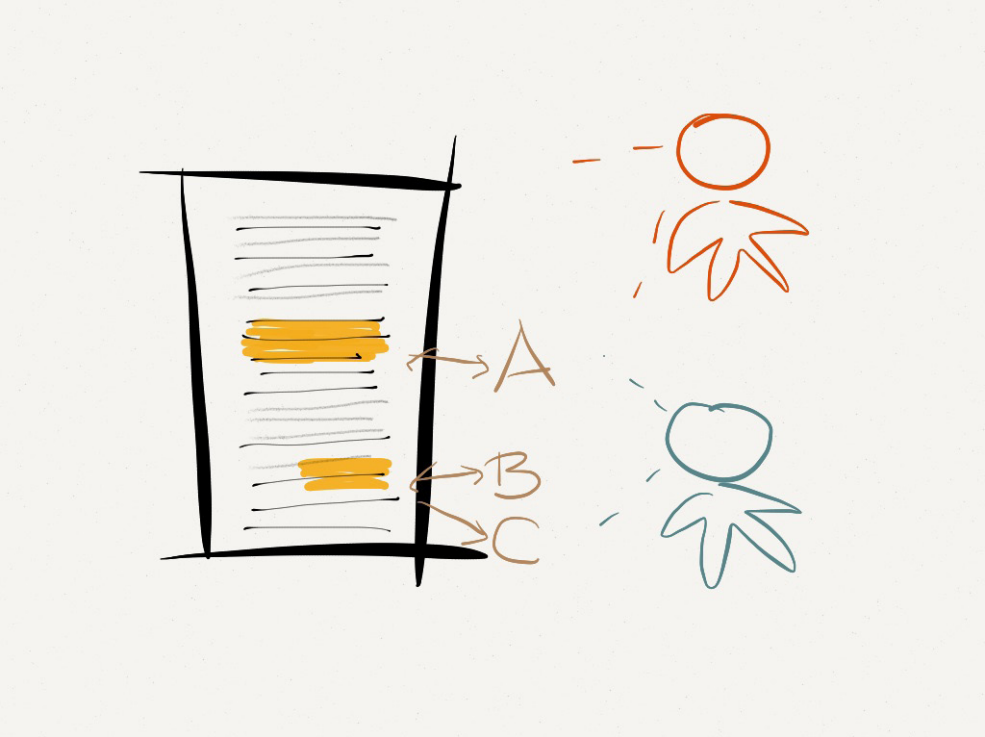
UNIVERSITY OF AGDER – JUNE 2012



COLLABORATIVE CODING OF QUALITATIVE DATA

WHITE PAPER – LA2020

Peter Axel Nielsen

peter.a.nielsen@uia.no

Collaborative Coding of Qualitative Data
White paper from an LA2020 project at University of Agder

Contents

Executive Summary

The purpose of this white paper is to explore how software applications can support the process in empirical research known as qualitative data analysis and in particular how these applications can support the collaborative process of coding qualitative, empirical data. This white paper explain the coding process, the collaborative coding process and then evaluates a prominent set of applications, namely NVivo, Atlas.ti, HyperResearch, Dedoose and to a lesser degree. The specific features of these applications are compared, contrasted and evaluated. It is concluded that the older tools NVivo, Atlas.ti and HyperResearch each have useful features to support individuals coding; but also that they are insufficient to support collaborative coding. Dedoose on the other hand is fully and transparently collaborative while it does not posses the many features to analyse the coding and it particularly does not easily relate codes to each other in a structure (hierarchical or network). The white paper ends with a decision structure to support the selection of an application to match a particular research set-up. The decision structure takes into account the trade-offs between functionality and ease of learning.

Background

An explanation of the background for the investigation is in place before we enter into the details.

LA2020

The report is the result of project under the programme LA20201 at the University of Agder. The project goal was to investigate how applications for qualitative data analysis can support a team of researchers in their research efforts. The LA2020 programme has provided the financial support together with the Department of Information Systems. Initially, the purpose was also to assess the two applications CAT and KiWi in particular. The initial investigation soon led to a broader scope where other applications and more advanced applications than CAT and KiWi were included and substituted with NVivo, Atlas.ti, HyperResearch, Knowdoo and Dedoose.

The Need for Support

The question driving the research has been:

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

This 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. All applications posses limited feature sets; these feature sets are constrained by some overall design or architectural design. It leaves open to try to assess the relationships between features and architectures.

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 is provided exactly. With that focus we also are closer to the usefulness and the utility of the applications rather than their usability. These cannot easily be separated, but the emphasis remains with the former and not the latter.

The term 'qualitative analysis' is perhaps too broad for this investigation as we are much closer—but not limited—to what the literature on research methods call 'coding' of qualitative data. Though coding is to many qualitative researchers 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 process as the coding process. It is the support

for this process by means of applications we are investigating.

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 a challenge to figure out how to support the individualist activity and try to determine which features are useful. It is somewhat more challenging if we consider how to improve the support for the individualist coding.

Some will then add that reliability of the individualist coding can be enhanced when several researchers code the same data and their codings are compared. Inter-coder reliability can then be addressed.

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 how they can arrive at a joint analysis. Differences in perceiving the data can then be view as 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 in this investigation.

Limitation of the Study

We have limited the study to where the qualitative data consists of text. In principle other qualitative data types like audio, video, and images can be treated in a similar manner; but that only in principle. The display features are very different and more difficult to realise. Reading other types will have some similarities with reading text, but also contain some significant differences. Searching in other types of data will be immensely difficult. Browsing and skimming of text and images is definitely possible and not difficult to support in an application. It is also feasible to some extent for video, but it makes little sense for audio.

The Coding Process

This white paper does not explain the coding process in any detail and in particular not in such a detail that it can substitute the literature on qualitative research, analysis of qualitative data, and the coding process.

The core of the coding process is to read the data, select which data will be relevant quotes, deciding which codes should be linked to which quotes. The analysis at a more abstract level involves also 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. In writing about the results in a paper the researcher

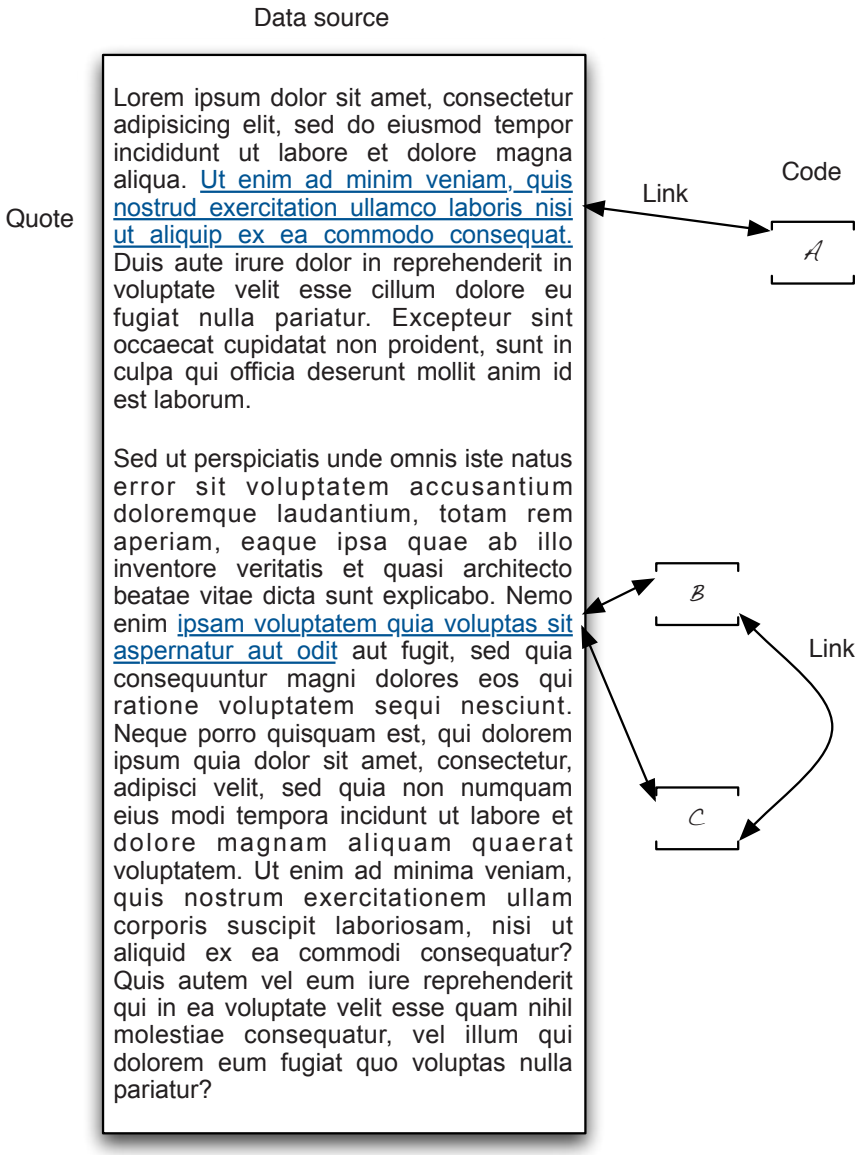


Figure 1: The basic concepts used in the coding process

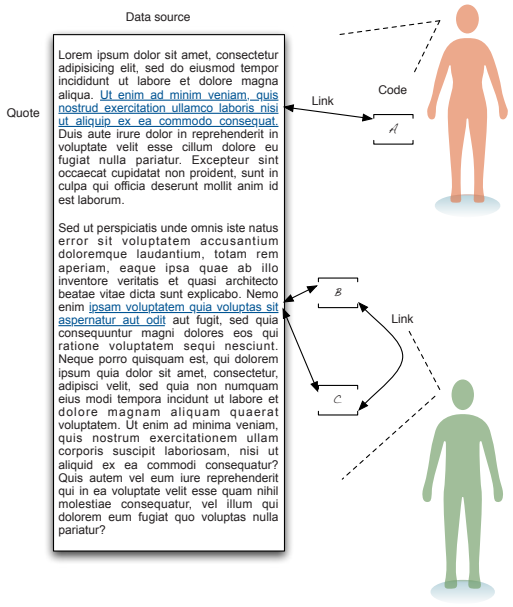


Figure 2: The basic idea of collaborative coding

will need to trace backwards from the network of codes to the codes to the quotes. This is illustrated in figure 1 where the the 'data source' contains the qualitative data to be coded. A 'quote' is then a cohesive fragment of the data that has been selected to be linked to by a code. A 'code' has a label; some label that the researchers find meaningful (and 'A' and 'B' are just abstract placeholders for something specific and more 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.

Miles & Huberman (1994) explains that there are two inherent challenges in qualitative data analysis: data overload and data retrieval. There is much too much data to process in a simple manner and the researcher may loose 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. Miles & Huberman describes in more detail how to code in their handbook (1994, pp. 55-72). What we notice that while they distinguish between different types of codes such as descriptive, interpretive, and pattern codes these will all be denoted simply as codes in an application and the differences in attributed meaning comes from the code's label and description and not its type.

It is central to the coding process that it is an iterative process. Bryman (2008) says: "Do it again" and "Review your codes". To some there is a gradual process where the coding progresses from open coding, axial coding, and selective coding (Strauss & Corbin 1990) or as phases like initial coding and focused coding (Charmaz 2006); 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. In particular 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.

It is also worth noticing that Miles & Huberman as well as most others writing about qualitative data analysis describe it as an individualistic activity. The exception is the concern for intercoder agreement and intercoder reliability (Cresswell 2009; Bryman 2008) where different coders' analysis can be compared and contrasted—or to what degree there is agreement between coders. The view we take on collaboration is much more than mere intercoder agreement. Several coders can collaborate by viewing each others coding and discussing their analyses. A simple yet powerful way to support collaboration could be similar to the features found in applications like Google Docs where several authors write and edit in the same online document simultaneously with display of co-authors' actions directly in the document.

We take the coding process to contain the characteristics outlined in table 1. The characteristics are generic and may well take many specific form depending on the researchers involved and the data sources they

are coding. The most simpel 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. The functionality that we will be looking for in an application concerns displaying the data, the quotes, the codes, and the links to support the reading and comprehending the data and the analysis in its current form. The reading and comprehending is further supported by the possibility to search all parts of the material whether they are data, quotes, codes, or links. It must be possible to create and edit all quotes, codes, and links.

Table 1: The contents of the coding process

Category	Goal	Function
Coding data	Linking codes with a quotes In a large data set	Displaying data
		Selecting and de-selecting quotes
		Creating and editing codes
		Creating and editing links
		Searching data as well as quotes, codes, and links
		Displaying quotes, codes, and links
Coding codes	Linking codes with codes In a complex set of codes	Analysing codes and links
		Creating and editing links between codes
		Searching codes and links
		Displaying links between codes
Iterating	Revising codes, quotes, and links	Editing quotes, codes and links
		Displaying quotes, codes, and links
Documenting	Linking memos to codes, quotes, and links	Creating and editing memos
		Associating memos with quotes, codes, and links
Collaboration	Authority Coordination	Administrating users and access rights
		Interleaved coding
		Displaying co-researchers' coding
		Simultaneous coding

At the 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 is addressed by linking codes with codes. The functionality we are looking for to support this econd order coding concerns displyaing codes, links between codes, creating and editing codes and links between codes, and searching among codes and links between codes.

It is fundamental in 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. The functionality required to do this simply repeats the functionality already listed further up the table, but it emphasise that the functionality must be provided in such a way that previous coding can be changed—repeatedly if necessary.

To further the analysis it is often necessary to document what the meaning associated with a code is. The label of the code hopefully signals that lready, but a brief memo explaining the meaning of a code may be advantageous---in particular when several researchers are collaborating. Hence, it must be possible to create and edit memos and to associate these with codes. In more general terms, it must be possible to associate memos not only to codes but also to quotes and links.

Collaboration covers all the above aspects of coding, but a few aspects supplements that actual coding process. This includes function where the goal is to control authority, i.e., who has access to what; and explicit coordination where co-researchers have to explicate the status of what they are doing or explicitly articulate what their task is and how it depends on co-researchers. The supporting functions encompass supporting (head) researchers in administration their co-researchers as users of the application and which access rights they should possess. It also encompass displaying the status of coding performed by co-researchers and making visible all codes, links and quotes.

These are features of collaboration and dependency between co-researchers that is fundamentally different from the view that co-researchers must be separated and independent. The latter view emphasises inter-coder reliability where the point is that many co-researchers analyse independently and then compare the results afterwards. We will not pursue co-researchers' independent coding further. We will in stead focus deliberately 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 turn 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 awhile after which it it 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 support for 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 by the users after editing.

Assessment of Applications

The assessment of applications to support collaborative coding of qualitative data is presented in an overview of the selection of applications, an assessment of the coding functions offered, a particular focus on the assessment of the collaborative functions, and ending with an overview. The purpose with the assessment is two-fold: to be able to distinguish between the applications at a detailed level, and elicit the degree of support for the coding process, cf. table 1.

Overview of Applications

This assessment encompass several different coding applications, see table 2.

Table 2: Overview of applications; information updated May 2012

Application	Source	Cost
NVivo Windows client	www.qsrinternational.com	1 license USD 670; 10 licenses USD 5360
Atlas.ti Windows client	www.atlasti.com	1 license USD 670; 10 licenses USD 4800 or USD 1400/year
HyperResearch Windows & Mac client	www.researchware.com	1 license USD 495; 10 licenses USD 4050
CAT Web service	cat.ucsur.pitt.edu/	Free
Knowdoo Web service	knowdoo.net	Free
Dedoose Web service	www.dedoose.com/	1 license USD 132/year; 10 licenses USD 1122/year

NVivo and Atlas.ti were included because these applications are already known by many qualitative researchers. Both applications are only available for Windows. HyperResearch is less comprehensive, available for Windows and Mac OS, and initially by many qualitative researchers considered easier to learn. CAT, Knowdoo, and Dedoose are offered as web services for all main browsers including Internet Explorer, Firefox, and Safari. CAT and Knowdoo both have limited functionality, but are included because they are free. The set of applications is a reasonably representative sample of major applications in the domain and of relevant applications.

The applications differ to a large extent as we shall see below. Any direct comparison is likely to disservice some features which may well be prominent in a single application; but in this report we have chosen to compare the applications against the generic coding process outlined in the previous section. The first part of the assessment concerns the features required by individual users; and the second part then

concerns the features required to collaborate between users.

Coding and Collaborative Coding

At the level of support for the coding by the individual researcher, cf. table 1. The bibliography provides reference to demos and tutorials for all the applications.

NVivo

Web➔
Demo➔
Tutorial➔

NVivo is a client application available for Windows. It started as NUD*IST in 1981 and has been updated and improved considerably since.

Data sources can be of many kinds, e.g., text, PDF, image, audio, and video. Selecting a piece of text and marking it with a node (the term for a code in NVivo) corresponds to linking a quote to a code. Displaying quotes, codes, and links is supported in browser windows, see figure 3. Clicking on a code creates in the lower right window a list of the quotes linked to by that code. There is noother or direct way of displaying links between quotes and codes where codes and quotes can be browsed in parallel.

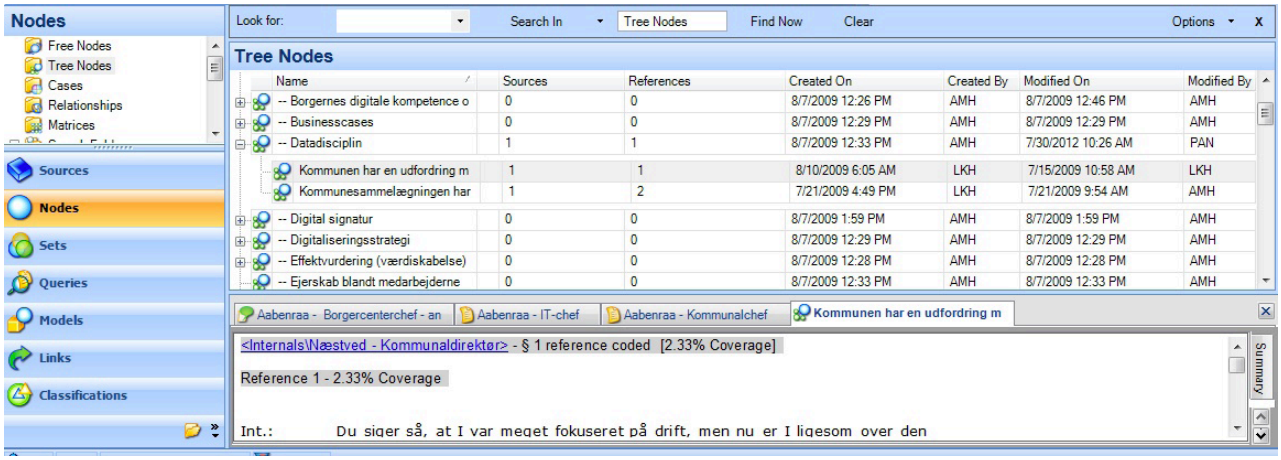


Figure 3a: Screen dump from NVivo

NVivo supports many advanced ways of querying the codes and quotes. This is achieved in two different ways either as boolean query building, see figure 3b, or by calculating matrices of frequencies of codes. It supports also hierarchical linking of codes (termed tree nodes); other ways of linking codes with codes are not supported.

Iteration is supported as all quotes, codes, and links can be edited.

Documentation is supported by allowing memos (termed descriptions) to be entered for all primary aspects, e.g., codes and queries.

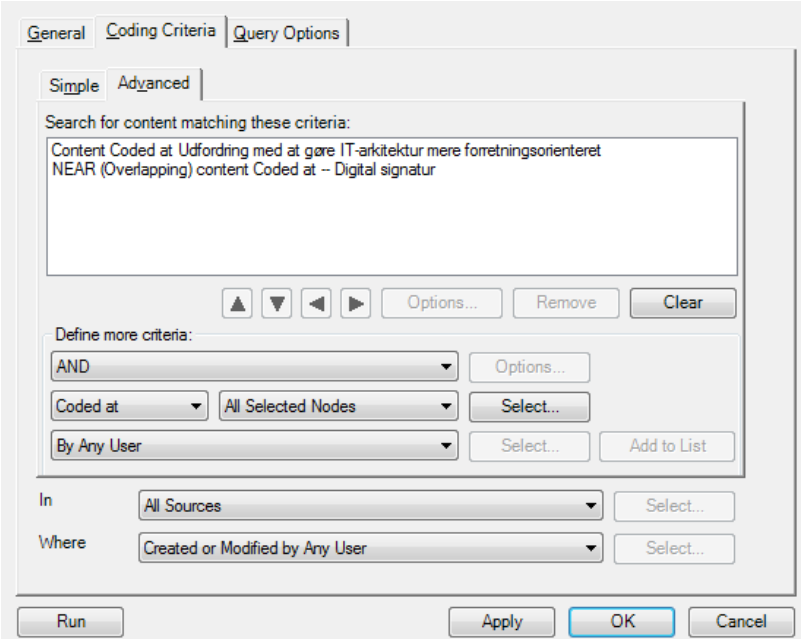


Figure 3b: Screen dump of query in NVivo

Authority is supported in NVivo by administrating users in a simple way. All actions on data sources, quotes, codes, and links are logged for each user, see figure 3a. It is for example logged who created and who modified a code as well as when the actions occurred.

NVivo supports interleaved coding and not simultaneous coding. The interleaved coding means that the coding file has to be transferred to co-reseaerchers or reside on a shared network drive. It also means that they cannot code simultaneously. After each co-researcher has ended coding the file is saved to the hard disk. When it has been saved it can be taken over by another co-researcher. This can with a bit of discipline known from sharing document between co-authors be brought to work. It is, however, increasing the overhead of coordination (i.e., its articulation) because synchronization has to be organised. Co-researchers must agree on when to swap the file from one co-researcher to another, and what the task of a co-researcher in the next period with exclusive access to the file.

After the completion of this comparative assessment NVivo has begun offering a server-based solution. Each user will still need a client license and the cost of the server license has to be added to the cost listed above. Quotes for the server license can be obtained from NVivo.

Web➔
Demo➔
Tutorial➔

Atlas.ti

Atlas.ti is a client application available for Windows. It is gradually been updated and improved since 1993. Apart from the most basic coding functionality, see figure 4a, it also features many advances function for searching and visualizing patterns in the data and among the codes.

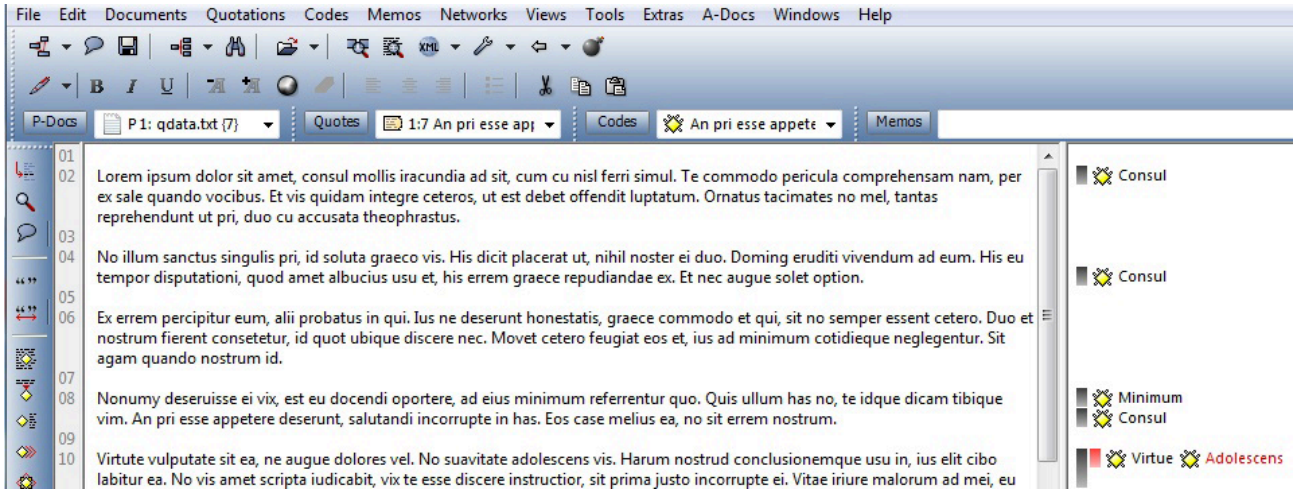


Figure 4a: Screen dump from Atlas.ti

Displaying quotes, codes, and links is supported in the main window, see figure 4a, where switching between data sources (P-Docs), quotes, and codes is easily done at the top. All links are then shown by displaying the codes next to the data source where the quote is; and clicking a code will highlight the quote in the data source. Atlas.ti can work on many different formats of data sources: plain text, formatted text, PDF, image, audio, and video. Audio and video is displayed in a particular browser where markers for starting and ending a quote can entered; but the browser obviously have to show a quote by running it in a player.

Atlas.ti has extensive support for searching the data, quotes, codes, and links through boolean queries. E.g., overlapping quotes and codes can be found with a small query or in a co-occurence table. There is altogether amble support for analysing the coding.

Codes can be linked to codes in a network editor. Links can be labelled with a standardized set of relationships commonly known from conceptual modelling.

Iteration is supported as all quotes, codes, and links can edited at any time.

Documentation is supported to a large extent as memos can be associated with all quotes and codes.

Authority is handled in a comprehensible way with first the administration of users, see figure 4b, and second that these user profiles are logged for every coding action. E.g., it is logged for every code who created and modified the code as well as when these actions were performed, see figure 4c.

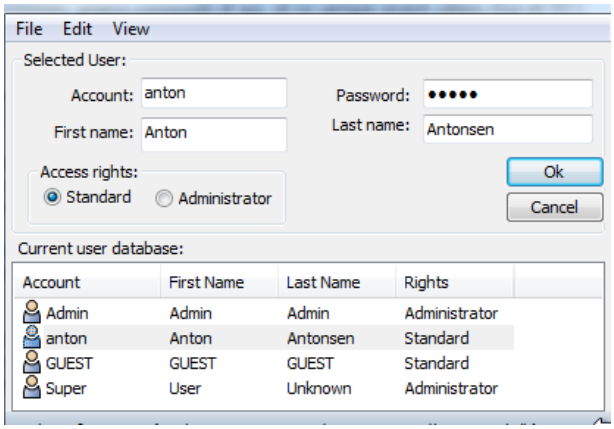


Figure 4b: Administration of users in Atlas.ti

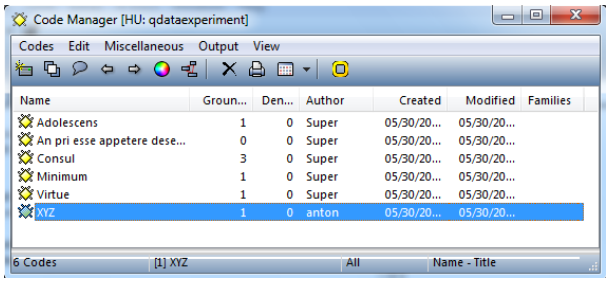


Figure 4c: Logging of users' actions in Atlas.ti

Coordination among a team of researchers is supported by the minutely logging all actions and by interleaved coding. The interleaved coding means that the coding file has to be transferred to co-reseaerchers or reside on a shared network drive. It also means that they cannot code simultaneously. After each co-researcher has ended coding the file is saved to the hard disk. When it has been saved it can be taken over by another co-researcher. This can with a bit of discipline known from sharing document between co-authors be brought to work. It is, however, limiting the possibility to code in smaller sessions because of the overhead of organised synchronization, e.g., when should the file be swapped from one co-researcher to another, and what is the task of a co-researcher in the next period where there is exclusive access to the file?

Web➔
Demo➔
Tutorial➔

HyperResearch

HypereResearch is a client application offered for both Windows and Mac. It is priced below both NVivo and Atlas.ti, but the number of features are also much less. It has been updated continually since 1999.

The functionality is rather simple to learn and to use. This makes it attractive for many researchers as their first coding application. The learning curve will be gentle, and there are several tutorials and demos. It has all the basic functions for coding, see figure 5. It covers the basic concept of data source, code, and link. These are easy to display and to search.

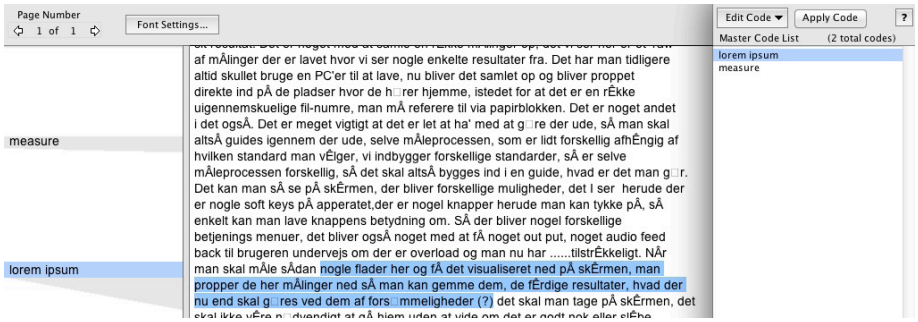


Figure 5: Screen dump from Hyper-Research

HyperResearch supports linking between codes in what is termed a code map. The links between codes are simple.

Iteration are supported as all quotes, codes, and links can be edited. There are additionally a few function for re-coding a large set of codes.

Documentation is supported with memos that can be associated with a link or a code (termed code descriptions). A code map showing links between codes can have several memos; and these can be used for turning simple links into links with more semantics—either specific to the data or general relationships as know from conceptual modelling.

Collaboration is not supported. There is no concept of more than one user in HyperResearch. An analysis is stored in a single files and that file may therefore be moved to a co-researchers hard disk or be shared over a network drive. Co-researchers will not be able to work on it at the same time and there will be no logging or identification of who edited which quotes, codes, and links. HyperResearch will thus induce the simplest of interleaved coding in a research team.

CAT

Web➔
Demo➔

CAT is a free web service. It is offered by the University of Pittsburgh and the data are stored at a reliable and safe server at the university.

A data source can be a set of plain text documents or XML documents. The data sources are split into paragraphs, and all coding is performed at the level of paragraphs. That is, any paragraph is in principle a quote. The researcher steps through all paragraphs one at a time and decides which code(s) should be linked to this quote, see figure 6. This the single defining feature of CAT. All other applications lets the user select the quote allowing for text not contained in a quote and allowing for quotes with a smaller scope than a full paragraph.

Figure 6: Screen dump from CAT

There is no support for linking codes with codes.

Iteration is possible by stepping through each of the paragraphs one at a time—again. However, the crux of CAT is that co-researchers perform their coding in a single iteration. There can afterwards be the task of checking that the right codes were linked to the quotes.

For each paragraph (i.e., quote) it is possible to associate one or more memos. It can be controlled by the user writing a memo who of the other users can see the memo.

Co-researchers can perform their coding simultaneously. For each quote it is possible to toggle whether co-researchers' codes are displayed or not. However, dependency between co-researchgers is de-emphasised, and collaboration in CAT is viewed as a question of inter-coder reliability. The analysis tools provided features statistical measures of reliability agreement (Fleiss' kappa and Krippendorff's alpha). These are used in the validation of the codes linked to the quotes and later in adjudicating the codes.

CAT is also available as open source and that makes it possible for users to set up their own web service. This will be convenient if university policies require that research data are kept behind the university's own fire wall.

CAT was not evaluated in full detail because its functionality is limited to coding whole paragraphs. It has only very weak support for collaboration and has a primary focus on analysing inter-coder reliability. We consider these as very limiting features, and CAT is probably only useful for special coding purposes and not for coding in general. Within these limitations CAT is faster to use that any other application we have assessed.

Knowdoo

Web➔
Demo➔

Knowdoo is rather simple yet with some useful features. It is built as a modification of the blogging software WordPress and many of its features are commonly known in blogging, e.g., entering text on a page (data sources), associating categories and tags to text (linking codes to quotes), publishing texts with categories and tags, and inviting comments at the bottom of each text. These basic blogging features resembles what is needed in coding of qualitative data. However, everything not commonly known from blogging software is absent, see figure 7.

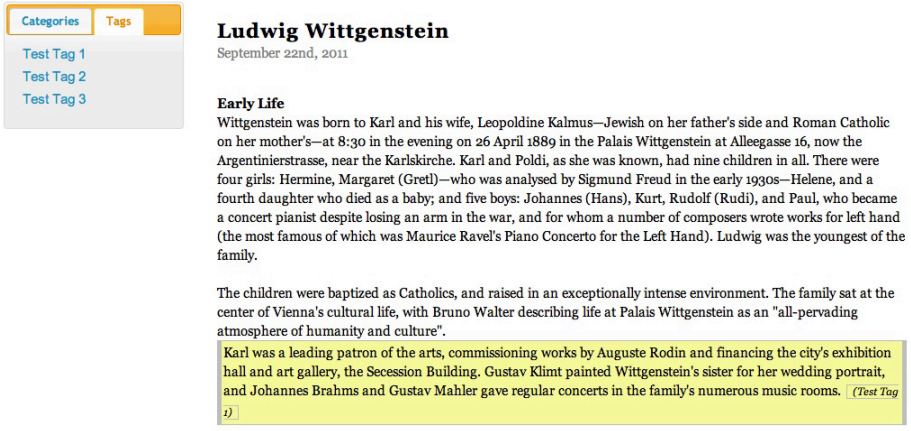


Figure 7: Screen dump from Knowdoo

Knowdoo is offered as a free web service by knowdoo.net. Data sourc-

es can be uploaded and each data source can have several categories that are hierarchically ordered. Tags can be linked to any segment of the data sources. The tags work as code; but there is no support for linking tags with tags.

Iteration is supported in the same way as one may iterate over a text document in an editor. There is no support for documenting the coding in memos.

There can be several users working on the same data source simultaneously. It works without moving files between users' hard disks and without have a shared file on a network drive. The limitation is that two co-researchers might try to update the data source with each their code at the same time. This will usually work fine. One co-researchers may however not necessarily see what another co-researcher just has updated.

There is no information about how reliable and safe the Knowdoo server is; and one should not assume that it is reliable and safe. The web service is provided as-is with no liability accepted. This is high risk for a research team depending on their analyses and investing time and effort in analysing.

Knowdoo is also available as open source and that makes it possible for users to set up their own web service. This will be convenient if university policies require that research data are kept behind the university's own fire wall.

Knowdoo was not evaluated in full detail because it is very visible that it has quite limited functionality. It is however very easy to learn and very easy to use.

Dedoose

Web➔
Demo➔
Tutorial➔

Dedoose is a new application and it is offered as a web service in a Web 2.0 design. As a web-based service it works in all internet browsers irrespective of underlying operating system.

Dedoose has the most basic coding functionality, see figure 8a, and it also features many advances function for searching and visualizing patterns in the data and among the codes.

Editing and displaying quotes, codes, and links is supported in the main window, see figure 8a. A quote (termed an excerpt) is selected in a data source when the quote has been created codes can be associated. Codes are defined before they are used and in vivo codes are not part of the underlying idea. Every data source displays it quotes as coloured highlights and when selecting a quote the codes become visible in a separate window (top-right in figure 8a).

Dedoose can work on many different formats of data sources: plain text, formatted text, PDF, image, audio, and video. Audio and video is displayed in a particular player where quotes can be created and edited.

Dedoose has extensive visual support for searching the data, quotes, codes, and links. There is a convenient set of predefined analyses that can performed. These are again very visual and can for example display tabulated frequencies, overlapping quotes and codes, and co-oc-

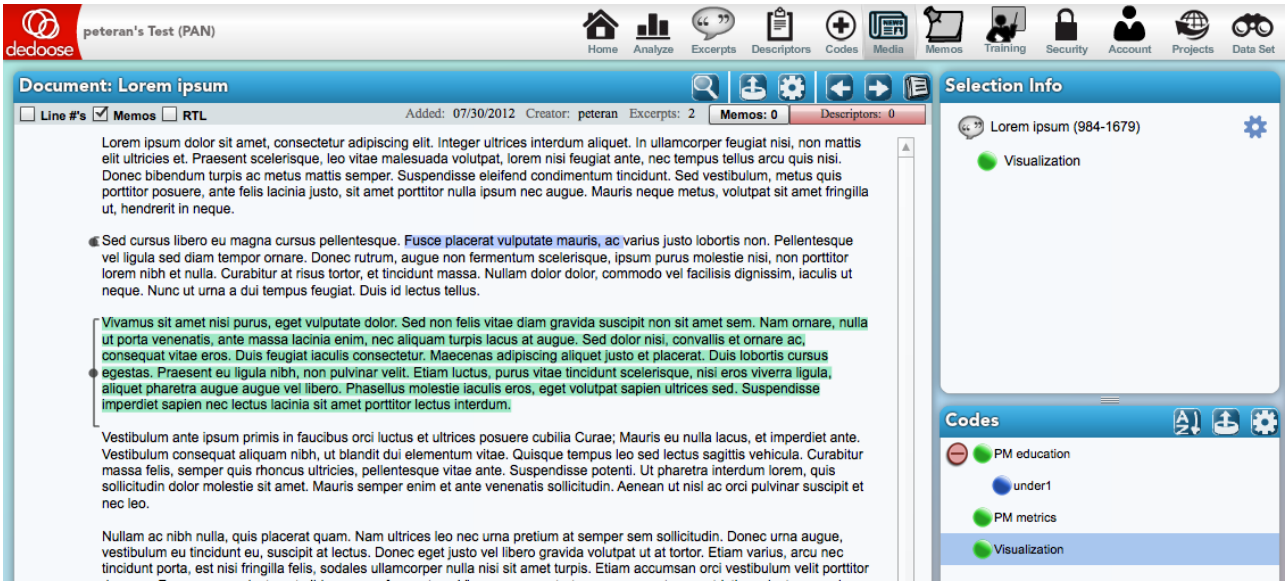


Figure 8a: Screen dump from Dedoose

curences. The options for analysis are fixed, see figure 8b, and there is no opportunity to define boolean queries.

Codes can be linked to codes in a hierarchical structure. Some of the predefined code-links are displayed by the analyses, e.g., co-occurence and tag cloud. There are no other means of relating codes with codes, e.g., it is not possible to express that one code is a specialization of another code.

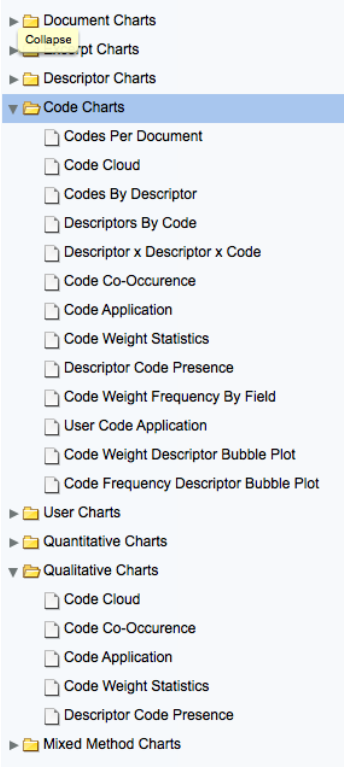


Figure 8b: Extract of analysis functions

Iteration is supported as all quotes, codes, and links can be edited at any time.

Documentation is supported to a large extent as memos can be associated with all quotes, codes, and links.

Authority is handled in a comprehensible way where the administration of users secures authentication of users and controls exactly which access right they have for every project they are involved in. The user profiles are also used for logging for coding action.

Coordination among a team of researchers is supported in four ways: (1) logging all actions; (2) displaying all coding done by co-researchers; (3) allowing interleaved coding where a current coding is passed on to a co-researcher; (4) an internal chat facility used to articulate coordination; and (5) simultaneous coding. The simultaneous coding work in a particular requiring users' actions. The current coding has to be synchronized with the server and other users have to re-fresh their view. This is somewhat limiting as it requires explicit interactions by the co-researchers; but

keeping an open chat will easily allow for immediate synchronization.

Comparison

A part of the conclusion must be that the free services are rarely sufficiently poweful and reliable for serious research. A PhD study should not without depend on the functionality offered by these applications. They appear immediately attractive because they are free. However, that may well carry a larger cost in the long run. They will not be sufficient in the long run, and the cost of shifting to a new application has to be added. The cost to the researcher of not having available full functional support may also be significant. They are also among the applications with amble support for collaboration. Nevertheless, it does add up to the conclusion that these are not worth the time and effort.

The four other applications in this assessment (NVivo, Atlas.ti, HyperResearch, and Dedoose) all have sufficient strengths to support researchers in the coding, in iteration, and in documentation. There is variation, but the differences are merely differences in style and not in functionality and in how well they support the basic coding. They all offer full flexibility in editing every quote, code, and link, but again there is some variation in how they support the researchers in gaining an overview of coding status and thereby allowing moving back and forth between the data sources on the one hand and the quotes, codes, and links on the other hand. Documentation varies only slightly, too. The variation is a matter of style and whether the underlying design idea has been that everything can have an associated memo or memos are free-floating entities that can be associated to everything.

The four applications differ on how they support linking of codes and well they support collaboration. There are four areas in which they differ significantly: analysis functions, linking of codes, authority, and coordination as either interleaved coding or simultaneous coding. Let us llok at these one at a time:

Analysis functions: The applications offer automatic ways of linking codes with each other. This is often found in two different types of analysis functions. Predefined analysis functions are common , e.g., a co-occurence analysis showing which codes are linked with the same quotes. Boolean query functions are found in several applications, e.g., a function in which a boolean expression can be formulated and tested on the code set. NVivo, Atlas.ti, and Dedoose offers several predefined analysis functions. HyperResearch offers none. NVivo and Atlas.ti have elaborate query functionality and HyperResearch has a slightly more limited query functionality. The more complex the query that needs to be formulated the more knowledge and experience the researcher will need to interact with the functions. Dedoose does not offer a query function.

Linking of codes: Codes may also be linked by the researchers by explicitly creating a link between two codes. Linking code comes in three forms: hierarchical, simple network, and complex network. Hierarchical linking of codes to each other means that the codes can be expressed as a tree structure where the root codes are decomposed into more and more detailed codes whenever the tree branches, and

a leaf code means that the code cannot be decomposed further. A tree structure is in principle a whole-part structure and it may be built in a mix of moving from the roots and moving from the leafs. A network means that any code can be linked with any code. In the simple network there is no further semantics associated with a link. In a complex network the links have further semantics. In a complex network it is possible to express that one code is a specialisation of another, that one is an instance of another, that two are associated, that a code contains another, etc. It usually is possible to express everything that can also be expressed in conceptual modelling, e.g., in UML. Atlas.ti offers functionality for linking codes in a complex network. HyperResearch offers functionality for linking codes in a simple network. NVivo and Dedoose offer functionality for linking codes in a hierarchy.

Authority: The control who can view the data sources and the coding and who can perform which actions is crucial. This is important for ethical reasons and unauthorized access must be prevented, but it is also important because co-researchers may well have different roles in the analysis of the qualitative data. Auhority conrol comes in three forms with increasing levels of control: none, user logging, and elaborate control. HyperResearch has no control of users. Atlas.ti and NVivo have functions for creating users and logging which users performed which actions. These logs are immediately visible in most lists of quotes, codes, and links. Dedoose has elaborate functionality for assigning users different roles and thereby control what they can view and what they can edit.

Interleaved coding: Co-researchers can collaborate through interleaved coding, but not at the same time. For a client application it means that the project file either has to be moved back and forth between co-researchers hard disks or that they have to share a network drive or service, e.g., a DropBox. Coordination has to be explicit as they will have to explicitly move the file or transfer the right to write in the file. Simultaneous writing is not possible. NVivo, Atlas.ti and HyperResearch work on such an explicit coordination mechanism. Dedoose is already on a server meaning that interleaved coding is easily supported without the explicit moving of files; but if turn taking is desired it will require explicit coordination.

Simultaneous coding: Co-researchers can collaborate at the same time if they share a resource or service where they all can write at the same time. Collisions—should tehy occur—are handled by the server application. Only Dedoose support simultaneous coding, It supports it in a slightly limited way where the synchronization occurs when it is explicitly activated or the browser is refreshed. This can be coordinated at a micro level through the built-in chat function. None of the assessed applications support implicit coordination of simultaneous coding where explicit synchronization is not necessary.

The major differences are summarised in table 3.

	NVivo	Atlas.ti	HyperResearch	Dedoose
Analysis function	Predefined & Query	Predefined & Query	Query	Predefined
Linking codes	Hierarchy	Complex network	Simple network	Hierarchy
Authority	Simple	Simple	None	Elaborate
Interleaved coding	File sharing	File sharing	File sharing	Server-based
Simultaneous coding	None	None	None	Synchronization
Learning curve	Steep	Steep	Gentle	Medium

Table 3: Major differences compared

It should also be noticed that the evaluation also concluded that HyperResearch has a very gentle learning curve due to its much simpler functionality. Dedoose has a medium learning curve where the coding is easy to learn and the analysis functions will take more effort to learn. Both NVivo and Atlas.ti are comprehensive in their own ways and offers many more functions than most researchers will use; and there learning curves are as a consequence much steeper.

A Road Map

With the above assessment as a background it is now possible to provide a road map for a team of qualitative researchers. It should be noted that a team of researchers can as small as a Ph.D. student and the supervisor and as large as a whole research project. The road map is intended to be useful in either situations.

Decision Tree

Figure 9 contains a decision tree that reflects the same conclusions that were reached summarised in table 3. Start in the upper left corner with the statement "Simplicity is key and less functionality is better" and answer 'yes' or 'no' to the statement based on how you consider your research setting. Answering 'yes' bring you to a choice between CAT and Knowdoo based on the statement "There are many coders and the focus is on paragraphs". If you answered 'no' to the first statement the next main issue is whether it is sufficient for you that coordination has to be through interleaving where you will have to explicitly move project files from one co-researcher to another. Answering 'no' here means that you require coordination through simultaneous coding, and that leads directly to Dedoose. Answering 'yes' can bring you to selecting HyperResearch if you think prefer that the application is a "Simple tool with gentle learning curve". If you do not require such a simple tool and is prepared to invest a bit more you should answer 'no' at this point and go to the statement "Predefined analysis functions will suffice". At this point you are well advised to select Dedoose if it a 'yes', because predefined analyses are more ready available and also visual.

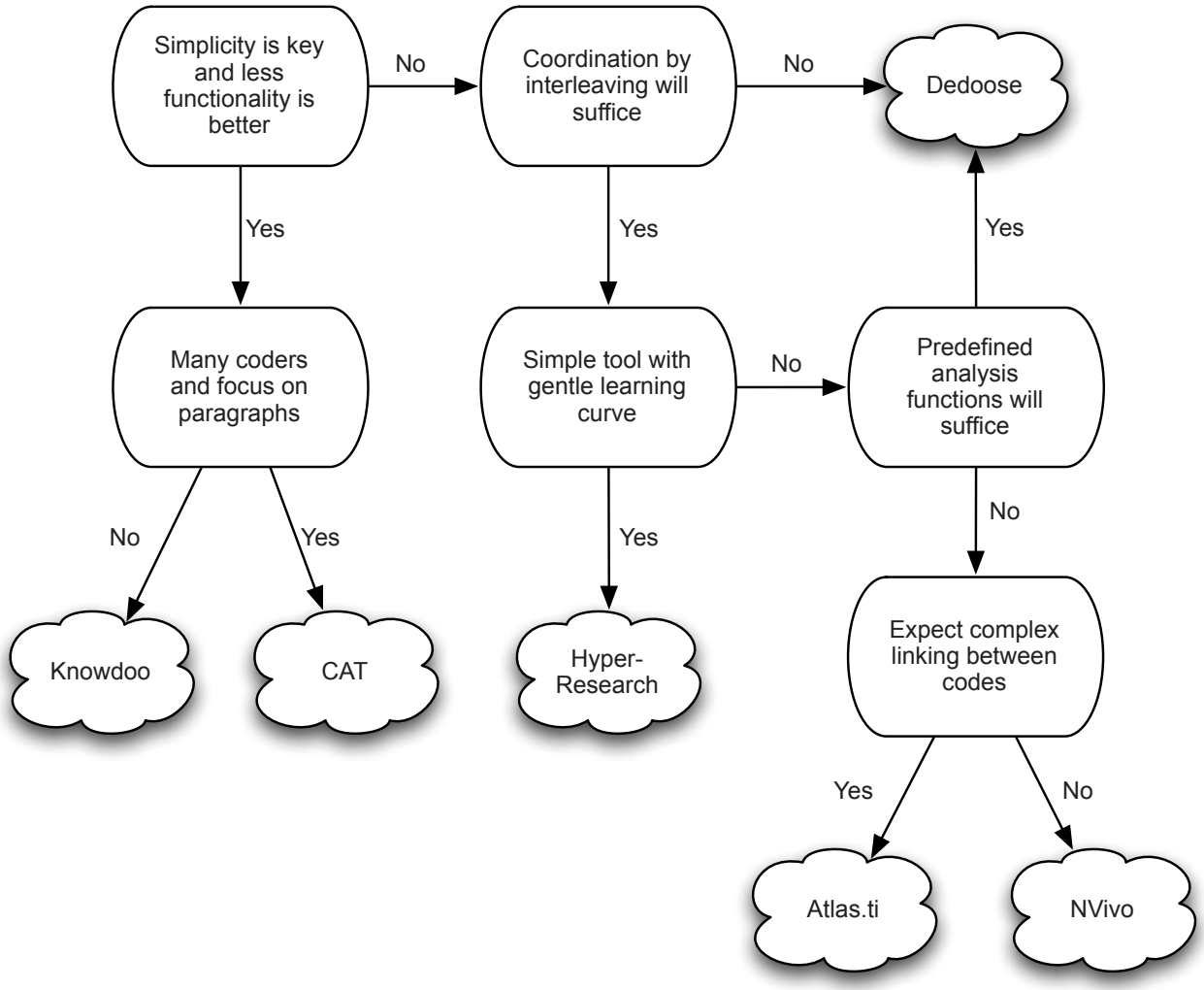


Figure 9: A decision tree for collaborative coding applications

If you find that you need to be able to formulate your own queries you have to choose between Atlas.ti and NVivo. From a functional point of view these two applications do not differ in significant ways—though there is much difference between their design styles and their usability. The decision tree is of course a simplification, and you may want to get an impression of the applications by following the links to the demo for each application.

Steps

1. Consider the team's research setting in terms of who you are, what your experience with coding is, your experience with collaboration is, how dependent you are on each other in the analysis, what your coding task is, etc.
2. Use the decision tree in figure 9 to make an initial decision on which application will be appropriate for your team. Also be informed by the comparative summary in table 3 and ideally also the previous assessments.
3. Consider how your choice will work in the long run also after the coding task at hand. The learning of a coding application in particular in a collaborative setting should be seen as a investment where the cost of coding is reduced considerably in the second and third task. If in doubt then return to step 1.

4. Spend time on the tutorials for your selected application. All members of the team should be familiar with the basics of the application before commencing on the coding task.
5. Do not just start in an ad hoc way. Plan your analysis, how you want to code, what you will be looking for, etc.—but be ready to change your plans.

Method

The research method behind this assessment is basically design science research. It has been conducted through design workshops.

Design Science Research

Design science research has the advantage of being a research activity conducted with the purpose of developing an artefact and at the same time acquire research knowledge about its features and its potential usage (Hevner et al. 2004; Peffers et al. 2007; Hevner 2007).

The long-term purpose of the research is to develop a research-based understanding of applications to support collaborative qualitative data analysis for a team of researchers. This white paper is part of this and the more specific focus is on evaluating the applications as design artefacts.

Design Workshop

A design workshop was organised with the purpose of answering the question we started this white paper with: What are the desirable and feasible features of applications to support collaborative qualitative data analysis for researchers? The focus was thus in design science research terms to evaluate the applications as artefacts.

Participants in the workshop were experienced qualitative researchers who had all considerable with coding using either Atlas.ti or NVivo. In the workshop we went through the details of Atlas.ti, NVivo, HyperResearch, and Dedoose. This was done by using the applications on a laptop connected to a projector to allow that all participants had the same view all the time. The structure of the workshop followed the functions in table 2. For each function we discussed and assessed in which ways and to what extent the application supported the function. For Atlas.ti and NVivo the participants shared their knowledge and took turn in showing how a particular function works in the application. For HyperResearch and for Dedoose the workshop organiser had the active role in presenting the different functions.

The assessments that came from the workshop were documented continuously at a white board. The documentation of the workshop was thus a recording of the white board by the end of the workshop.

In a follow-up these preliminary results were presented to a group of 8 qualitative researchers about to start on a coding task. This session further elaborated and consolidated the findings. This did not lead to any changes in the assessment, but it did improve the way in which the

assessment was formulated.

Conclusion

This LA2020 project has taken on the task of assisting teams of qualitative researchers in figuring out how to collaborate through coding of their empirical data. The application CAT was initially considered as a potentially useful collaborative tool, but that idea had to be abandoned due to its limited functionality. The other initial idea in the project was to extend the open source software KiWi that had already been developed for collaborating on sharing knowledge through a semantic web application. This idea was soon overtaken by Dedoose that independently had been developed. Dedoose has included many of the features which we were planning for an extended KiWi. Hence, it became more feasible to focus on Dedoose in stead.

For the sake of focussing attention on *both* coding and collaboration the more traditional coding tools like NVivo, Atlas.ti and HyperResearch were also included in the assessment.

In this white paper the intention has been to provide sufficient information for would-be users of a collaborative coding application that they can make informed choices. The assessments focus on the overall impression stemming from the evaluation sessions rather than providing all the details of all the functions in all the applications. The major differences are summarised in table 3 which in turn is a main contribution of this white paper. The following road map is also a contribution to the practical side of what a team of researchers should attend to in their implementation of an collaborative coding application.

Glossary

Code: A code is a tag or label for assigning units of meaning to the qualitative data. Codes are usually linked to quotes of varying size, e.g., words, phrases, sentences or paragraphs (Miles & Huberman 1994, p. 56).

Link: A relationship between a quote and a code or between two codes.

Quote: A segment of a data source. E.g., in text it may be a word, a phrase, a sentence, a paragraph or have scope somewhere in between.

Data source: The body of empirical material. This is traditionally contained in a set of text file, but can be contained in files of varying format, e.g., image, PDF, audio, and video.

Memo: A brief explanatory text that can be associated with a quote, a code, or a link.

Bibliography

Atlas.ti.

Main web site: http://www.atlasti.com/index.html.
Demo: http://www.atlasti.com/uploads/media/QuickTour_a7_en.pdf.
Tutorial: http://www.atlasti.com/tutorials.html.

Bryman, A., (2008). *Social Research Methods*, 3rd edition, Oxford University Press, Oxford.

CAT.
Main web site: http://cat.ucsur.pitt.edu/.
Demo: http://www.screencast.com/t/ifo3uS8dR. The demo is also a tutorial for a small part of the application.

Charmaz, K., (2006). *Constructing Gounded Theory: A practical guide through qualitative analysis*, Sage Publications, Thousand Oaks.

Cresswell, J. W., (2009). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*, 3rd edition, Sage Publications, Thousand Oaks.

Dedoose.
Main web site: http://dedoose.com/.
Demo: http://dedoose.com/LearnMore/FeaturesAndBenefits.aspx.
Tutorial: http://dedoose.com/LearnMore/VideoTour.aspx.

Hevner, A. R., (2007). The three cycle view of design science research. *Scandinavian Journal of Information Systems*, 19(2): 87–92.

Hevner, A. R., S. T. March, J. Park & S. Ram, (2004). Design science in information systems research. *MIS Quarterly*, 28(1): 75–105.

HyperResearch.
Main web site: http://www.researchware.com/.
Demo: http://www.researchware.com/products/hyperresearch/quick-tour.html.
Tutorial: http://www.researchware.com/support/online-training/380-hyperresearch-basics.html.

Knowdoo.
Main web site: www.knowdoo.org.
Demo: http://knowdoo.net/demo/.

Miles, M. B. & A. M. Huberman, (1994). *Qualitative Data Analysis*, 2nd edition, Sage Publications, Thousand Oaks.

NVivo.
Main web site: http://www.qsrinternational.com.
Demo: . http://www.youtube.com/watch?v=K3wdeZUZGVY&lr=1
Tutorials: http://www.qsrinternational.com/support_tutorials.aspx.

Peffers, K., T. Tuunanen, M. A. Rothenburg & S. Chatterjee, (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3): 45–77.

Strauss, A. & J. Corbin, (1990). *Basics of Qualitative Research*, Sage Publications, London.